"POLITEHNICA" University of Bucharest
Faculty of Automation and Computer Science

New techniques for computing and information processing

Quantum Computing

Doctorand: **Lucian Dragne**
Adviser: **Prof. dr. ing. Florica Moldoveanu**,
Computer Science Department, Faculty of Automation and Computer Science, UPB

Summary

Chapter 1. describes the base theoretical concepts that constitute the foundation of the quantum computing paradigm, including the main assumptions that guide the developments in this field of research. This chapter also presents the Church-Turing conjectures that have been driving the algorithm theory from its conception, the how they are now disputed under the new quantum computing paradigm.
Chapter 2. presents in extended detail the main problem addressed in quantum information processing and the very reason this approach took off: maximizing the efficiency of computing processes. The author conceived a few demonstrations for some known examples that prove the power of quantum computing and quantum information processing, explaining how they are deeply linked to physical processes. The author analyzed Deutsch-Jozsa problem and its generalization to Boolean functions extended to n – dimensional finite spaces, and conceived a classical probabilistic algorithm for solving this problem. The performances of quantum algorithm and the classical probabilistic ones were analyzed and compared, proving the increased time efficiency of the quantum algorithm.
In chapter 3. a set of theorems together with their formal demonstrations, designed by the author, were presented in order to justify the graphical representations of qubits using three dimensional unit spheres and to provide for the quantum operations modeling through three dimensional rotations, only around the coordinate axes. Therefore it was proved there is an interesting conceptual relationship between the spherical rotations on one hand, and the one qubit unitary operations on the other.
Chapters 4. and 5. present in detail the theory of quantum circuits together with a few quantum circuits designed by the author:

- a circuit that offers a minimal implementation of the generic controlled operator on two qubits, this implementation using only one qubit gates and CNOT gates
- a circuit that implements the Fredkin gate using only one Toffoli gate and two CNOT gates
- a circuit that implements the Fredkin gate using only six gates, each on two qubits
- a circuit that implements the generalized Toffoli gate, without using working qubits. This circuit has a polynomial complexity of the second grade
- a circuit that implements a generic controlled operator on an arbitrary number of qubits, without using working qubits. This circuit too has a polynomial complexity of the second grade

Chapter 6. presents a rigorous analysis on the universality of quantum gates. The author analyzes the exact universality of infinite sets of gates on one qubit, and how complicated circuits can be decomposed into simpler ones. This chapter also includes the complexity evaluations, as conceived by the author. Then, the author provides a proof based on construction for an approximate universality, with arbitrary small error, using a discrete set of quantum gates: Hadamard, phase, CNOT and Toffoli.
In the last three chapters, a few known algorithms are analyzed, under some new light; these algorithms are a proof for the increased efficiency of the quantum computers, with respect to the temporal complexity, as compared to the corresponding known algorithms from classical computing.
Quantum Fourier Transformation: considering a natural number $N = 2^n$, the quantum Fourier transformation is defined as:

$$|j\rangle = |j_1, \dots, j_n\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

As demonstrated in Chapter 7. it is possible to implement this transform with a quantum circuit containing $\Theta(n^2)$ gates.

Phase estimation: consider the unitary operator U and one of its singular states $|u\rangle$ with the corresponding singular value $e^{2\pi i \varphi}$. As it is described in Chapter 8., it is possible to design an algorithm that efficiently computes the $|\tilde{\varphi}\rangle|u\rangle$ state, where $\tilde{\varphi}$ is an approximation of φ on $t - \left\lceil \log_2(2 + \frac{1}{2\epsilon}) \right\rceil$ bits. The algorithm returns the correct result with a probability of at least $1 - \epsilon$.

Order determination: The order of an integer number x modulo N is the smallest positive integer r such that $x^r \bmod N = 1$. Chapter 9. shows how this number can be computed on a quantum computer using $O(L^3)$ operations by using the quantum phase estimation algorithm applied to the integer numbers on L bits: x and N.

Factorization: The prime factors of a positive integer on L bits, N, can be determined using $O(L^3)$ operations by reducing this problem to the previous one: computing the order of a purely random number x, co-prime with N.

In the last chapter, the author provides an analysis of the current languages for quantum computing, both at low and high level. The author also proposes a new language, built around .Net framework architecture, which extends from the C# language and uses both imperative and functional programming paradigms. The architecture for the compiler of this new language is described in both local environments and on an architecture based on cloud computing.

1. Basic concepts

Quantum computing represents the study of the information processing tasks that can be realized using systems behaving according to the quantum mechanics laws, as they are currently formulated in the mainstream science. The unit of quantum information is called quantum bit, or shortly, qubit [37].
As the standard, classical (as opposed to quantum) hardware components become smaller and smaller, at a fast pace, according to the Moore's law, which is more or less surprisingly still relevant, it is envisaged that not very far away in the future the quantum dimensions [11] will come into play. And when that happens, the classical methods of implementing computing systems will hit a barrier which will slow down their development, as the electronic components will start suffering from quantum interferences, and their design and physical implementation will become increasingly difficult.
One of the proposed solutions to overcome these problems is to try and change the computing paradigm, and switch to one based on quantum information processing technologies that are based on the somehow strange laws of quantum mechanics [4][20]. It has been proven that although any classical computer can in theory be used for simulating a quantum computer, this simulation cannot be performed efficiently, i.e. the increase in processing time depends at most polynomial on the input size. Therefore, the quantum computers may provide a significant advantage in the processing speed [8]. This advantage is so great that, according to some scientists, the classical computers will never be able reach the same level of performance as the quantum computers. But, such powerful quantum computers are still a distant way from being physically ready for processing significant problems, with relevant input sizes. For now, all that is available are just promising prototypes.

1.1. Church – Turing postulate

The theoretical basis for classical computer science were developed by Alan Turing in 1936, who introduced a model of information processing machine, now widely known under the name of *Turing Machine*. He also proved the existence of a *Universal Turing Machine*, which is capable of simulating any other Turing Machine [52]. Furthermore, he postulated that this Universal Turing Machine can implement any process defined in an algorithmic manner. That is, if a problem has an algorithm that can be implemented by a physical system (such us a modern computer for example), then there is an equivalent algorithm for a Universal Turing Machine that can resolve exactly the same problem [73]. This is known as the *Church-Turing postulate*:

Any algorithmic process can be simulated using a Turing Machine.

