Quantum informatics and quantum computing
Short introduction

Jakub Pilch
3yakuya

Contents

I Introduction

Definitions and intuitions related to quantum computing are often against those related to classical theory of computation and computational complexity. To make this topic a bit easier to digest I decided to introduce a few names and rules that are crucial to understand some ideas behind quantum computing.

1. Strong Church – Turing Thesis

To start thinking about computability in general and the role of quantum computing in information science we need to define the most basic computing machine.

The Turing machine

An abstract machine consisting of:

- An infinite tape, divided into fields containing single signs from the input and work alphabets
- A head capable of reading or writing a single sign from/on the tape and moving one field left or right in one operation
- The move function, determining explicitly (in case of a deterministic Turing machine) the behavior of the machine in a given configuration (in other words: if head is above a given sign, what should be written to the tape and in which direction should the head move.)

The Turing machine has its special place in the Theory of computation thanks to such thesis:

The Church-Turing Thesis:

Every intuitively computable problem is solvable by a Turing machine.

Of course this thesis is not verifiable (for which one reason is the word *intuitively* used in it,) but for years it has been treated as one of the fundamental rules of computer and information science. It should be noticed that it treats about computability in general, without referring to complexity.

The universal Turing machine is one that can simulate any other Turing machine by reading the simulated machine's configuration first. It is easy to guess that such simulation comes at a price of complexity, and the computation itself may take polynomially or exponentially more time than it would on a dedicated machine. In terms of computability in general, the universal Turing machine is equivalent to standard deterministic Turing machine.

Now let's imagine a generalization of a Turing machine and extend the standard three parameters from the definition by another one: a random bit generator (think about it as, for example, a machine capable of throwing a coin and checking its state.) Such a machine is called a probabilistic Turing machine. Similarly to UTM, it can be proven equivalent to classical deterministic Turing machine in terms of computability. However, if we take complexity into account there are quite a lot of differences. There are known problems that we can solve in polynomial time on a probabilistic machine, but the fastest known deterministic algorithms require exponentially more. Knowing the above, we can introduce another definition:

Classical Strong Church-Turing Thesis:

A probabilistic Turing machine can efficiently simulate any realistic model of computation.

Notice an important difference to the previous similar definition: we talk about a *realistic computing model*. For example: we can build dedicated machines that would solve their problems much faster than universal ones, but this may include using some rules of nature that would take polynomial or exponential time to simulate. In the thesis above we mean any model that can be built in accordance to the laws of physics and count all the resources used by a machine.

The classical Church-Turing thesis introduces a new problem – physical limitations. As commonly known, classical physics rules are not doing much good in describing the quantum effects. In such a small scale we need to use quantum physics to adequately describe reality. This means that the thesis presented above does not include one of the possible computation models – the quantum computer. To start talking about quantum computing we need to take one more step and use a more general, important thesis:

Quantum Church-Turing Thesis:

A quantum Turing machine can efficiently simulate any realistic model of computation.

Introducing above definitions is a good base to justify the value of studying quantum computing. If a quantum Turing machine is capable of *efficiently* simulating any realistic computations then it can also solve many problems that are very expensive for a classical deterministic Turing machine using acceptable (usually polynomial) time.

As for today (2015) there is no proof that any quantum computation can't be simulated efficiently using deterministic machines. However, there are known problems for which the fastest deterministic algorithms require exponential resources, while there are quantum algorithms capable of solving those in polynomial time.

2. Circuit model of computation

In order to continue thinking about quantum computing (and especially in the context of building a quantum computer simulator) an important step is to introduce an alternative model of computation – the circuit model. Instead of talking about Turing machines we will discuss complete sets of simple reversible circuits. Let us define what do we mean by *circuit*:

Circuit

A simple device made of gates performing logical operations on binary values carried through connections.

Our circuits need to fulfill some requirements in order to be usable like we expect. First, we only consider acyclic circuits, so the binary values are only carried in one direction through our circuit and there are no loopbacks. Second, our gate set need to be universal:

Universal gate set

A set of gates is said to be universal for classical computation if for any positive integers n, m and a function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ a circuit computing f can be built using only gates from this set.

Third, in a single time unit one bit carried through some connection can enter at most one gate, and an operation performed by a single gate takes a single unit of time. Fourth, we require our gates to perform reversible operations:

Reversible gates

A gate is reversible (is performing a reversible operation) if for any output we can determine the original input.

A typical example of an universal gate set for classical computation are NAND and FANOUT gates. Using only these two we can build circuits computing any computable functions. If we limit ourselves to reversible gates, however, things get a bit more complicated because we can not achieve universality through using only single and two-bit gates. Let us introduce a 3-bit gate then:

Toffoli gate

It works as follows: if (and only if) first two input bits are set to 1, the third input bit is negated on the output. Nothing changes otherwise.

A set consisting only of a single Toffoli gate (which is reversible) is complete for classical computation as long as we are allowed to use constant-bit generators.

While with Turing machines we discussed complexity in terms of time of memory required for computation, in circuit model we will talk about the number of gates used and circuit's *depth* (the number of sequential rows of parallel gates.)

Just like with Turing machines, for Circuit model we can also introduce a probabilistic element: a random bit generator capable of returning a random binary value in a single time unit (without any dependence on any input.)

To describe a state of a deterministic machine we could simply present all binary values from a specific piece of the circuit. In probabilistic circuits we are going to use probability vectors, related to occurrences of specific values in a part of the circuit.

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$$

The above should be treated as follows: in the measured fragment of the circuit a bit may have the value of 0 with probability p_0 and the value of 1 with probability p_1.

As we are using vectors to describe states in the circuit, we would like to describe operations performed by gates in a form that is applicable to value vectors. For this reason in an algebraic representation of the circuit model of computation we will use matrices to represent gates. The *NOT* gate is a simple example:

$$NOT\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{pmatrix} p_1 \\ p_0 \end{pmatrix}$$

Note that it performs a reversible operation, so it fulfills our requirement.

Finally, to describe a state of a multi-bit probabilistic circuit we need to present all possible value combinations along with probabilities of their occurrence. For example let us assume we want to describe a state of a circuit consisting of only two connections and nothing else. Probability of reading 1 is p_1 for the first connection and q_1 for the second one. Similarly, probabilities of reading 0 are p_0 and q_0. There are four possible states of such circuit, and this can be presented in a vector form:

$$\begin{pmatrix} p_0q_0 \\ p_0q_1 \\ p_1q_0 \\ p_1q_1 \end{pmatrix}$$

Because we can describe states consisting of multiple bits as vectors, we should also be able to use matrices to describe operations performed by multi-bit gates. As an example let us introduce a simple gate (as it will turn out, its quantum version is very important):

***CNOT* gate**

It takes control and target input bits. If the control bit is set to 1, the target bit is negated on the output. Nothing is changed otherwise.

The *CNOT* (controlled *NOT*) gate can be presented as a matrix as follows:

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

From above it is pretty clear that a computational circuit can be described algebraically, using vectors of value-occurrence probabilities and matrices representing logic gates.

The circuit model of computation is an important step towards understanding the idea of quantum computing.

3. State of a quantum system – qubits

In 1930s the world of physics went through some kind of a revolution, after it turned out that classical rules that have been known and used for years are not enough to describe current observations. A new model in physics started to be created, one that by definition should be consistent with observed experiments to allow physicists understand them better.

Similarly in information science, willing to describe and process information stored in quantum systems we are forced to accept some rules of quantum mechanics. Our approach to information processing in general is probably going to change. It appears that information science is going to go through a transformation similar to the one that happened to physics around 1950s.

Classic information and computer sciences made us used to binary data representation. According to words of Claude E. Shannon, author of the theory of information, nearly any information can be represented as a finite word made of 0s and 1s. A single bit is capable of holding one of two states at once, true (1) or false (0), and a series of such bits can describe any classical information system.

Quantum systems are described by …quants of information, meaning just some discrete states. Thanks to their discreteness they can be presented in a vector form:

State Space Postulate

Any physical system can be completely described by a unit state vector in state space being a Hilbert space H.

The dimension of Hilbert space mentioned above depends on degrees of freedom of the given system. From information science point of view, we are interested in systems with two possible states, representing 0 and 1. An example physics implementation could be a particle spin, which is possible to be described by a unit vector in Hilbert space. We can represent spin-up as $|0\rangle$ and spin-down as $|1\rangle$ using the Dirac bra-ket notation.

From the description above it may appear that the quantum state's representation does not differ from classical one's – for the simplest information pack we have two possible states, true or false. However, in quantum systems we may deal with *superposition* of states 0 and 1, meaning the state is *simultaneously 0 and 1*.

In order to grab some intuition about this let us imagine some probabilistic bit, which value can not be determined for sure until measured explicitly. We only know that it can be in one of the states 0 or 1 with probabilities p_0 adn p_1 accordingly, where $p_0 + p_1 = 1$ (because our bit is for sure either 0 or 1.) The general state of this probabilistic bit could be written as $p_0 + p_1$.

Superposition of two possible quantum states is goes a step further. The amplitudes (coefficients) of possible quantum states may be complex[1,2]. What is more, complex amplitude a may be decomposed as $a = e^{i\theta}|a|$, and according to state space postulate our system can be described by a normalized vector in Hilbert space. From physical perspective, the global phase (rotation about the real value) does not matter, so we can describe our systems in a bit easier form:

General quantum bit (qubit) state representation

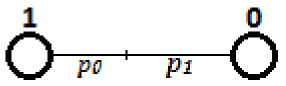
$$\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{-i\varphi} sin\left(\frac{\theta}{2}\right)|1\rangle$$

In above representation $|0\rangle$ and $|1\rangle$ are base vectors of our state space, θ is the angle between the basic state axis and our state vector (in other words: width on the Bloch sphere[1,2]) and φ is the angle of state vector's relative phase (length on the Bloch sphere.) We write $\left(\frac{\theta}{2}\right)$ instead of just θ because we want to treat vectors $|0\rangle$ and $|1\rangle$ as orthogonal from Cartesian perspective (the angle between $|0\rangle$ and $|1\rangle$ is π, so using the formula above we will make $|1\rangle$ orthogonal to $|0\rangle$ from Cartesian point of view.) Quantum bits, which states we describe as above are called shortly qubits.