Starting from the idea that, not only a Turing Machine can simulate any algorithmic process, but in addition, it can do this in an efficient way, a stronger version of this postulate has been formulated:

Any algorithmic process can be efficiently simulated using a Turing Machine.

So, any problem that can be resolved efficiently in any arbitrarily chosen computing model can also be efficiently implemented on a Turing Machine. If this assumption (postulate) is correct, it follows that regardless of the machine used to run an algorithm, that machine can be efficiently simulated by a Turing Machine. This implies that for analyzing if a

computational process can be efficiently implemented, it is sufficient to analyze that problem using the model based on Turing Machines.
A first possible provocation to the strong postulate above was raised from an analogic computing perspective. It was noted that some types of analogical computers can resolve efficiently some problems that are believed to have no efficient solution on Turing Machines. At a first glance these analogical computers seem to violate the strong postulate, but in the end it was proven that their efficiency fades away if one takes into account a real working environment, that includes noise.
This lesson – that the noise effects within realistically limits must be taken into account when evaluating the efficiency of a computing model – is one of the biggest provocations that the quantum computing paradigm must face. For dealing with the noise problems, there are two directions of research: quantum errors correction and robust quantum computing. So, unlike the analogic computing, the quantum computing paradigm can, at least in theory, tolerate finite noise levels, without losing its advantage in computational efficiency.
The first major provocation to the strong Church – Turing postulate came in the 70s, when Robert Solovay and Volker Strassen showed that it is possible to verify that a number is prime or not, by using a probabilistic algorithm that manipulated purely random numbers; the solution was of course obtained only with finite probability. Therefore, this implies that if a computer has access to a real (as opposed to pseudo) random number generator, it can perform computations with efficiency greater than the one known for a classical, deterministic, Turing Machine.
Yet, this type of provocation can be easily resolved by integrating the probabilistic concepts into the postulate:

Any algorithmic process can be efficiently simulated using a probabilistic Turing Machine.

Or, to use the formulation from the complexity theory:

Any computing model can be simulated using a probabilistic Turing Machine by adding at most a polynomial number of operations.

The question that was then raised was: isn't possible in the future another computing model who could resolve problems even more efficiently than a probabilistic Turing Machine? Isn't possible to come up with a single, generic model that can be rigorously proved to efficiently simulate any other computing model?

1.2. Church – Turing – Deutsch postulate

Starting from these questions, in 1985 David Deutsch tried to find a method to deduct an even stronger version of the Church – Turing postulate, starting from the physical laws [19]. So, as long as the respective physical laws are considered valid, this new version of the postulate would be also valid. More precisely, Deutsch tried to find a computational model that is capable of emulating any physical process; hence, any computing machine which is built using these physical processes can be emulated by that model. The physical laws chosen by Deutsch were the quantum mechanics laws, which is debatable if they really are the ultimate laws of physics. And the computing machine that would be built according to these quantum laws is now obviously called quantum computer.
It is not clear yet though if Deutsch's notion on universal quantum computer is enough to emulate any arbitrarily chosen physical system. Confirming or infirming this hypothesis is one of the great challenges confronting the scientific community working in this area. It could be possible, for example, that some computing process defined based on quantum

fields theory, or relativity theory, or string theory, or some other physical theory, to overcome the capabilities of the universal quantum machine, providing then an even more powerful computing model.
The quantum computing model proved to be a provocation for the strong Church-Turing postulate, because Deutsch proved that it is possible for a quantum computer to resolve efficiently computing problems that have no known efficient solution on a classical computer, simulated by a probabilistic Turing Machine. Therefore, following these results, the new Church – Turing – Deutsch postulate can be formulated as [20]:

Any physical algorithmic process can be simulated efficiently using a Quantum Machine.

These Deutsch's first results were improved with more examples in the next decades. From these, the most important are Peter Shor's results on factoring numbers and the discrete logarithm problem. Both problems have no efficient solution on classical probabilistic computers, but have probabilistic efficient solutions in the quantum computing model [44].
Another result that demonstrates the power of quantum computers was proved by Lov Grover [36], which tackled a very important problem: searching in an unstructured space. This problem too can be solved more efficiently on a quantum computer than on a classical one (as far as the known classical algorithms go), but the obtained speed up it is not as dramatic as in the case of factoring numbers.
During the same time, other researchers proved an idea introduced by Richard Feynman back in 1982: the simulation of quantum mechanical systems on classical computers suffers from fundamental performance issues. Yet these can be eliminated by moving the respective simulations on quantum computers.
If there still are other types of problems that could be resolved more efficiently by algorithms for quantum computers is still a big open question. Discovering new such algorithms is difficult firstly because they require a special kind of thinking, oriented towards quantum mechanics principles, and secondly because they have to outperform all known algorithms from the classical, including probabilistic, computing [21].

1.3. Quantum information theory

The information communication theory, as developed by Claude Shannon in 1948 is another aspect that would require some modifications when switching to the quantum models of computation [43]. There are two main problems addressed by Shannon's theory:

- the resources required for transmitting of information across a communication channel,
- the ways to protect the sent information against corruption from the noise, which inevitably existent across the channel.

For resolving these two problems, Shannon proposed and proved two fundamental theorems:

- the theorem for information transmission across an ideal (i.e. noiseless) channel: this quantifies the resources that are required to store the received information.
- the theorem for information transmission across a real (i.e. noisy) channel: this quantifies the information that can be sent across a noisy communication channel and provides error correction codes for protecting this information against the noise.

The quantum theory of information was developed in a similar way [6]. In 1995, Ben Schumacher presented the quantum analogue for the first Shannon theorem: information transmission across a noiseless quantum channel, by defining the notion of quantum bit as a physical resource. But, an equivalent for the second theorem, transmission across a noisy channel, is yet to be discovered. Yet, there already is a quantum computing theory for errors

correction that was used for developing prototypes of quantum computers that operate in the presence of noise.
In 1992, Charles Bennett and Stephen Wiesner presented another application of the quantum information concepts: transmission of classical information (i.e. bits) through a quantum communications channel [7]. They proved the concept of supra-dense codification: how one can transmit two bits of classical information by sending just one qubit form the source to the destination.
Another open issue is the theory of interconnecting several quantum communication channels, and the development of quantum communication networks, that should contain quantum repeaters, switches, etc.