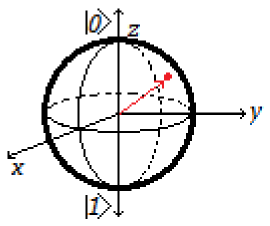
Some intuition about bits, probabilistic bits and qubits may be formulated as follows:

- **State of a classical deterministic bit** can be graphically represented as selected one of two points (0 or 1),

- **State of a classical probabilistic bit** can be represented as a point on a unit section terminated by states 0 and 1, distanced by p_1 from state 0 and p_0 from state 1,



- **State of a quantum bit (qubit)** can be represented as a point vector in a three-dimensional sphere (called the Bloch sphere).



In its nature qubit can be in one of an infinite numbers of states[7] (which is related to an infinite number of points on a sphere), but when measured it will always reveal either 0 or 1.

4. Closed systems and quantum entanglement

Evolution Postulate

Evolution of a closed quantum system in time is described by a unitary transformation.

From above postulate[1] it is clear that for any evolution (series of transformations) of a closed quantum system there exists an unitary operator U such that if we mark the input state as $|v_1\rangle$, a and the output state as $|v_2\rangle$, then $|v_2\rangle = U|v_1\rangle$. When talking about quantum computing, unitary operators transforming single qubits will be called **single-qubit quantum gates.** It is worth noticing that usage of unitary operators is a guarantee of reversibility of our transformations.

In accordance to what has been mentioned earlier (in part 2, on circuit model of computation), gates can be represented as matrices. For a single qubit (in Hilbert space of dimension 2) a gate can be represented by a 2 x 2 matrix. Any gate is a set of transformations for any possible input. Just as described for circuit model in general, computation performed by any gate can be represented as a product of the input state (represented by a column vector) and the transformation matrix.

Until now, we have not mentioned the behavior of a multi-qubit system. Using the information above we can describe any system consisting of many isolated qubits, but such model is not quite practical for any real computation. In order to consider computing in general we need to be able to describe how qubits behave when they are interacting with each other. The following postulate comes to help us:

System Composition Postulate

State of a composite quantum system is the tensor product of its quantum sub-systems' states. If $|v_1\rangle$ is the state of the first sub-system and $|v_2\rangle$ is the state of the other one, then the joined state is $|v1\rangle \otimes |v2\rangle$.

Above postulate can be scaled through induction to a quantum system made of *n* joined subsystems.

Not every multi-qubit system can be represented by a tensor product, though. Two isolated qubits form two separated closed systems, and so we can write their combination as above. However, if qubits are not isolated from each other they may interact and create states that are impossible to write as tensor products. In such case, we say that those ***qubits are entangled***. Let us present the following example:

$$|\boldsymbol{v}\rangle = \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}|\mathbf{00}\rangle + \frac{\mathbf{1}}{\sqrt{\mathbf{2}}}|\mathbf{11}\rangle$$

One can show that there are no such coefficients (amplitudes) that would allow to write the above as a tensor product of two states[1,8]. This example is one of the EPR pairs, named after Einstein, Podolsky and Rosen, who considered such couples in their studies of quantum mechanics.

Finally, when we measure the state of a qubit (willing to extract information from it) the system is no longer closed. This means its state is destroyed, as an open system is not of much use for computation. The actual behavior during measurement is presented by the following postulate[9]:

Quantum Measurement Postulate

The measurement of a quantum state can be represented by a set of operators $\{M_m\}$ operating on the system state space. The probability p of measuring the value m during measurement of the state $|v\rangle$ is

$$p(m) = \langle v|M_m^t M|v\rangle$$

After the measurement system is left in state

$$|v'\rangle = \frac{M_m|v\rangle}{\sqrt{p(m)}}$$

Above postulate implies (among others) that if we perform multiple consecutive measurements of the state without changing it in between, we are guaranteed to get the same value every time.

5. Implementation of a qubit

As described before, we require a few properties from our qubits:

- Possibility of reading 0 or 1
- Possibility of being in superposition of 0 and 1
- Possibility of getting entangled with other qubits

There are still some more practical requirements that a qubit would need to fulfill in order to be usable:

- Possibility of manipulation (setting the input, processing and resetting the state)
- Stability
- Feasibility

Though at first these requirements may appear to be obvious, different approaches may meet different difficulties.

One of possible qubit physical implementations is a photon, which polarization may represent the state. Another one is a spin of an electron, which can also be described by a vector in the Hilbert space of size 2. Yet another approach is selecting particles in which the energetic gap between the first two levels is small compared to next differences (it can be excited or not.) The direction of current flow in a tiny superconductor loop, or turn of the tiny magnetic field created by such circuit are also considered.

One of the most challenging problems while implementing a qubit is its stability. Quantum phenomena are only observed in nano-scale, so even the tiniest particle or electromagnetic wave may cause decoherence (bring the qubit to one of its basic states, destroying the information.) What's more, in accordance with the postulate from chapter I-4, we need to ensure that qubits in the quantum processor are isolated from other ones, in particular from those outside of the machine. If we fail to fulfill these requirements properly it may turn out that our system is not closed, which makes it unusable for computation (as unitary operators do not apply then.)

Knowing the requirements and limitations, current attempts to build quantum processors rely on heavy lead shielding and big coolers and radiators, cooling the computing cores down to below 1K.

D-Wave, the company known for research in quantum computing, uses last of qubit implementations mentioned above[6]. It should be noticed that D-Wave processors were designed to perform a specific algorithm called *Quantum Annealing* (which, analogically to *Simulated Annealing* is a global optimization algorithm.) As for now, the company is working to increase the number of qubits that can be entangled together, to get the machine more applicable to real-life problems.

Microsoft has been pursuing an entirely different approach for over 10 years now[5]. Directed by Michael Freedman, they search for some particular kind of quibt called topological qubit. The idea itself could be the topic of a separate paper, but it should be mentioned that if they succeeded, super-stable and noise-resistant qubits could be made. As for today there is some research in finding particles called *non-Abelian Anyons*, which existence is the base for the theory behind topological qubits.

II Quantum model of computation

Understanding the mentioned earlier circuit model of computation makes the quantum model of computation much more digestive. Of course, the first was related to classical computation, where a bit different rules apply, but introduced reversibility of computation, which is very important from our point of view.

The quantum model of computation (more accurately: the gate quantum model) introduces a vision very similar to the circuit model: we have some input qubits that are then processed by some gates. One need to notice, however, that qubits may not go through some wires or connections from gate to gate because of their possible implementations. Instead, we can imagine performing a series of operations on some kind of quantum register holding qubits, until we reach the desired result.

There is one more significant difference that is unique to quantum computing: the measurement. If we want to gain anything from our transformations of the quantum state, we need to extract the classical state at the end of computation. As it was mentioned earlier, measuring the quantum state destroys it irreversibly by setting every qubit to either 0 or 1 (so all the information hidden in relative phase etc. will be gone.) What is more, for some computations the measured output is correct with some probability only! This means we need to run some algorithms multiple times, so that our results will be more reliable.

The chapter below introduces concepts tightly related to quantum computing, describes some mechanisms behind and explains how they work.

1. Single-qubit quantum gates

Quantum gates are supposed to perform some desired transformation of the input sate, under some conditions. As we know from earlier parts of this paper, quantum state can be written using vectors. Operations performed on the state are then just some transformations of the state vector.

Typically, quantum gates are represented graphically on quantum circuit diagrams using rectangles marked by gate's symbol:

1 Quantum gate G

Let us have a closer look. Remember the intuition from the end of chapter I-3? The state of a single qubit can be visualized by a point vector in the Bloch sphere. Now let us take an example single-qubit gate G. The transformation performed by such a gate can be seen as some kind of rotation of the point vector representing the input state. For example, the NOT gate is basically rotating the vector around the x axis in the Bloch sphere.

There are four important single-qubit quantum gates, called Pauli Gates:

$I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
$Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Notice that all of them are related to some rotation of the vector state. This implies we can write their transformations as functions:

$$\boldsymbol{R_x(\theta) \equiv e^{\frac{-i\theta X}{2}}}$$

$$\boldsymbol{R_y(\theta) \equiv e^{\frac{-i\theta Y}{2}}}$$

$$\boldsymbol{R_z(\theta) \equiv e^{\frac{-i\theta Z}{2}}}$$

Knowing the above we can conclude:

Every single-qubit quantum gate can be represented by a sequence of rotations around the axes of the Bloch sphere:

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

Of course this does not mean that we will write all single-qubit gates as sequences of rotations (though all of them can be interpreted as such.) The following two gates have some special place in the quantum model of computation:

Hadamard gate (usually marked ***H***) maps the base vectors as follows:

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

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

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

Notice that Hadamard gate is its own reverse ($H = H^{-1}$).

The $\boldsymbol{\frac{\pi}{8}}$ gate (usually marked ***T***) maps the base vectors as follows:

$$T|0\rangle = |0\rangle$$

$$T|1\rangle = e^{i\frac{\pi}{4}}|1\rangle$$

$$T = \begin{bmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

The equivalence of matrix forms written above is true without regard to global phase (which is not important from computational perspective.)

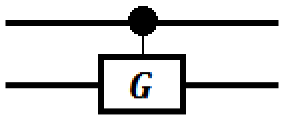
2. Controlled quantum gates

Just like for the circuit model, in quantum model we want to use controlled gates that perform some transformations under certain circumstances. The *CNOT*[1,4] gate presented in chapter I-2 is an example of a controlled gate that negates the target bit only if the control bit is set to 1. Note that in quantum computation the gate may operate on the state being in some superposition.

Given a single-qubit gate G, we can define its controlled version $c\text{-}G$ as follows:

$$c\text{-}G|0\rangle|v\rangle = |0\rangle|v\rangle$$
$$c\text{-}G|1\rangle|v\rangle = |1\rangle G|v\rangle$$

When drawing diagrams, controlled gates are represented like below:



2 Controlled quantum gate G

3. Entangling quantum gates

Entangling gates

Two-qubit quantum gate is entangling if for some input state $|v\rangle|u\rangle$ being the tensor product, the output state is no longer a tensor product (the qubits get entangled.)

CNOT gate is an example of an entangling gate, which can be easily observed[8]:

Let us take some quantum state $(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle = \alpha|00\rangle + \beta|10\rangle$, where $\alpha, \beta \neq 0$. Notice that such state can be provided as an input to the *CNOT* gate if the control qubit is in a superposition $(\alpha|0\rangle + \beta|1\rangle)$, and the target qubit is in state $|0\rangle$.

Remember the matrix representation of the *CNOT* gate?

$$CNOT = \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 0 \end{bmatrix}$$

If we transform our input state with CNOT we get $CNOT(\alpha|00\rangle + \beta|10\rangle) = \alpha|00\rangle + \beta|11\rangle$. The output state is entangled, which we can prove by contradiction:

Let us assume that $\alpha|00\rangle + \beta|11\rangle$, $\alpha, \beta \neq 0$, is not entangled. This means we can represent it by a tensor product of two states:

$$\alpha|00\rangle + \beta|11\rangle = (k|0\rangle + l|1\rangle) \otimes (m|0\rangle + n|1\rangle) = km|00\rangle + kn|01\rangle + lm|10\rangle + ln|11\rangle$$

The line above implies that $\{km \neq 0,\ kn = 0,\ lm = 0,\ ln \neq 0\}$. These equations can not be fulfilled simultaneously, so the line above is wrong, which proves the original assumption was wrong.

4. Universal set of quantum gates

Just like with classical computation, in quantum computing we are interested in performing complicated algorithms, operating on n qubits. At the same time, we can't build gates that take more and more qubits. We need to find a finite set of gates from which we could build any computing circuit.

Let us fit our definition of universal set of gates to the quantum model of computation:

Universal set of quantum gates

Set of quantum gates is universal if for any natural number $n > 0$, any n-qubit unary operator can be approximated with given precision using only quantum gates from this set.

Earlier theorem (from chapter II-1) about presenting single-qubit gates as sequences of rotations can also be a bit generalized so that it treats about gates others than Pauli gates.

A set of two single-qubit gates is universal for single-qubit gates if:

1. Axes around which the gates perform rotations are not parallel,
2. Gates perform rotations by angles α and β such that $\alpha, \beta \in [0, 2\pi]$ and $\frac{\beta}{\pi}$ and $\frac{\alpha}{\pi}$ are irrational real numbers.

It can be proven that a pair of gates HTHT and THTH fulfills the theorem above, meaning that {H, T} is an universal set of single-qubit quantum gates.

Because we need to approximate results of any n-qubit gates, our universal set need to include at least one gate operating with two or more qubits. Again, the *CNOT* gate operating on two qubits comes to help us.

Knowing above theorems and definitions we can state the following:

A set of quantum gates consisting of all 1-qubit gates and any 2-qubit entangling gate is universal.

Using two theorems from this chapter and the theorem on entangling gates from chapter II-3, we can conclude that set of quantum gates {*CNOT, H, T*} is universal.

5. Bell states

As we are going to see later on, Bell states are special ones that play an important role in quantum computing. They form a group of four quantum states, spanning an orthonormal basis:

$\lvert\beta_{00}\rangle = \frac{1}{\sqrt{2}}\lvert 00\rangle + \frac{1}{\sqrt{2}}\lvert 11\rangle$	$\lvert\beta_{01}\rangle = \frac{1}{\sqrt{2}}\lvert 01\rangle + \frac{1}{\sqrt{2}}\lvert 10\rangle$
$\lvert\beta_{10}\rangle = \frac{1}{\sqrt{2}}\lvert 00\rangle - \frac{1}{\sqrt{2}}\lvert 11\rangle$	$\lvert\beta_{11}\rangle = \frac{1}{\sqrt{2}}\lvert 01\rangle - \frac{1}{\sqrt{2}}\lvert 10\rangle$

As you may have noticed, all of above states are EPR pairs.
To change the basis of two qubits *i* and *j* to the Bell basis β_{ij} we need to transform qubit *i* using the Hadamard gate, and then perform *CNOT* targeting *j* with *i* being the control qubit. The above sequence in reverse would result in the opposite effect (change the basis from Bell basis to standard computing basis.)

III Quantum protocols

From computer science perspective, the whole theory of quantum computing is only useful when applied to some problems. People have been trying to understand quantum computing and spending millions of dollars trying to build quantum computers because of some quantum algorithms and protocols that are far beyond the reach of current classical machines.

The next chapters are going to introduce some of the most-known quantum protocols along with their problems and potential applications. While these are not tightly related to building a quantum computer simulator, their educational value is significant and understanding them may turn out to be very helpful later.

1. Superdense coding

Transmitting information plays the key role in information technology. From the easiest phone calls, through online shopping, to Internet banking. Information is sent often and in great numbers. One of the most important problems of telecommunications are securing the data from undesired access and saving data in an efficient way.

In classical information theory we can say that information is sent efficiently if selected encoding does not create too much overhead. It is pretty safe to say that if we could send information without any overhead and still guarantee its correctness this would be the perfect coding. However, using quantum mechanics opens a world of new possibilities. It enables as to send two classical bits by sending only a single qubit through a quantum communication channel. This is what we call superdense coding.

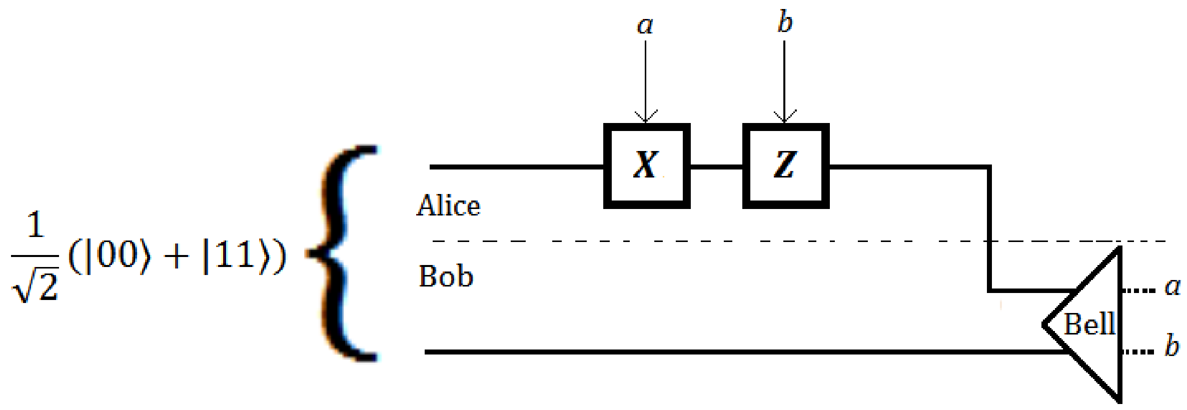
Using some naming convention typical for cryptography, let us assume that Alice wants to send two classical bits for Bob. As we want to utilize quantum mechanics, we assume that sender and receiver are connected through a quantum communication channel (which implementation may vary, for ex. it could be optical fiber for photons.) We also require Alice and Bob to have one qubit from an entangled pair in Bell state $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$[9] each. Note that these are no trivial requirements. Entangling two qubits on its own may cause problems, as we need to assure proper isolation etc.

Willing to send two classical bits, Alice can send Bob one of four values: 0, 1, 2 or 3. To send 0, Alice performs an Identity operation on her qubit from the shared pair. To send 1 – she transforms her qubit using Pauli X gate. To send 2 – she applies Z to her qubit. To send 3 – she performs Z•X (first she applies X and then Z gate.) Because Alice's and Bob's qubits are entangled, these operations should be considered to operate on a pair of qubits:

To send	Transform	State $|\boldsymbol{\beta_{00}}\rangle$ is transformed to:
0d = 00b	$I \otimes I$	$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\beta_{00}\rangle$
1d = 01b	$X \otimes I$	$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\beta_{01}\rangle$
2d = 10b	$Z \otimes I$	$\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\beta_{10}\rangle$
3d = 11b	$Z \cdot X \otimes I$	$\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\beta_{11}\rangle$

After transforming her qubit appropriately Alice sends it to Bob, so that he has two entangled qubits in one of four Bell states. He can then measure their state in the Bell basis to obtain two classical bits sent by Alice.

Quantum circuit performing superdense coding may be represented as follows:



3 Superdense coding - circuit

2. Quantum teleportation

Considering the huge amount of information carried by a qubit (not to mention an entangled pair of qubits) it may not come as a big surprise that we can "pack two classical bits into a single qubit." How about the reverse operation? Could we send the state of a qubit by sending only two classical bits? It appears to be impossible, especially if we consider that even the *sender* does not know the actual state of the qubit (as they cannot measure it without destroying its state.) What is more, even if we somehow did collect all the information about our qubit's state, we would require a potentially infinite number of bits to store this information.

Quantum teleportation is a way to do what appears to be impossible in this case. Using this protocol Alice can make Bob's qubit have *exactly the same state* as her before teleportation, and all this by sending only two classical bits. In other words, this protocol enables sending a quantum state between clients not connected by any quantum communication channel.

Just as with superdense coding, quantum teleportation requires that both sender (Alice) and receiver (Bob) have a single qubit from an entangled pair $|\beta_{00}\rangle$ each. In order to send the state of some qubit $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ to Bob, Alice needs to measure this qubit together with her qubit from the shared entangled pair in the Bell basis. This will get her two classical bits a and b, and the three processed qubits after the measurement will be left in one of the following states with probabilities equal $\frac{1}{4}$ for each:

$$|\beta_{00}\rangle|v\rangle$$
$$|\beta_{01}\rangle(X|v\rangle)$$
$$|\beta_{10}\rangle(Z|v\rangle)$$
$$|\beta_{11}\rangle(XZ|v\rangle)$$

This knowledge allows Bob to perform necessary transformations (depending on classical bits a and b) and transform his state to the state of qubit $|v\rangle$:

a	b	Operation
0	0	I
0	1	X
1	0	Z
1	1	$Z•X$

After that his qubit will be left in state originally held by $|v\rangle$. Alice's quantum state was teleported to Bob.