1.4. Quantum cryptography

Currently, the widest available cryptographic systems are based on public key cryptography, which are based on the hardly reversible mathematical functions. This kind of systems doesn't have to deal with the keys distribution problem. But many of them are based on the assumption that factorization is a computable-hard problem, which is not the case anymore as soon as one changes the paradigm and switches to quantum computers.
Another kind of cryptographic systems use secret keys, but they have to address the problem of distributing these keys across unsecured channels. And one way to address this is to use quantum communication channels.
The idea is based on the principle of measuring in quantum mechanics: observing a quantum system leads to inevitably and irreversible disturbing it [46]. This principle guarantees that if a key is compromised, the interested parties know about it. But this principle solves a problem and raises another: it is very difficult, if at all possible, to introduce a repeater in such a quantum communication channel; which means the channel has a quite limited length. Yet, some quantum cryptographic systems are already making their ways into the commercial applications.

1.5. Quantum information representation

The bit is the fundamental concept in the classical information and computing theory. Quantum computing and quantum information processing uses a similar fundamental concept: the quantum bit, or qubit [22].
Abstracting away the physical implementation of qubits, they are defined as mathematical entities. Using the Dirac notation, two possible states of a qubit, correspondents of the classical bit states 0 and 1, are denoted as $|0\rangle$ and $|1\rangle$ – which are named as computational basis states. Unlike a classical bit though, a qubit can be in infinity of other states. More precisely, it can be in any normal state, defined as a complex linear combination of the computational basis states. This state, noted with $|\psi\rangle$ is called superposition:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers, satisfying the normality relation: $|\alpha|^2 + |\beta|^2 = 1$
Therefore, the qubit state is represented by a normed vector in a bi-dimensional complex vector space, where the two computational basis states form an orthonormal basis.
By measuring this qubit one can obtain the result 0 with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$.

A conceptual example of a qubit is represented by an electron that orbits a nucleus, according to the Hydrogen atom model, with the two computational basis states being represented by the basis and the excited states:

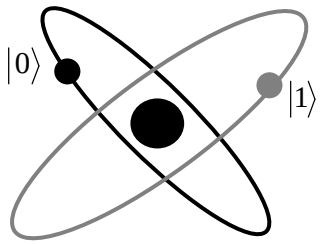


Figure 1. The Hydrogen atom model

Because α and β can have an infinity of values, it may look like a qubit can represent an infinite quantity of information. But this is not true, because what matters are the values that can be observed, and these are just two of them. Furthermore, after measuring the qubit its state will collapse into $|0\rangle$ or $|1\rangle$. The only way to measure α and β with infinite precision is to make an infinity of measurements on a an infinity of identically prepared qubits, which is not possible.

For representing systems of more than one qubits, the tensor products are used. So, the state of a system on n qubits, any qubit being in the state $|\psi_i\rangle$, is represented by the normed vector:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ..... \otimes |\psi_n\rangle$$

1.6. Practical implementations of quantum computing systems

There are two main issues that have to be considered when talking about practical implementations of such systems:

- the real environments contain noise and this may affect the computing system
- quantum mechanics may not be the theory that correctly and completely describes the physical reality

Regarding the first issue, it has been proved that noise doesn't represent problem of principle in physical implementation of quantum computers (like it did for analog computers). That is, if the noise can be kept under a fixed value, the quantum correction codes can be used to decrease the noise even further; and this correction process can be repeated indefinitely without fundamentally affecting the efficiency.

Yet, until now, quantum computers were only implemented on small scales, of just a few qubits.

1.7. Quantum computing – quantum gates

The quantum computing studies the transformations performed on qubits' quantum states. In the same way as a classical computer is built from electronic circuits containing interconnected logical gates, a quantum computer is built from quantum circuits containing interconnected quantum gates. The mathematical formalism used for describing the transformations applied by the quantum circuits on the qubits is the theory of linear operators. Since the state of a qubit is a normed vector in a bi-dimensional complex space, a quantum circuit processing that qubit is a unitary linear operator on that same vector space. This definition is extended to circuits processing more than one qubits. The operator describing a

circuit has to be unitary so that at the circuit exit we find a transformed qubit, but still a qubit. That means that after applying the operator, the state vector is still normed.
Because any unitary operator is bijective and reversible (its inverse operator being its adjoint), it follows that all circuits are reversible: knowing the states of the qubits at the circuit's exit, one can calculate the states of the qubits at the entrance in the circuit.
This property doesn't hold for all classical circuits. Consider for example the NAND gate. And yet this doesn't confer to the classical circuits any computing advantage. This is because there is a reversible gate on three bits (the universal Toffoli gate) which can implement any classical circuit.

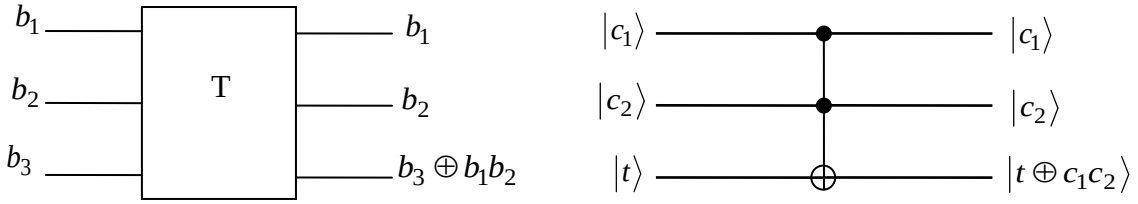


Figure 2. The classical Toffoli gate and its quantum correspondent

The Toffoli gate flips the third bit if and only if the first two bits are set.
Another very important gate in quantum computing is the two qubit variant of the Toffoli gate, named CNOT (conditional NOT) gate:

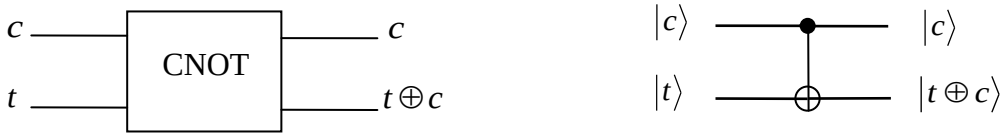


Figure 3. CNOT classical gate and its quantum correspondent

Hence the CNOT gate is said to implement addition modulo 2 on two bits. Here is a circuit that implements the addition modulo 4 on 2 bits: $|x_1 x_0 y_1 y_0\rangle \to |x_1 x_0\rangle|(x_1 x_0 + y_1 y_0) \bmod 4\rangle$, where $x_1, x_0, y_1, y_0 \in \{0,1\}$

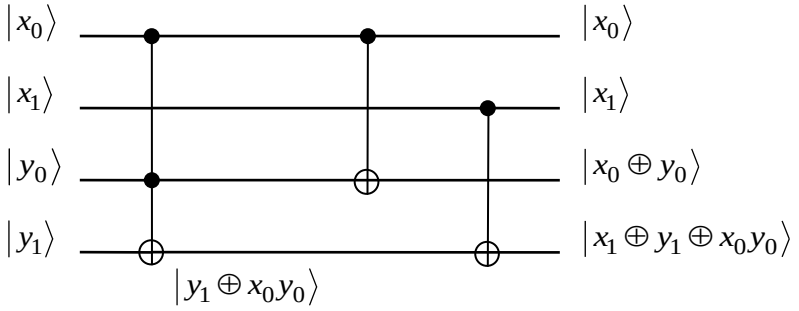


Figure 4. The quantum circuit for implementing addition modulo 4

It is very important to mention that measuring one or more qubits is not a unitary operation. Therefore a physical device performing the measurement cannot be represented by a quantum circuit. After performing the measurement, the state of the respective qubits is not relevant any more, what is relevant is the information obtained from the measurements – this information being represented by classical probabilistic bits.

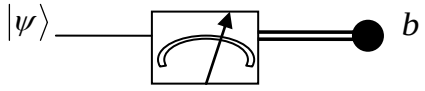


Figure 5. The representation of a measurement on a qubit

Here are a few examples of quantum gates on one qubit:

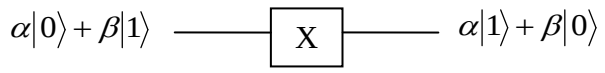


Figure 6. NOT gate

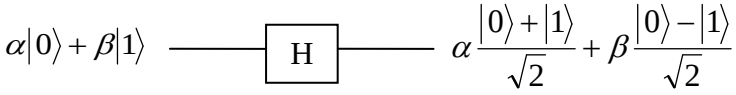


Figure 7. Hadamard gate

It is worth noting that the lines in quantum circuits don't represent physical lines in a quantum computer (like electrical wires or anything similar). They represent the life of a qubit during time. This is why it is not accepted for the quantum circuits to have loops. They are always direct acyclic graphs.

1.8. The impossibility of copying a qubit

In classical computing paradigm, there is a very simple circuit for copying an arbitrary bit.

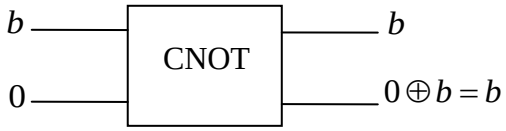


Figure 8. Classical circuit for copying a bit

One of the most striking differences between the classical circuits and the quantum ones is the fact that it is impossible to build a quantum circuit for copying exactly a qubit in an arbitrary state [59]. This is known as the non-cloning theorem for qubits.
The only quantum circuits that can be built for exactly copying qubits, are just the ones for duplicating two orthonormal states of a qubit.
Indeed, if one supposes there is such a quantum circuit on two qubits, this would lead to contradictions with some fundamental physical principles: the possibility of distinguishing through measurement of the un-orthogonal states, or even the possibility of transferring information with the speeds greater than the speed of light.
Such a quantum circuit would need to transform the initial state $|\psi\rangle|\psi_t\rangle$ in the state $|\psi\rangle|\psi\rangle$, for any state $|\psi\rangle$, using only unitary transformations. So, there would exist the unitary transformation U, such that $U|\psi\rangle|\psi_t\rangle=|\psi\rangle|\psi\rangle$, for any $|\psi\rangle$. Suppose the circuit would copy states $|\psi_1\rangle$ and $|\psi_2\rangle$:

$$\begin{cases} U|\psi_1\rangle|\psi_t\rangle=|\psi_1\rangle|\psi_1\rangle \\ U|\psi_2\rangle|\psi_t\rangle=|\psi_2\rangle|\psi_2\rangle \end{cases}$$

Calculating the scalar product of the above relations it results:
$$\left(U|\psi_1\rangle|\psi_t\rangle,U|\psi_2\rangle|\psi_t\rangle\right)=\left(|\psi_1\rangle|\psi_1\rangle,|\psi_2\rangle|\psi_2\rangle\right)\Rightarrow\langle\psi_t|\langle\psi_1|U^{\dagger}U|\psi_2\rangle|\psi_t\rangle=\langle\psi_1|\psi_2\rangle\langle\psi_1|\psi_2\rangle$$
Considering U is a unitary operator, and $|\psi_t\rangle$ is a normed state, it follows:

$$\langle\psi_1|\psi_2\rangle^2=\langle\psi_1|\psi_2\rangle\Leftrightarrow\langle\psi_1|\psi_2\rangle\left(\langle\psi_1|\psi_2\rangle-1\right)=0$$

Which means this is one of the following two cases:

- $\langle\psi_1|\psi_2\rangle=1$, that is the states to be copied are identical: $|\psi_1\rangle=|\psi_2\rangle$

or

- $\langle\psi_1|\psi_2\rangle=0$, that is the states to be copied are orthonormal: $|\psi_1\rangle\perp|\psi_2\rangle$

Yet, the CNOT quantum circuit can be used to copy a qubit whose state is $|c\rangle$, where $c \in \{0,1\}$.

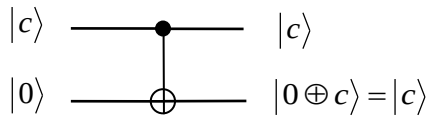


Figure 9. Quantum circuit for copying orthonormal states $|0\rangle$ or $|1\rangle$

Even if unitary transformations are accepted, it still remains valid that only orthogonal states can be copied. It has been proved that a quantum circuit for copying un-orthogonal states can be developed only if one accepts approximate copies.