Correctness of the above can be proven[9]. Measuring in the Bell basis is equivalent (in terms of returned classical values) to performing $(CNOT \otimes I)$ followed by $(H \otimes I)$ on the tensor product $|v\rangle \otimes \beta_{00}$. This will leave us in the following state:

$$\frac{1}{2}(\alpha|000\rangle + \beta|001\rangle + \beta|010\rangle + \alpha|011\rangle + \alpha|100\rangle - \beta|101\rangle - \beta|110\rangle + \alpha|111\rangle) =$$

$$\frac{1}{2}\big(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\big)$$

After the measurement in the standard computation basis the state of the first two qubits will be set, and so the third qubit will be left in one of the following states:

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

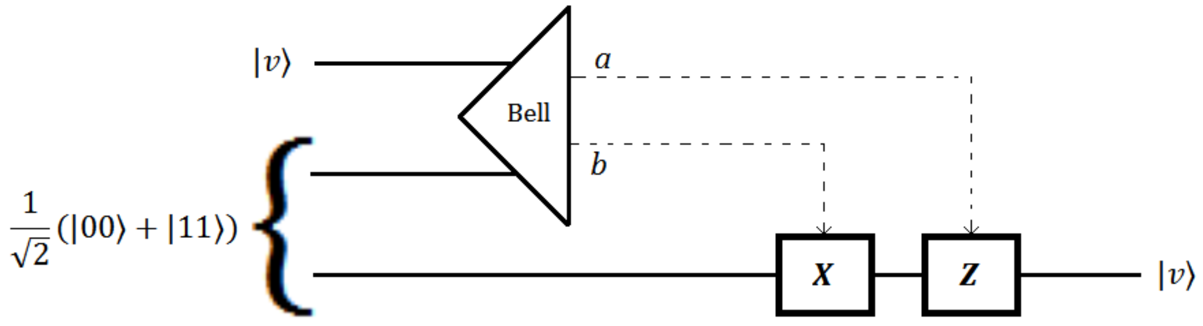
$$(\alpha|1\rangle + \beta|0\rangle)$$

$$(\alpha|0\rangle - \beta|1\rangle)$$

$$(\alpha|1\rangle - \beta|0\rangle)$$

The above can be transformed by I, X, Z or $Z•X$ appropriately to get state $\alpha|0\rangle + \beta|1\rangle$, which is the original state of $|v\rangle$. Alice has indeed teleported her quantum state to Bob by sending only two classical bits.

Circuit performing quantum teleportation can be represented as follows (dotted line represents classical communication channel):



4 Quantum teleportation - circuit

IV Quantum algorithms

The goal of the part below is to present how quantum model of computation can be used for solving problems. Additional introductory chapter is intended to compare classical probabilistic and quantum computing, and to explain and underline the differences between the two. There are also some chapters that introduce important theoretical concepts that are crucial to understanding more complex quantum algorithms that follow.

1. Quantum vs. probabilistic computation

The word "nondeterministic" is often used in quantum computing context, which often misleads people to think that quantum computing is simply like classical probabilistic computing. This approach is very wrong, as quantum mechanics provides some behaviors that are impossible to observe for probabilistic computing. Quantum entanglement, superposition and quantum measurement are not observed for any classical computations.

We will use a simple example to show the difference between the two models mentioned above. Let us take a circuit consisting of two Hadamard gates, run on the input state $|0\rangle$. If we measured the state after each of the gates our quantum circuit would behave just like the probabilistic model (with no hidden data, destroyed on measurement.) Just after the first gate the measured state will be $|0\rangle$ or $|1\rangle$, each with probability equal to $\frac{1}{2}$. After the second gate, just before the second and final measurement, the state will be $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ or $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, each with probability of $\frac{1}{2}$.

Now let us perform just a single measurement at the end, after both gates. The final state of the system just before the measurement can be written as follows:

$$|v_{final}\rangle = H(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle) = \frac{1}{\sqrt{2}}H|0\rangle + \frac{1}{\sqrt{2}}H|1\rangle$$

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

$$= \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|0\rangle - \frac{1}{2}|1\rangle = |0\rangle$$

You can see quantum interference in action – the negative amplitude has reduced the positive one, which resulted in an entirely different state in the end. Superposition, quantum measurement and interference do have an impact on the result, and in quantum algorithms we want to make use of all of them for our computations.

2. Deutsch algorithm

While Deutsch algorithm may not be related to too many real-world problems, it certainly shows some quantum phenomena and carries a great educational value.

In Deutsch problem our input is some function *f(x)* mapping {0,1} → {0,1}. We do not have its formula, we only know it is either constant (always returns 0 or 1) or balanced (returns 0 for half of possible inputs and 1 for the other half.) We want to know whether *f(x)* is constant or not.

From classical perspective we need to call f twice, with argument 0 and 1. If $f(0)\ XOR\ f(1) = 0$, it means that f is constant. Using the quantum model of computation, however, we would need to call *f* only once.

Let us define a quantum gate D_f operating on two qubits, which can be represented by a unitary operator:

$$|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$$

where $\oplus$ is the exclusive OR operation (XOR). If we set $|y\rangle = |0\rangle$, then for the first qubit $|x\rangle = |0\rangle$ the second qubit will be set to $|0 \oplus f(0)\rangle = |f(0)\rangle$. Otherwise the second qubit will be set to $|0 \oplus f(1)\rangle = |f(1)\rangle$.

If we set the first of our qubits to superposition $|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and the second to $|y\rangle = |0\rangle$ then the complete input state of our gate will be $\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle$. The D_f gate will transform it as follows:

$$D_f(\frac{1}{\sqrt{2}}|0\rangle|0\rangle + \frac{1}{\sqrt{2}}|1\rangle|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle|f(0)\rangle + \frac{1}{\sqrt{2}}|1\rangle|f(1)\rangle$$

Notice, that in some way the values of $f(0)$ and $f(1)$ were computed simultaneously. It does not matter, however, if we cannot extract both of them (and as we know, measuring the state once will destroy it, leaving us with only one value.)

We can build a simple quantum circuit that would make it possible to interpret the output of D_f properly. Let us set $|x\rangle$ to $|0\rangle$, and $|y\rangle$ to $|1\rangle$. Now transform both of these using the Hadamard gate. Send such qubits to the D_f gate, and after that transform $|x\rangle$ with the Hadamard gate again. The first two steps may be written as follows:

$$H|0\rangle H|1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = |v_1\rangle$$

$$D_f|v_1\rangle = \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$= (-1)^{f(0)}\left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = |v_2\rangle$$

Second line results from the following:

- For $f(x) = 0$ we have $D_f|y\rangle = D_f\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\oplus f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$
- For $f(x) = 1$ we have $D_f|y\rangle = D_f\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\oplus f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right) = \left(\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right)$

So in general we can write:

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

The equivalence shown above is called the *phase kickback*.

Phase kickback

It can be observed for a controlled quantum gate performing transformation U operating on target qubit in the state being the eigenvector $|v\rangle$ of transformation U. Phase being the eigenvalue of transformation U may be "kicked back" to the control qubit.

Third line results from the fact that $(-1)^{f(0)}(-1)^{f(1)} = (-1)^{f(0)\oplus f(1)}$.

After transformations described above we can make third and last step to solve the Deutsch problem. Let us transform the $|x\rangle$ qubit using the Hadamard gate. Remember that if f is constant, then $f(0) \oplus f(1)$ equals 0, and 1 if otherwise. So if f is constant, then:

$$|v_2\rangle = (-1)^{f(0)}\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

After using the Hadamard gate on the first qubit our system will be in state:

$$|v_{final}\rangle = (-1)^{f(0)}|0\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

If f is balanced then the final state will look like:

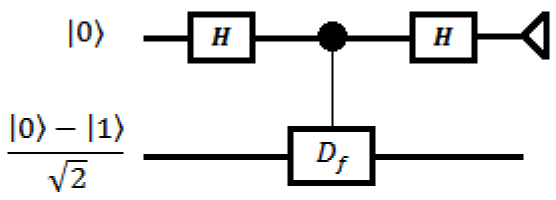
$$|v_2\rangle = (-1)^{f(0)}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

In this case, after transforming the first qubit using the Hadamard gate, the final state will be:

$$|v_{final}\rangle = (-1)^{f(0)}|1\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

To sum the Deutsch algorithm up: if the function f is constant then we are *sure* to read $|0\rangle$ on measurement of the $|x\rangle$ qubit. If f is balanced – we will surely read $|1\rangle$. As you can see, we can solve Deutsch problem by evaluating f only once.

The quantum circuit performing the Deutsch algorithm can be represented as follows (triangle at the end of a qubit path symbolizes measurement):



5 Deutsch algorithm - circuit

3. Deutsch-Jozsa algorithm

While the algorithm presented in previous chapter works faster than the classical version by a constant amount of evaluations of some function *f*, its modification shows a significant speedup. The Deutsch-Jozsa problem (and the quantum algorithm to solve it) is also about determining whether some function *f* is constant or balanced (it returns 0 for half of possible inputs, and 1 for the other half.) However, this time *f* maps $\{0,1\}^n \rightarrow \{0,1\}$, so the domain of *f* consists of all *n*-bit binary numbers.

To decide whether *f* is constant or balanced a classical deterministic algorithm would need to evaluate the function $2^{n-1}+1$ times, in the worst case (if the first half of checked arguments all returned the same value, we need to check at least one argument from the other half.) The quantum Deutsch-Jozsa algorithm makes it possible to solve this problem with only *one* evaluation of *f*.

The quantum circuit designed to solve Deutsch-Jozsa problem is analogical to that solving the Deutsch problem. Its key element is some quantum gate DJ_f. However, in this version it is controlled by an *n-element qubit register* (marked as $|\boldsymbol{x}\rangle$), rather than a single qubit.

$$|\boldsymbol{x}\rangle|y\rangle \rightarrow |\boldsymbol{x}\rangle|y \oplus f(\boldsymbol{x})\rangle$$

An input to our circuit is provided through n control qubits, every one set to $|0\rangle$ (what we can write as $|\boldsymbol{x}\rangle = |0\rangle^{\otimes n}$), and a single target qubit $|y\rangle = |1\rangle$. The first step is to transform all input qubits using Hadamard gates. This leaves the circuit in the following state:

$$|v_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |\boldsymbol{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Note that we are using a mechanism available only for quantum computing – the $|\boldsymbol{x}\rangle$ register holds a superposition of all possible n-bit binary strings.

The second step is about using the gate DJ_f on target qubit $|y\rangle$, controlled by the register $|\boldsymbol{x}\rangle$. After that, the state will be like:

$$|v_2\rangle = \frac{1}{\sqrt{2^n}} DJ_f\left(\sum_{x\in\{0,1\}^n} |\boldsymbol{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |\boldsymbol{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Just like in the Deutsch algorithm, the line above contains the phase kickback.