2. Quantum computing efficiency

2.1. Quantum parallelism

Quantum parallelism is a fundamental characteristic of most of the quantum computing algorithms.

For example, considering a function $f:\{0,1\}\mapsto\{0,1\}$ that takes a bit into another bit. The purpose is to find a way to compute the two values this function takes, by using a quantum circuit on two qubits, a data qubit and a destination qubit. Such a circuit would perform the transformation U_f described by $|x,y\rangle \to |x, y\oplus f(x)\rangle$, where $|x,y\rangle$ is a computational basis state:

$$|00\rangle \to |0, f(0)\rangle$$
$$|01\rangle \to |0, \neg f(0)\rangle$$
$$|10\rangle \to |1, f(1)\rangle$$
$$|11\rangle \to |1, \neg f(1)\rangle$$

Because $f:\{0,1\}\mapsto\{0,1\}$ there are four possible such functions $|\{0,1\}|\times|\{0,1\}| = 2\times 2 = 4$:

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

For each such function, its transformation matrix is:

$$U_{f_{00}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; U_{f_{01}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}; U_{f_{10}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; U_{f_{11}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

These can be put in a single, parameterized matrix definition:

$$U_{f_{ij}} = \begin{bmatrix} \neg i & i & 0 & 0 \\ i & \neg i & 0 & 0 \\ 0 & 0 & \neg j & j \\ 0 & 0 & j & \neg j \end{bmatrix}$$

which is a unitary matrix: $\overline{U_{f_{ij}}^T} \times U_{f_{ij}} = I_4$ and therefore it can be implemented by a quantum circuit:

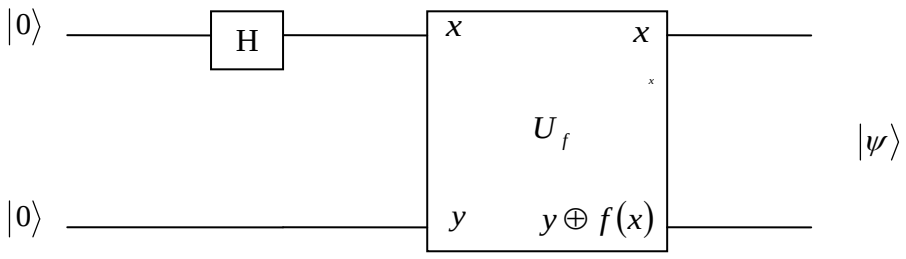


Figure 10. Quantum circuit for evaluating *f*

The circuit provides the following succession of states:

$$|00\rangle \xrightarrow{H} \left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]|0\rangle \xrightarrow{U_f} \frac{|0, f(0)\rangle+|1, f(1)\rangle}{\sqrt{2}} = |\psi\rangle$$

The two terms in the final state contain information about both values the function can take, $f(0)$ and $f(1)$, and yet the function f was applied only once. So, this looks as if f has been evaluated simultaneously for both input values. This characteristic is known as quantum parallelism, and is different from the parallelism in the classical computing paradigm. This new kind of parallelism makes use of the quantum computers capability of being in a superposition of states.

This procedure can be easily generalized to functions on any number of bits, using a general Walsh-Hadamard transformation, implemented with n Hadamard gates, performing in parallel on n qubits. This transformation is noted as $H^{\otimes n}$.

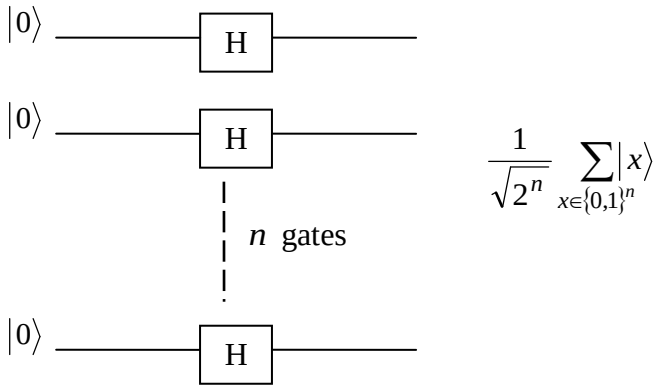


Figure 11. Walsh-Hadamard transformation on *n* qubits

Applying this transformation on a set of n qubits, each in state $|0\rangle$, the final state is:

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

Therefore, the Walsh-Hadamard transformation produces a balanced superposition of all the computational basis states. Furthermore, this is done in a very efficient way, by producing a superposition of 2^n states and using only n gates.

Quantum parallel evaluation of a function $f : \{0,1\}^n \mapsto \{0,1\}$ can be then done as following.

One prepares a state on $n+1$ qubits: $|0\rangle^{\otimes n}|0\rangle$ then the Walsh-Hadamard transformation is applied on the first n qubits followed by the quantum circuit implementing U_f. The sequence of states is:

$$|0\rangle^{\otimes n}|0\rangle \xrightarrow{H^{\otimes n}} \left[\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\right]|0\rangle \xrightarrow{U_f} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|\, f(x)\rangle$$

Although it looks like the quantum parallelism allows for evaluating all the possible values of a binary function f, by applying the function only once, this is not necessarily useful solely by itself. This is because to find these values, measurements need to be made, and each measurement can reveal only one value. For example, for the two bit case measuring the final state will return a final state that is $|0, f(0)\rangle$ *or* $|1, f(1)\rangle$! Similarly, in the generic case, measuring state $\sum_{x\in\{0,1\}^n} |x\rangle|\, f(x)\rangle$ returns only $f(x)$ for a single x value. This is not very

different for what a classical computer could do. So, something more than the mere quantum parallelism is required in order to make the quantum computing more efficient that the classical correspondent. It is necessary to find a way to extract the desired information in an efficient way. One example for achieving this is Deutsch's algorithm.

2.2. Deutsch's algorithm

A simplified version of this algorithm can be used to demonstrate the way quantum computers can outperform the classical counterparts. The algorithm, implemented by the quantum circuit below, uses quantum parallelism combined with quantum interference.