Third and final step of the Deutsch-Jozsa algorithm is to transform all qubits from the $|\boldsymbol{x}\rangle$ register using Hadamard gates. To make calculation much simpler, let us introduce the following identities:

$$H|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) = \frac{1}{\sqrt{2}}\sum_{z\in\{0,1\}}(-1)^{xz}|z\rangle$$

$$H^{\otimes n}|\boldsymbol{x}\rangle = H|x_1\rangle H|x_2\rangle \dots H|x_n\rangle = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{\boldsymbol{xz}}|\boldsymbol{z}\rangle$$

Using this notation, after the third step our state can be presented as follows:

$$|v_3\rangle = \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{\boldsymbol{xz}}|\boldsymbol{z}\rangle\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
$$= \frac{1}{2^n}\sum_{z\in\{0,1\}^n}\left(\sum_{x\in\{0,1\}^n}(-1)^{f(x)+\boldsymbol{xz}}\right)|\boldsymbol{z}\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

To read the result, we measure the qubit register $|\boldsymbol{z}\rangle$ (which consists of the same qubits that we initially wrote as $|\boldsymbol{x}\rangle$.) Let us notice what the amplitude of the state $|\boldsymbol{z}\rangle = |0\rangle^{\otimes n}$ is:

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}$$

Now, if f is a constant function, then the amplitude can have one of the following values:

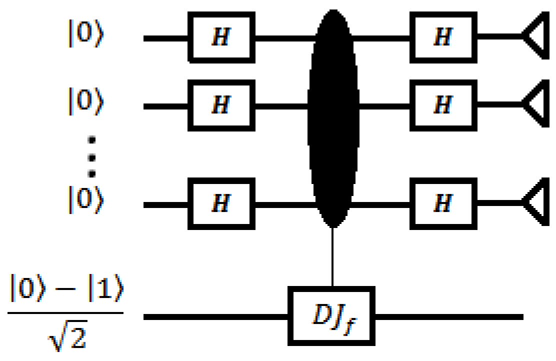
- $\frac{(-1)\times 2^n}{2^n} = (-1)$ for $f(x) = 1$
- $\frac{(-1)\times 2^n}{2^n} = 1$ for $f(x) = 0$

Otherwise, if f is a balanced function, the positive amplitudes will reduce the negative amplitudes (for all values of $x \in \{0,1\}^n$ there will be the same amount of 1s and –1s.) This means that the amplitude of the state $|\boldsymbol{z}\rangle = |0\rangle^{\otimes n}$ will be equal to 0, which implies $|\boldsymbol{z}\rangle \neq |0\rangle^{\otimes n}$ (there is at least a single 1 in the register.)

To sum this up: we have a guarantee that if f is constant, then measuring the register will return all 0s. If f is balanced, at least a single 1 will be measured. As you can see, we were indeed able to determine the character of f using only a single evaluation of it.

The quantum Deutsch-Jozsa algorithm works in $O(1)$ time, while the classical deterministic algorithm needs $O(2^{n-1})$ evaluations. We observe a super-polynomial speedup. (It should be noticed, however, that there exist some probabilistic algorithms capable of solving this problem in polynomial time, with pretty low error rate.)

The circuit performing the quantum Deutsch-Jozsa algorithm can be presented as follows:



6 Deutsch-Jozsa algorithm - circuit

4. Simon's algorithm

Simon's algorithm was constructed to solve the following problem. Given some function f mapping $\{0,1\}^n \rightarrow X$ (where X is some finite set) and knowing that there exists some string $\boldsymbol{s} = s_1 s_2 \dots s_n$ such that $f(x) = f(y)$ only when $x = y$ or $x = y \oplus \boldsymbol{s}$, find the string $\boldsymbol{s}$. We will consider the domain of our function to be a vector space Z_2^n over Z_2. We will also assume that $X \subseteq \{0,1\}^n$ (which is quite expected from the information theory point of view.)

Any classical algorithm (both deterministic and probabilistic) requires at least exponential time in terms of n (which is the length of $\boldsymbol{s}$) to find the solution. Simon's algorithm is capable of solving this problem with an expected number of f evaluations being linear in terms of n, and using $O(n^3)$ elementary quantum gates.

In chapter IV-3 (about the Deutsch-Jozsa algorithm) it has been shown how n Hadamard gates transform n qubits from one of the basis states. For states being an equally weighted superposition of two basic states the n-qubit Hadamard transform may be written as follows:

$$H^{\otimes n}(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|s\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} |\boldsymbol{z}\rangle + \sum_{z \in \{0,1\}^n} (-1)^{sz} |\boldsymbol{z}\rangle$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} (1 + (-1)^{sz}) |\boldsymbol{z}\rangle$$

Let us define a set $S^{\perp} = \{\boldsymbol{z} \in \{0,1\}^n | \boldsymbol{sz} = 0\}$. Note that $S^{\perp}$is a vector subspace of Z_2^n. We should also point out that if $\boldsymbol{sz} = 1$, then $\boldsymbol{z}$ disappears from the equation. Otherwise it stays with amplitude $\frac{1}{\sqrt{2^{n+1}}} \times 2 = \frac{1}{\sqrt{2^{n-1}}}$. Having this definition we can rewrite the earlier formulation of the Hadamard transform:

$$H^{\otimes n}(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|s\rangle) = \frac{1}{\sqrt{2^{n-1}}} \sum_{z \in \{s\}^{\perp}} |\boldsymbol{z}\rangle$$

What is more, if we state that $a, b \in \{0,1\}^n$, $s = a \oplus b$, then we can write the following:

$$H^{\otimes n}(\frac{1}{\sqrt{2}}|a\rangle + \frac{1}{\sqrt{2}}|b\rangle) = H^{\otimes n}(\frac{1}{\sqrt{2}}|a\rangle + \frac{1}{\sqrt{2}}|a \oplus s\rangle) = \frac{1}{\sqrt{2^{n-1}}} \sum_{z\in\{s\}^{\perp}} (-1)^{\boldsymbol{az}} |\boldsymbol{z}\rangle$$

Now let us assume we have some reversible quantum gate $S_f\colon |\boldsymbol{x}\rangle|\boldsymbol{y}\rangle \to |\boldsymbol{x}\rangle|\boldsymbol{y} \oplus f(x)\rangle$. The circuit necessary to perform the quantum subroutine of the Simon's algorithm transforms the input register $|\boldsymbol{x}\rangle$ with Hadamard gates. After that, the S_f gate operates at state $(H^{\otimes n}|\boldsymbol{x}\rangle)|\boldsymbol{0}\rangle$. Finally, the $|\boldsymbol{x}\rangle$ register is transformed again using Hadamard gates, and gets measured afterwards.

The Simon's algorithm may be written as follows[3]:

```
1. Counter i = 1
2. Prepare state $\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|\boldsymbol{0}\rangle$ (transform $|\boldsymbol{x}\rangle = |0\rangle^{\otimes n}$ using $H^{\otimes n}$).
3. Perform $S_f$ - state will be transformed to $\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle|f(x)\rangle$.
4. Transform the control register with $H^{\otimes n}$.
5. Measure the control register and save the result $w_i$.
6. If the dimension of linear subspace spanned by $\{w_i\}$ is equal to
   $n$ - 1 then go to step 7. Else increment i, and go to step 2.
7. Solve $Ws^T = \boldsymbol{0}^T$.
8. Return the non-zero solution
```

To analyze the algorithm presented about let us notice that $\{0,1\}^n$ can be divided into 2^{n-1} pairs of words in form of $\{x, x \oplus s\}$ (for every word from 2^n possible words we determine its "sibling" by performing *XOR* with the string $\boldsymbol{s}$ – there will be twice less pairs than all the words.) Define the set I as the subset of $\{0,1\}^n$ containing a single representative from every pair $\{x, x \oplus s\}$. Then, the state of a system after using the S_f gate can be written as [12]:

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle = \frac{1}{\sqrt{2^{n-1}}} \sum_{x\in I} \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)|f(x)\rangle$$

The above is true because the fact that with equal probability we can measure any of 2^n possible values is equivalent to the fact that we can measure one of two values from one of the 2^{n-1} pairs with equal probability.

We already stated before that:

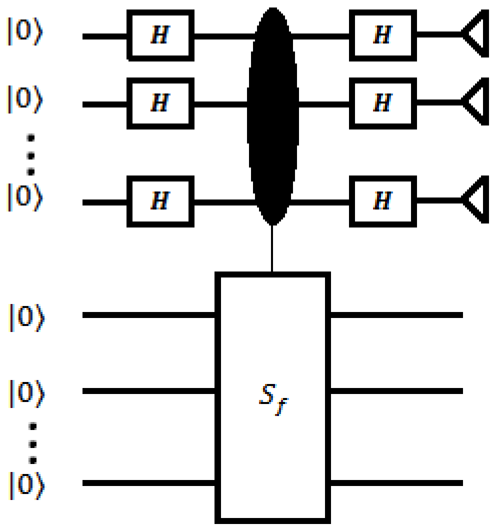
$$H^{\otimes n}(\frac{1}{\sqrt{2}}|x\rangle + \frac{1}{\sqrt{2}}|x \oplus s\rangle) = \frac{1}{\sqrt{2^{n-1}}} \sum_{z\in\{s\}^{\perp}} (-1)^{\boldsymbol{xz}} |\boldsymbol{z}\rangle$$

The transformation above leaves the control register in superposition of all states from $s^{\perp}$ with equal amplitudes. This means that the values $\boldsymbol{s_i}$ measured in step 5 of the quantum procedure are indeed some random vectors from $s^{\perp}$. We need n – 1 linearly independent vectors from $s^{\perp}$ to be able to determine $\boldsymbol{s}$, and this is why we check the dimension of the vector subspace spanned by measured vectors.

After the transformation performed above we know that the control register contains the superposition of all vectors $\boldsymbol{z}$ such that $\boldsymbol{zs} = 0$. Having n – 1 different vectors $\boldsymbol{z}$ (so every one differs by at least a single bit from the others) we can determine all bits of the ***s*** vector, and this is what we do in steps 7 and 8. We are not interested in the trivial solution of **0**.

Solving a set of $n - 1$ linear equations can be done in time linear in terms of n (for ex. using Gauss elimination.) To find n linearly independent vectors ***z*** we need to evaluate f about $n - 1$ times (assuming we are going to measure a different value every time, which is quite expected if n is large.).