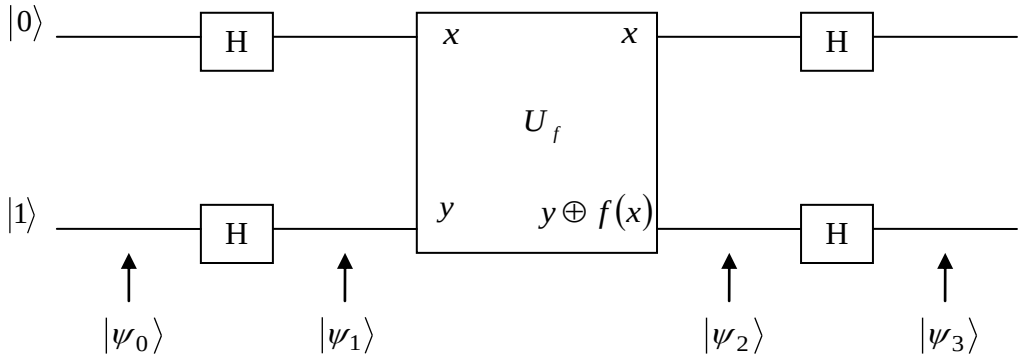


Figure 12. Quantum circuit implementing Deutsch algorithm

The two qubits in the circuit start in the computational basis state:

$$|\psi_0\rangle = |0\rangle|1\rangle$$

After applying the Hadamard gate, the state becomes:

$$|\psi_1\rangle = \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

By defining the transformation $|x, y\rangle \xrightarrow{U_f} |x, y \oplus f(x)\rangle$ the following equation follows:

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

$$= \begin{cases} |x\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], \text{if } f(x) = 0 \\ |x\rangle\left[-\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], \text{if } f(x) = 1 \end{cases} = (-1)^{f(x)}|x\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

Applying this transformation to the state $|\psi_1\rangle$ the new state becomes:

$$|\psi_2\rangle = \left[\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = (-1)^{f(0)}\left[\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

$$|\psi_2\rangle = \begin{cases} \pm\left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], \text{if } f(0) = f(1) \\ \pm\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right], \text{if } f(0) \neq f(1) \end{cases}$$

The final Hadamard gates take the qubits in the state:

$$|\psi_3\rangle = \begin{cases} \pm|0\rangle|1\rangle, \text{dacă } f(0) = f(1) \\ \pm|1\rangle|1\rangle, \text{dacă } f(0) \neq f(1) \end{cases}$$

And by using the summation modulo 2, this final state can be written as:

$$|\psi_3\rangle = \pm|f(0)\oplus f(1)\rangle|1\rangle$$

So, by measuring the first qubit, one can evaluate directly the sum modulo 2: $f(0)\oplus f(1)$.
This means the quantum circuit above computes a global propriety of the input function $f(x)$, $f:\{0,1\}\mapsto\{0,1\}$, namely $f(0)\oplus f(1)$, by performing just one evaluation of the function $f(x)$. This is twice as fast as it is possible with a classical computing machine, as this would need at least two evaluations: $f(0)$ and $f(1)$.
One can also deduce an important difference between the quantum parallelism and the classical probabilistic algorithms. At a first glance, one could assume the quantum state $\frac{|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle}{\sqrt{2}}$ corresponds closely to a classical probabilistic computer that evaluates $f(0)$ with probability $\frac{1}{2}$, or $f(1)$ with probability $\frac{1}{2}$. But the fundamental difference is that in the classical case, the two cases are mutually exclusive, while in the quantum computer the two cases are interfering with each other to produce the global propriety. The essence of many quantum algorithms consists in the proper choosing of the desired function and of the final transformations.

2.3. Deutsch-Jozsa algorithm

This is the generalized version of the previous algorithm, tackling the following problem.

2.3.1. Deutsch-Jozsa problem

Consider a function $f:\{0,1\}^n \mapsto \{0,1\}$ which is either constant or balanced [29]. Find an efficient deterministic algorithm that decides the type of the function.
f is constant iff it satisfies one of the two conditions:

$$(\forall)x \in \{0,1\}^n \Rightarrow f(x) = 0$$
$$(\forall)x \in \{0,1\}^n \Rightarrow f(x) = 1$$

f is balanced iff there exist sets $A \subset \{0,1\}^n$ and $B \subset \{0,1\}^n$ that simultaneously satisfy the following properties:

$$A \cup B = \{0,1\}^n$$
$$(\forall)x \in A \Rightarrow f(x) = 0$$
$$(\forall)x \in B \Rightarrow f(x) = 1$$

$$|A| = |B| = 2^{n-1}$$

Using a classical algorithm, this problem can be solved like this: generate sequentially the elements $x \in \{0,1\}^n$ and for each of them compute $f(x)$. If at the current step the value obtained is different from the previous one, the function is balanced. If after $2^{n-1}+1$ were computed, the same value was obtained, the function is constant. The classical algorithm is:

1. choose $x \in \{0,1\}^n$
2. initialize $A = \{0,1\}^n - \{x\}$
3. initialize $B = \{x\}$
4. initialize $y_0 = f(x)$
5. if $|B| > 2^{n-1}$ then return „ f constant"
6. choose $x \in A$
7. compute $y = f(x)$
8. if $y \neq y_0$ then return „ f balanced"
9. $y_0 \leftarrow y$
10. $A \leftarrow A - \{x\}$
11. $B \leftarrow B \cup \{x\}$
12. go to step 5.

So, in the worst case, when f is constant, this algorithm has exponential complexity $O(2^{n-1}+1)$. The problem can be formulated in probabilistic terms, in which case it has a more efficient solution.

2.3.2. Probabilistic Deutsch-Jozsa problem

Consider a function $f : \{0,1\}^n \mapsto \{0,1\}$ which is known to be either constant or balanced. $\forall \varepsilon > 0$, error probability, find an efficient probabilistic algorithm that decides the type of the function with probability at least $1-\varepsilon$.

The classical probabilistic algorithm is similar with the deterministic one presented above. The main difference is the way of choosing $x \in A$:

- in the deterministic case, the elements are chosen sequentially (note A is an ordered set: $(\forall)x, y \in A \Rightarrow x \leq y$ or $y \leq x$)
- in the probabilistic case, the elements are chosen purely at random

Also, in the probabilistic case, because an exact output answer is not required, it is not necessary to compute all the $2^{n-1}+1$ values for f, but instead, a smaller number, dependent on $\varepsilon : M_\varepsilon < 2^{n-1}+1$. Therefore, step 5. from the deterministic classical algorithm above is replaced in the probabilistic case by:

.....

5. if $|B| > M_\varepsilon - 1$ then return „ f constant"

.....

Note this probabilistic algorithm never returns the wrong answer if the function is constant. But if the function is balanced, there is a non-zero probability that this algorithm will return the wrong answer. This can happen when at all the M_ε steps, the same value for f was computed.

When f is balanced, at the first pass through step 7. in the algorithm, the probability of obtaining the value $y_0 \in \{0,1\}$ when evaluating $y_0 = f(x_0)$ where $x_0 \in \{0,1\}^n$ is $p_0 = \frac{2^{n-1}}{2^n}$. At the second pass, the probability to obtain again $y_1 = y_0 \in \{0,1\}$ when evaluating $y_1 = f(x_1)$ where $x_1 \in \{0,1\}^n - \{x_0\}$ is $p_1 = p_0 \times \frac{2^{n-1}-1}{2^n-1}$. After M_ε passes, the error probability is therefore:

$$\varepsilon = p_{M_\varepsilon - 1} = \frac{2^{n-1}}{2^n} \times \frac{2^{n-1}-1}{2^n-1} \times \ldots \times \frac{2^{n-1}-M_\varepsilon+1}{2^n-M_\varepsilon+1} \leq \frac{1}{2} \times \frac{1}{2} \times \ldots \frac{1}{2} = \frac{1}{2^{M_\varepsilon}}$$

And this means that:

$$\log_2 \varepsilon \leq -M_\varepsilon$$

$$M_\varepsilon \leq \log_2 \frac{1}{\varepsilon}$$

So, by running the algorithm for $\log_2 \frac{1}{\varepsilon}$ evaluations of the input function, the correct answer is obtained with the chosen error probability. Hence the complexity of the classical probabilistic algorithm is $O\left(\log_2 \frac{1}{\varepsilon}\right)$ – which doesn't depend on the size of the input, only on the chosen error probability.

2.3.3. Deutsch-Jozsa quantum circuit

The generic deterministic Deutsch-Jozsa problem can be solved by the quantum circuit below.

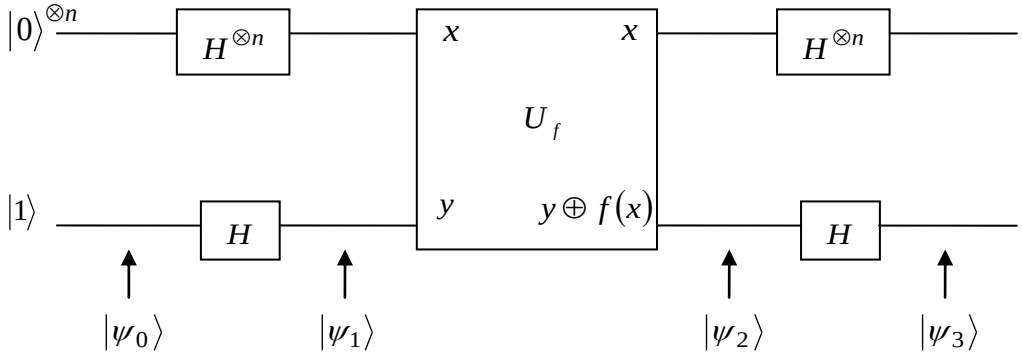


Figure 13. Quantum circuit that implements the generic Deutsch-Jozsa algorithm

The data register is on n qubits, because $f : \{0,1\}^n \mapsto \{0,1\}$. The initial state is:

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$$

Then, by applying the Hadamard gates, the new state is:

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle \xrightarrow{H^{\otimes n}H} |\psi_1\rangle = \left[\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle\right] \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

So, the data register is in a balanced superposition of the computational basis states and the target register is in another type of balanced superposition of $|0\rangle$ and $|1\rangle$.

And because,

$$|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \xrightarrow{U_f} (-1)^{f(x)}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

After passing the linear quantum circuit U_f, the state becomes:

$$|\psi_1\rangle=\left[\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \xrightarrow{U_f} |\psi_2\rangle=\left[\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

And because $H|x\rangle=\frac{\sum_{z\in\{0,1\}}(-1)^{x\cdot z}|z\rangle}{\sqrt{2}}$ it implies $H^{\otimes n}|x\rangle=\frac{\sum_{z\in\{0,1\}^n}(-1)^{x\cdot z}|z\rangle}{\sqrt{2^n}}$

Using these formulas, the new state can be computed as:

$$|\psi_2\rangle=\left[\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]\xrightarrow{H^{\otimes n}H}$$

$$|\psi_3\rangle=H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right]H\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

$$|\psi_3\rangle=\left[\sum_{z\in\{0,1\}^n}\sum_{x\in\{0,1\}^n}\frac{(-1)^{x\cdot z+f(x)}|z\rangle}{2^n}\right]|1\rangle$$

Because this is a quantum state, the amplitudes of the computational basis states forming the interference in the data register, has to satisfy the relation: the sum of all probabilities must be 1, so: $\sum_{i=0,2^n-1}|A_i|^2=1$.

The amplitude of state $|0\rangle^{\otimes n}$ is $A_0=\sum_{x\in\{0,1\}^n}\frac{(-1)^{f(x)}}{2^n}$ and there are two cases:

I. function f is constant. If $f(x)=0$ then $A_0=1$; if $f(x)=1$ then $A_0=-1$. In both sub-cases, the probability of obtaining 0 for each qubit when measuring state $|\psi_3\rangle$ is exactly 1.

II. function f is balanced. Then $\sum_{x\in\{0,1\}^n}(-1)^{f(x)}=0\Rightarrow A_0=0$. So, the probability of obtaining 0 for each qubit when measuring state $|\psi_3\rangle$, is exactly 0.

In conclusion, the generic Deutsch-Jozsa problem can be resolved by measuring the first n qubits from state $|\psi_3\rangle$. If all qubits are $|0\rangle$ then f is constant, if at least one qubit is $|1\rangle$ then f is balanced.