The quantum circuit performing Simon's algorithm quantum subroutine may be presented as follows:



7 Simon's algorithm – quantum subroutine – circuit

5. Quantum phase estimation

Many quantum algorithms, and especially those based on the quantum Fourier transform, rely on the possibility of determining or estimating the phase of a qubit. This is related to the fact that quite often in quantum computing important information get *encoded in the relative phase* of the quantum state.

The basic coder and decoder of information written in the relative phase is the Hadamard gate. Let us look at it from a different perspective, for some basic state $|x\rangle, x \in \{0,1\}$:

$$H|x\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle$$

One can notice that the Hadamard gate is somehow *encoding information* about $|x\rangle$ in the relative phase between the basic states $|0\rangle$ and $|1\rangle$. Because the Hadamard gate is its own reverse, we can use it to decode the information from the relative phase as well:

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

What is more, we can generalize above statements to n-qubit quantum states. As it was mentioned before, the n-qubit Hadamard gate transforms an n-qubit register $|\boldsymbol{x}\rangle$ as follows:

$$H^{\otimes n}|\boldsymbol{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{y}\in\{0,1\}^n} (-1)^{\boldsymbol{xy}}\,|\boldsymbol{y}\rangle$$

This transformation may be considered as encoding the information about the $|\boldsymbol{x}\rangle$ register in the relative phases $(-1)^{\boldsymbol{xy}}$of basic states $|\boldsymbol{y}\rangle$. Of course, transforming this state through the same gate again will decode this information:

$$H^{\otimes n}\left(\frac{1}{\sqrt{2^n}} \sum_{\boldsymbol{y}\in\{0,1\}^n} (-1)^{\boldsymbol{xy}}|\boldsymbol{y}\rangle\right) = H^{\otimes n}(H^{\otimes n}|\boldsymbol{x}\rangle) = H^{\otimes n}H^{\otimes n}|\boldsymbol{x}\rangle = I|\boldsymbol{x}\rangle = |\boldsymbol{x}\rangle$$

Phases in the form of $(-1)^{xy}$ are of course special cases. We need a more precise description to work with general cases:

Relative phase of a quantum state

Relative phase of a quantum state is a complex number $e^{2\pi i\omega}$ for any real number $\omega \in (0, 1)$.

It can be noticed that Hadamard gate on its own is not capable of encoding more complex information stored in the relative phase. However, there exists a mechanism of estimating the quantum phase, being the extension of the standard Hadamard coder/decoder.

Let us assume that for some reason we would like to know the relative phase ω of some quantum state written as follows:

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{2\pi i\omega y}|y\rangle$$

where $|y\rangle$ is an integer belonging to $[0, 2^n)$. Notice that ω can be written in binary form:

$$\omega = 0.x_1x_2x_3 \ldots$$

The line above makes it relatively easy to note:

$$2^k\omega = x_1x_2 \ldots x_k.x_{k+1}x_{k+2} \ldots$$

It is also known that $e^{2\pi ik} = 1$ for any integer k. This makes it possible to write:

$$e^{2\pi i(2^k\omega)} = e^{2\pi i(x_1x_2\ldots x_k)}e^{2\pi i(0.x_{k+1}x_{k+2}\ldots)} = e^{2\pi i(0.x_{k+1}x_{k+2}\ldots)}$$

For a single-qubit quantum state with some relative phase $\omega = 0.x_1$ we can state:

$$\frac{1}{\sqrt{2}}\sum_{y=0}^{1} e^{2\pi i(0.x_1)y}|y\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{1} e^{2\pi i\left(\frac{x_1}{2}\right)y}|y\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{1} e^{\pi i(x_1y)}|y\rangle = \frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{x_1y}|y\rangle$$

$$= \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1}|1\rangle)$$

As we have shown earlier, x_1 can be decoded using the Hadamard gate:

$$H(\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{x_1}|1\rangle)) = |x\rangle$$

Because $\omega = 0.x_1$, by reading x_1 we got ω. This is still the basic case, however. What do we do if the phase is more complicated?

Let us introduce a very useful identity:

$$\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{2\pi i\omega y}|y\rangle =$$

$$= \left(\frac{|0\rangle + e^{2\pi i(2^{n-1}\omega)}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + e^{2\pi i(2^{n-2}\omega)}|1\rangle}{\sqrt{2}}\right) \otimes \dots \otimes \left(\frac{|0\rangle + e^{2\pi i(\omega)}|1\rangle}{\sqrt{2}}\right) =$$

$$= \left(\frac{|0\rangle + e^{2\pi i(0.x_n x_{n+1}\dots)}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + e^{2\pi i(0.x_{n-1} x_n\dots)}|1\rangle}{\sqrt{2}}\right) \otimes \dots \otimes \left(\frac{|0\rangle + e^{2\pi i(0.x_1 x_2\dots)}|1\rangle}{\sqrt{2}}\right)$$

Now let us take some two-qubit state $\frac{1}{\sqrt{2^2}}\sum_{y=0}^{2^2-1} e^{2\pi i\omega y}|y\rangle$. This time we will assume that $\omega = 0.x_1x_2$. Using the identity defined above we can write:

$$\frac{1}{\sqrt{2^2}}\sum_{y=0}^{2^2-1} e^{2\pi i(0.x_1x_2)y}|y\rangle = \left(\frac{|0\rangle + e^{2\pi i(0.x_2)}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + e^{2\pi i(0.x_1x_2)}|1\rangle}{\sqrt{2}}\right)$$

As you see, x_2 can be read immediately, by transforming the first qubit using the Hadamard gate and measuring it. If the measured value is 0, then to read x_1 we only have to transform the second qubit using the Hadamard gate and measure it as well. Otherwise, the state of the second qubit can be presented as $\frac{|0\rangle + e^{2\pi i(0.x_1 1)}|1\rangle}{\sqrt{2}}$. To get x_1 we need to transform this state to a form that allows us to do so.

Let us define a single-qubit phase rotation gate:

Single-qubit phase rotation gate

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

where k means position 2^{-k} in phase ω.

In our case, we are interested in "freeing" x_1 from x_2, which is at the 2nd position after the decimal point. For this purpose we can make use of the R_2 gate, or more accurately – its inverse:

$$R_2 = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i(0.01)} \end{bmatrix}$$

$$R_2^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-2\pi i(0.01)} \end{bmatrix}$$

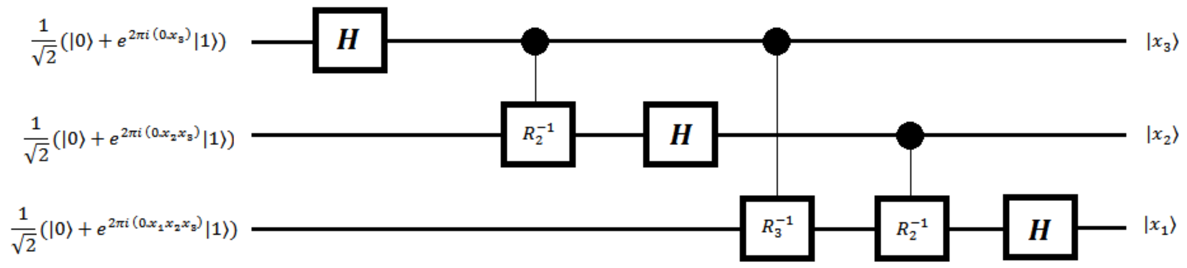
We will now see the effect of transforming the second qubit from our example with R_2^{-1}:

$$R_2^{-1}\left(\frac{|0\rangle + e^{2\pi i(0.x_1 1)}|1\rangle}{\sqrt{2}}\right) = \frac{|0\rangle + e^{2\pi i(0.x_1 1 - 0.01)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{2\pi i(0.x_1)}|1\rangle}{\sqrt{2}}$$

We now have the state from which we can extract x_1 using the Hadamard gate. Knowing both $x_1$and x_2 we know $\omega = 0.x_1x_2$. We have read the relative phase of a two-qubit state.

Notice that phase rotation is only required when the first qubit ($|x_2\rangle$) returns 1 when measured. Our circuit could use a controlled R_2^{-1} gate then, where the first qubit would be controlling the second one. After that, both qubits would be encoded using the Hadamard gate and their values could be measured.

The solution proposed above is easily scalable. For example, let us add another qubit to our system. Its state would be $\frac{|0\rangle + e^{2\pi i(0.x_1x_2x_3)}|1\rangle}{\sqrt{2}}$, which can be transformed using R_3^{-1} controlled by the first qubit, and R_2^{-1} controlled by the second one. The original two qubits should be treated analogically to the earlier procedure. Finally all qubits could be transformed by Hadamard gates and measured.


$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_3)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_2x_3)}|1\rangle)$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.x_1x_2x_3)}|1\rangle)$
H
R_2^{-1}
H
R_3^{-1}
R_2^{-1}
H
$|x_3\rangle$
$|x_2\rangle$
$|x_1\rangle$

8 Quantum phase estimation - circuit

Note that this algorithm determines phase that can be written like $0.x_1x_2 \dots$. It can be proven that for phases ω that are impossible to write in this form, the algorithm presented above returns an approximation x_{approx}, which fulfills:

$$\left|\frac{x_{approx}}{2^n} - \omega\right| \leq \frac{1}{2^n}$$

with probability at least $\frac{8}{\pi^2}$.

6. Quantum Fourier transform

Circuit performing the algorithm presented in previous chapter allows us to determine the relative phase $\omega = 0.x_1x_2 \dots x_n$ of a quantum state $\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{2\pi i\omega y}|y\rangle$. In other words, it requires ω to be possible to write as $\frac{x}{2^n}$ for some integer x (for ω impossible to write like this the algorithm returns the closest possible approximation with high probability.) This means that the presented circuit performs the following transformation:

$$\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{2\pi i(\frac{x}{2^n})y}|y\rangle \rightarrow |x\rangle$$

Of course, output qubits are in reversed order (the first qubit stores the output value of the last one etc.,) but this is only a matter or indexes.

Let us write the general form of discrete Fourier transform [10, 11]:

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{-2\pi i \frac{k}{N} n} x_n$$

Now, consider the transformation performed by the reverse of our phase estimation circuit (we can do it because quantum circuit are reversible by definition):

Quantum Fourier transform – QFT

$$QFT_N \colon |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \frac{x}{N} y} |y\rangle$$

Similarity is not accidental. The second form is called Quantum Fourier transform acting on one of N basic states (for the form above $N = 2^n$).

We know from chapter IV-5 how to build a circuit performing phase estimation, so we are able to build its reverse by changing gates to their reverses and propagating the circuit backwards. This allows us to calculate QFT easily for any quantum state being the superposition of basic states.

QFT can also be presented as a unitary matrix:

$$\frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \dots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}, \omega = e^{\frac{2\pi i}{N}}$$

It should be mentioned that because QFT is unitary, we can easily calculate its reverse (using the conjugate operator.)

Inverse quantum Fourier transform – QFT^{-1}

$$QFT_N^{-1} \colon |x\rangle \to \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{-2\pi i \frac{x}{N} y} |y\rangle$$

7. Determining period of periodic states

The problem presented below may appear not to be directly related to reality. However, it is an important step on the way to understanding one of the most known quantum algorithms – Shor's integer factorization algorithm. What is more, it shows quantum Fourier transform in action.

A periodic state is a quantum state in form:

$$|\phi_{r,b}\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|xr+b\rangle$$

where r is the period, N – number of repetitions of the period, b – offset chosen randomly from $\{0,1,\dots,r-1\}$. The period finding problem is about finding r being given Nr and a machine generating periodic quantum states of some unknown parameters.

As we do not know any of the parameters, normal measurement will only give us some random state from $\{0,1,\dots,r-1\}$, which gives us no information about r. However, we have the inverse quantum Fourier transform coming to help us! It can be proven that:

$$QFT_{Nr}^{-1}|\phi_{r,b}\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}e^{-2\pi i\frac{b}{r}k}|Nk\rangle$$

It means that transforming the periodic state generated by our machine with inverse QFT will give us a state that we can measure to obtain some number Nk for some random $k \in \{0,1,\dots,r-1\}$. This allows us to compute $\frac{Nk}{Nr} = \frac{k}{r}$ and write it in lowest terms in order to get r.

One of the efficient ways of bringing $\frac{k}{r}$ to lowest terms is to use the extended Euclidean algorithm. It enables effective calculation of the greatest common divisor of k and r (and $\frac{k}{r}$ expressed in lowest terms is $\frac{\frac{k}{GCD(k,r)}}{\frac{r}{GCD(k,r)}}$.) Moreover, the following theorem can be proven:

> Let r be some positive integer, and k_1 and k_2 be two independent numbers selected randomly from $\{0,1,\dots,r-1\}$. Let c_1, c_2, r_1, r_2 be some positive integers that fulfill $GCD(r_1,c_1) = GCD(r_2,c_2) = 1$ and $\frac{k_1}{r} = \frac{c_1}{r_1}$ and $\frac{k_2}{r} = \frac{c_2}{r_2}$.
>
> Then with probability $\frac{6}{\pi^2}$ we have $r = LCM(r_1,r_2)$.

In other words, measuring $\frac{k}{r}$ twice and using the extended Euclidean algorithm makes it possible to effectively compute r with high probability of success.

There exists some risk that k and r share a common divisor greater than 1. This means that when bringing our fraction to lowest terms we will divide both k and i r by that divisor, and so we will read the wrong value of r. However, probability of two natural numbers being relatively prime [13] is $\frac{6}{\pi^2} \approx 61\%$, so the expected amount of calling our procedure to read the correct r is $\frac{2r}{5}$. It should be recognized that there are some simple classical methods allowing to check if we found the correct period.

There is also some more general version of the described problem, in which we do not know Nr and periodic states generated by our machine are in form $|\phi_{r,b}\rangle = \frac{1}{\sqrt{m_b}}\sum_{x:0\le xr+b\le 2^n}|xr+b\rangle$, where n is given and $m_b \approx \frac{2^n}{r}$ guarantees the state norm to be 1. For educational reasons we will write two theorems without proof. Let x be the result of measuring $QFT_{2^n}^{-1}|\phi_{r,b}\rangle$. It can be proven that for any x fulfilling $\left|\frac{x}{2^n} - \frac{k}{r}\right| \le \frac{1}{2m_b r}$ for some k, probability of measuring x is at least $\frac{m_b}{\pi^2 2^{n-2}}$. It can be also proven that if measurement will return one of two closest approximations of $\frac{k}{r}$ and $2^n \geq 2r^2$ then it is possible to determine $\frac{k}{r}$.

8. Eigenvalue estimation

Just like with quantum state phase estimation, being able to estimate the eigenvalue of some transformation plays a crucial role in many algorithms.

The point of eigenvalue estimation is to give the best possible approximation of ω having an operator U, its eigenvector $|v\rangle$ and knowing that eigenvalue is in the form $e^{2\pi i\omega}$.

As we have shown in previous chapters, considering controlled quantum gates targeting their eigenvectors, we can observe the phase kickback (phase is kicked back to the control register.) We have interpreted this action like encoding the information about the state in the relative phase. In this chapter we consider yet another perspective.

Consider an unitary operator U, with eigenvector $|v\rangle$ and eigenvalue $e^{2\pi i\omega}$, for which we know an effective quantum gate. The controlled version of U can be represented in the following way:

$$controlled - U|1\rangle|v\rangle = |1\rangle U|v\rangle = |1\rangle e^{2\pi i\omega}|v\rangle = \ e^{2\pi i\omega}|1\rangle|v\rangle$$

$$controlled - U|0\rangle|v\rangle = |0\rangle|v\rangle$$

If this gate was applied to a superposition $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ it would effect in the following state:

$$controlled - U(\frac{1}{\sqrt{2}}|0\rangle \ + \frac{1}{\sqrt{2}}|1\rangle)|v\rangle = (\frac{1}{\sqrt{2}}|0\rangle + \frac{e^{2\pi i\omega}}{\sqrt{2}}|1\rangle)|v\rangle$$

This means that the information about the eigenvalue was encoded in the relative phase of the control qubit. Let us also notice that for transformation U^n, where n is a positive integer, the eigenvector remains the unchanged ($|v\rangle$), and the eigenvalue is in form $\left(e^{2\pi i\omega}\right)^n = e^{2\pi i n\omega}$. This means we have the following:

$$controlled - U^n(\frac{1}{\sqrt{2}}|0\rangle \ + \frac{1}{\sqrt{2}}|1\rangle)|v\rangle = (\frac{1}{\sqrt{2}}|0\rangle + \frac{e^{2\pi i n\omega}}{\sqrt{2}}|1\rangle)|v\rangle$$

The tensor product of states resulting from the above transformation for $n = 2^x$, where $x \in \{0,1,\dots,X-1\}$ is:

$$\left(\frac{|0\rangle + e^{2\pi i(2^{X-1}\omega)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + e^{2\pi i(2^{X-2}\omega)}|1\rangle}{\sqrt{2}}\right)\cdots\left(\frac{|0\rangle + e^{2\pi i\omega}|1\rangle}{\sqrt{2}}\right) = \frac{1}{\sqrt{2^X}}\sum_{y=0}^{2^X-1} e^{2\pi i\omega y}|y\rangle$$

As we know from chapters IV-5 and IV-6 (about phase estimation and the quantum Fourier transform) transforming the above state with QFT^{-1} will result in a state, which can be treated as a binary integer after measurement. If we mark the measured value as x then our approximation is $\omega_{approx} = \frac{x}{2^X}$. Paradoxically, the content of $|v\rangle$ is of no interest for us (from presented algorithm's perspective it can be dismissed after transformations.

Instead of using X controlled U^{2^x} gates to create the desired state we could build a single controlled U^X gate operating on X qubits (transforming nth qubit with $U^{2^{X-n-1}}$ for $n \in \{0, 1, \dots X-1\}$). X Hadamard gates necessary to prepare the desired input state (X qubits set to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ each) can be replaced with QFT (it is relatively easy to show that $QFT|0\rangle^{\otimes X} = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)^{\otimes X}$).

Eigenvalue estimation procedure can be presented as follows:

1. Set X-qubit control register to $|0\rangle^{\otimes X}$ and prepare the eigenvector $|v\rangle$ in target register
2. Transform the control register with QFT
3. Perform $controlled - U^X$
4. Transform the control register with QFT^{-1}
5. Measure the control register
6. Return the measured value divided by 2^X

9. Factorization

The year of 1994 brought some kind of a revolution. After Peter Shor has presented a quantum algorithm capable of efficiently computing prime factors for integers, the IT world has suddenly become very interested in quantum computing. It has become clear that classical cryptography, based on factorization being difficult, is very easy to break with a sufficiently large quantum computer.

Factorizing an integer N is about finding some positive integers $p_1, p_2, \ldots, p_n$ and $r_1, r_2, \ldots, r_n$ such that numbersp_i are distinct prime numbers and $N = p_1^{r_1} p_2^{r_2} \ldots p_n^{r_n}$. In addition let us assume that we are interested only in odd numbers (because for even numbers one of the factors is surely 2) and such that are not prime number powers (it can be shown that it is easy to find prime factors otherwise.) Note that every number N fulfilling above limitations has at least two different prime factors.

Factorization can be reduced to the following problem (called the *Order finding problem*): having two positive relatively prime integers N and a, find the order of $a\ mod\ N$ (which is a positive integer r, such that $a^r\ mod\ N = 1$). Note that for a proper input there always exists a solution for this problem. If $GCD(N, a) = 1$ then the number 1 will occur at some place in the series of $a\ mod\ N, a^2\ mod\ N, a^3\ mod\ N \ldots$ (after which the values will repeat). We will describe how finding orders help us factorize integers later.

Let us introduce some quantum operator U_a:

$$U_a\colon |s\rangle \rightarrow |sa \bmod N\rangle, s < N$$

$$|s\rangle \rightarrow |s\rangle, s \geq N$$

We know that there exists an inverse of a modulo N (which is implicated by a and N being relatively prime) a' such that $a'a \equiv aa' \equiv 1 \pmod N$. This implies that U_a is an unitary operator. Based on this and the definition of order finding problem we can state:

$$U_a^r\colon |s\rangle \rightarrow |sa^r \bmod N\rangle = |s\rangle$$

Now let us consider some quantum state:

$$|u_k\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{-2\pi i \frac{k}{r} s} |a^s \bmod N\rangle$$

Transforming it with U_a will result in the following:

$$U_a|u_k\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{-2\pi i \frac{k}{r} s} U_a |a^s \bmod N\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{-2\pi i \frac{k}{r} s}\, |a^{s+1} \bmod N\rangle =$$

$$= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{-2\pi i \frac{k}{r}(s+1)} e^{2\pi i \frac{k}{r}} |a^{s+1} \bmod N\rangle = e^{2\pi i \frac{k}{r}} \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{-2\pi i \frac{k}{r}(s+1)} |\, a^{s+1} \bmod N\rangle =$$

$$= e^{2\pi i \frac{k}{r}} |u_k\rangle$$

Last identity is true because $e^{-2\pi i\frac{k}{r}r}|a^r \bmod N\rangle = e^{-2\pi i\frac{k}{r}0}|a^0 \bmod N\rangle$ (remembering that $e^{-2\pi ik} = 1$ for any integer k). This implies that $|u_k\rangle$ is the eigenvector of the operator U_a with eigenvalue $e^{2\pi i\frac{k}{r}}$.

Having the state representing the eigenvector of U_a we would be able to determine $\frac{k}{r}$, using the eigenvalue estimation algorithm. As we already know, this would allow us to get the value of r with high probability. This would make us solve the order finding problem. However, we do not know r, so we cannot prepare the state $|u_k\rangle$.

Let us look at the problem from a bit broader perspective. We may notice that we do not need to read some specific eigenvalue $e^{2\pi i\frac{k}{r}}$ of some specific vector $|u_k\rangle$. Any from the eigenvalues of U_a would be helpful, and we can get *some* eigenvalue for any eigenvector $|u_k\rangle$ for some k. In other words, we can use an equally weighted superposition of all eigenvectors of the U_a operator and get the result for one randomly measured.

We will now show how we can prepare such superposition, even without knowing r. We want to get the following state:

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|u_k\rangle = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}e^{-2\pi i\frac{k}{r}s}|a^s \bmod N\rangle$$

We know that $|a^s \bmod N\rangle = |1\rangle$ for $s \equiv 0 \pmod r$. Because $s \in \{0, 1, \dots, r-1\}$ then the only value of s congruent to 0 (mod r) is 0. This is why the integer amplitude of state $|1\rangle$ is the sum of all amplitudes of states for which $s = 0$:

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\frac{1}{\sqrt{r}}e^{-2\pi i\frac{k}{r}0} = \frac{1}{r}\sum_{k=0}^{r-1}1 = \frac{r}{r} = 1$$

Because total amplitude of $|1\rangle$ is equal to 1, we cannot measure any other state (all other amplitudes must be 0, because the sum of squares of all amplitudes must be equal to 1.) This implies the following:

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|u_k\rangle = |1\rangle$$

Willing to use an equally weighted superposition of all eigenvectors of U_a it is sufficient to use the state $|1\rangle$. This means we can create the following state:

$$|0\rangle|1\rangle = |0\rangle\left(\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|u_k\rangle\right) = \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|0\rangle|u_k\rangle$$

Now we can use the eigenvalue estimation algorithm to read $\frac{k}{r}$ for some randomly selected $k \in \{0, 1, \dots, r-1\}$:

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}|0\rangle|u_k\rangle \rightarrow |x\rangle|u_k\rangle$$

where $\frac{x}{2^n}$ is an approximation of $\frac{k}{r}$, which will give us a chance to determine r with high probability, using the continued fractions algorithm.

***Continued Fractions Algorithm* - CFA**

For every rational number $\frac{x}{2^n}$ there exists a series (of length linearly limited by n) of approximations $\frac{a_1}{b_1}, \frac{a_2}{b_2}, \dots, \frac{a_m}{b_m}$, where:

- $\frac{a_m}{b_m} = \frac{x}{2^n}$
- $a_1 < a_2 < \cdots < a_m$
- $b_1 < b_2 < \cdots < b_m$

Moreover, if there exists some fraction $\frac{k}{r}$ such that $\left|\frac{x}{2^n} - \frac{k}{r}\right| \le \frac{1}{2r^2}$, then $\frac{k}{r}$ is in the series of approximations of $\frac{x}{2^n}$.

List of all elements of the series can be computed in polynomial time in terms of n.

The quantum Order Finding Algorithm can be presented as follows:

```
1.  Select an integer $n$ fulfilling $2^n \ge 2r^2$ (for ex. $\lceil 2logN \rceil$)
2.  Set an $n$-qubit control register to $0^{\otimes n}$.
3.  Set an $n$-qubit target register to $|000\dots1\rangle = |1\rangle$.
4.  Transform the control register with $QFT$
5.  Perform $c - U_a^x$
6.  Transform the control register with $QFT^{-1}$
7.  Measure the control register to obtain the approximation of $\frac{x_1}{2^n}$
8.  Use CFA to get $c_1, r_1$ fulfilling $\left|\frac{x_1}{2^n} - \frac{c_1}{r_1}\right| \le \frac{1}{2^{\frac{n-1}{2}}}$. If not found then
    return **FAIL**.
9.  Repeat steps 1 - 8 to obtain approximation of $\frac{x_2}{2^n}$ and another
    pair $c_2, r_2$ fulfilling $\left|\frac{x_2}{2^n} - \frac{c_2}{r_2}\right| \le \frac{1}{2^{\frac{n-1}{2}}}$
10. Compute $r = LCM(r_1, r_2)$
11. If $a^r \bmod N = 1$ return r. Else return **FAIL**.
```

It can be proven that the order finding algorithm will return the correct result r being the order of a mod N with probability at least $\frac{384}{\pi^6}$. Otherwise it will return some multiple of r or the value **FAIL**.

One potentially problematic element in terms of complexity is U_a^x. For $x = 2^j$ performing $c - U_a^{2^j}$ is about performing $c - U_a$ 2^j times. However, one may prove that

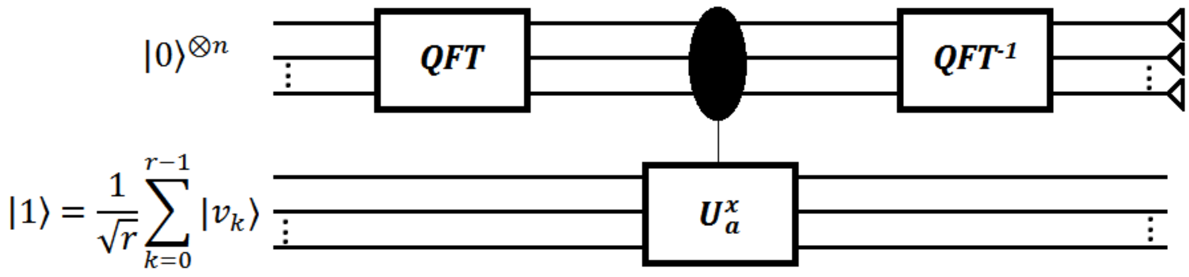
$c - U_a^{2^j} = c - U_{a^{2^j}}$ (multiplying by a mod N 2^j times is equivalent to multiplying by a^{2^j} mod N). Because there are some effective classical methods of computing a^{2^j} mod N we can use them to precompute some values and therefore provide an exponential speedup from multiplying by a mod N 2^j times.

Factorization of an integer N algorithm can be presented as follows:

```
1. Run the quantum order finding algorithm for input $N$ to
   determine the order $r$.
2. If we got an even order:
      a. If for $b = (a^{\frac{r}{2}} \bmod N)$ the $GCD(N, b-1)$ is a non-trivial
         divisor of $N$ return $GCD(N, b-1)$
3. Repeat from step 1 using another $a$.
```

Notice that for an even order r we have $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$. In addition we know that because $a^r \equiv 1 \pmod N$ then $a^r - 1 \equiv 0 \pmod N$. This means that $a^r - 1$ is divisible by N (because r is even we know that $\frac{r}{2}$ is a positive integer, just like $a^{\frac{r}{2}} - 1$ and $a^{\frac{r}{2}} + 1$.) Knowing $b = a^{\frac{r}{2}}$ mod N we hope that $GCD(N, b-1)$ is going to be a non-trivial divisor of N. If not – we try again for another a.

Circuit performing the quantum subroutine of the integer factorization algorithm may be presented as follows:



9 Integer factorization algorithm – quantum subroutine - circuit

The quantum circuit used in this algorithm requires only $O\big((\log N)^2 \log\log(N) \log\log\log(N)\big)$ elementary quantum gates. The best known classical deterministic algorithms require using $e^{O((\log N)^{\frac{1}{2}}(\log\log N)^{\frac{1}{2}})}$ gates, and the best heuristics require $e^{O((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})}$ gates. We observe a super-polynomial speedup for integer factorization problem.

It should be mentioned that the original algorithm presented by Peter Shor was a bit different from the one presented above. Algorithm used in this chapter is based on the possibility of determining the eigenvector for a given transformation (here: U_a). This procedure is equivalent to the original one, which was related to determining the period of a periodic state (described in chapter IV-7.) The difference lays in different state space basis. Peter Shor used the standard computation basis, while the algorithm presented above is relies on the basis spanned by the eigenvectors of the U_a operator.

Bibliography:

1. Kaye P., Laflamme R., Mosca M., *Introduction to Quantum Computing*, 1st edition, Oxford University Press 2007
2. Nielsen M. A., Chuang I. L., *Quantum Computation and Quantum Information*, 10th Anniversary Edition, Cambridge University Press 2011
3. Krzysztof Giaro, Marcin Kamiński, *Wprowadzenie do algorytmów kwantowych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2003
4. http://www.ibm.com/developerworks/library/l-quant/index.html (read 20.10.2014)
5. http://www.technologyreview.com/featuredstory/531606/microsofts-quantum-mechanics/ (read 28.11.2014)
6. http://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware (read 29.11.2014)
7. http://qudev.ethz.ch/content/courses/QSIT07/qsit05_v1_2page.pdf (read 29.11.2014)
8. http://www.stat.physik.uni-potsdam.de/~pikovsky/teaching/stud_seminar/Bell_EPR-1.pdf (read 2.12.2014)
9. http://qudev.phys.ethz.ch/content/courses/QSIT09/QSIT09_V04_slides.pdf (read 17.12.2014)
10. http://www.eecs.berkeley.edu/~luca/quantum/lecture06.pdf (read 3.01.2015)
11. http://www.dsp.agh.edu.pl/_media/pl:dydaktyka:fft.pdf (read 3.01.2015)
12. http://www.eecs.berkeley.edu/~luca/quantum/lecture07.pdf (read 3.01.2015)
13. https://primes.utm.edu/notes/relprime.html (read 4.01.2015)

Project website:

- https://github.com/3yakuya/Quantum-Shell