It has to be mentioned that in this quantum algorithm, the evaluation of f was performed only once. This means an exponential improvement in the worst case, in comparison to the deterministic classic algorithm, which may evaluate f, $2^{n-1}+1$ times.

2.4. Super-dense coding

Suppose Alice holds a classical information on two bits (coded as usual as $00, 01, 10, 11$), and she wants to send it to Bob using a communication channel. It is shown that this can be achieved by transferring only one qubit, if initially Alice and Bob share a pair of qubits in an EPR state: $|\psi_0\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$. Alice holds the first qubit and Bob holds the second. Note this is an initial fixed state, and it doesn't depend on the information to be sent. There is no need to transfer any qubits through the communication channel to prepare this state.

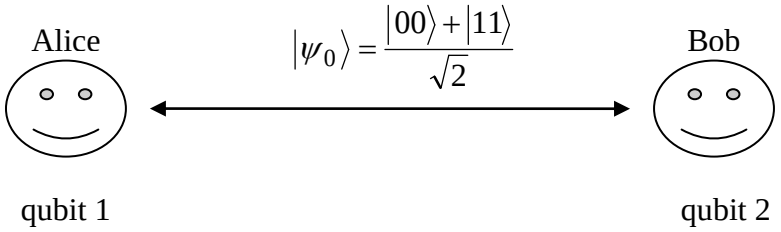


Figure 14. Initial state required for super-dense coding

Then, depending on the classical information she wants to send, Alice applies a Pauli operator on her qubit:

- for sending 00, apply $I = |0\rangle\langle 0| + |1\rangle\langle 1|$, the combined state being
$$|\psi_{00}\rangle = I \otimes I|\psi_0\rangle = \frac{I|0\rangle \otimes I|0\rangle + I|1\rangle \otimes I|1\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$
- for sending 01, apply $X = |0\rangle\langle 1| + |1\rangle\langle 0|$, the combined state being
$$|\psi_{01}\rangle = X \otimes I|\psi_0\rangle = \frac{X|0\rangle \otimes I|0\rangle + X|1\rangle \otimes I|1\rangle}{\sqrt{2}} = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$
- for sending 10, apply $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$, the combined state being
$$|\psi_{10}\rangle = Z \otimes I|\psi_0\rangle = \frac{Z|0\rangle \otimes I|0\rangle + Z|1\rangle \otimes I|1\rangle}{\sqrt{2}} = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$
- for sending 11, apply $iY = |0\rangle\langle 1| - |1\rangle\langle 0|$, the combined state being

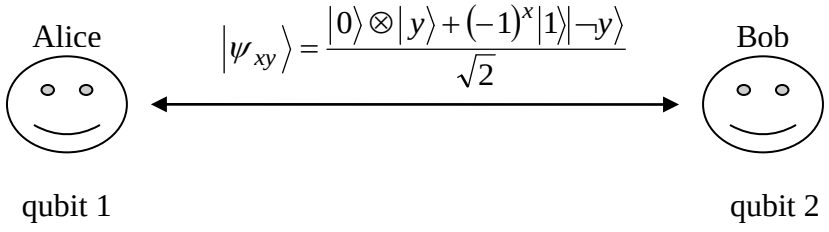


Figure 15. Combined state after applying one Pauli operator

Because $\langle \psi_{xy} | \psi_{zt} \rangle = \delta_{xy,zt}$, states $|\psi_{xy}\rangle$ (known as Bell state, Bell basis or EPR pairs) are orthonormal, hence they form an orthonormal state in the state space $|\psi\rangle$.
Next, Alice sends her qubit to Bob.

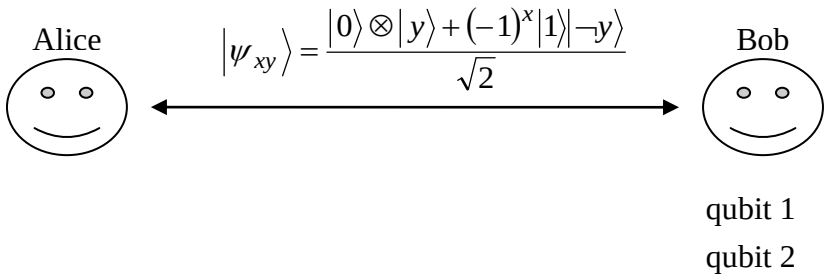


Figure 16. Bob holds now both qubits

Having both qubits, Bob can now perform a projective measurement in Bell basis, using for measurement operators (projections):

$$P_{xy} = |\psi_{xy}\rangle\langle\psi_{xy}|, \text{ where } x = 0,1 \text{ and } y = 0,1$$

Assuming state $|\psi_{xy}\rangle$ is the real one, the probability of obtaining result xy is in this case:

$$p(xy) = \langle\psi_{xy}|P_{xy}|\psi_{xy}\rangle = 1;$$

while if the real state is $|\psi_{tz}\rangle$, the probability of obtaining xy is:

$$p(xy) = \langle\psi_{tz}|P_{xy}|\psi_{tz}\rangle = 0, (\forall) tz \neq xy$$

So, bob can precisely calculate value xy which represents the very information Alice wanted to send him.

Because the projective measurement doesn't in this case actually change the state of the system at all:

$$|\psi_1\rangle = \frac{P_{xy}|\psi_{xy}\rangle}{\sqrt{p(xy)}} = \frac{|\psi_{xy}\rangle\langle\psi_{xy}|\psi_{xy}\rangle}{1} = |\psi_{xy}\rangle$$

it means the received information can be read as many times as necessary. It is guaranteed each time the same value will be read.

A generalization for this protocol is possible: for sending $2n$ bits of information, Alice and Bob must share an entangled state on $2n$ qubits: n qubits are in Alice's possession and the other n qubits are Bob's. each time Alice wants to send two bits of information, she applies a Pauli operator upon one qubit and sends the resulted qubit to Bob. In turn Bob applies a projective measurement on the state resulted from combining the received qubit with its entangled pair.

Another characteristic of this protocol is related to its security. If Eve listens on the communication channel used by Alice and Bob, she intercepts the qubit Alice sends and wants to calculate the respective information.

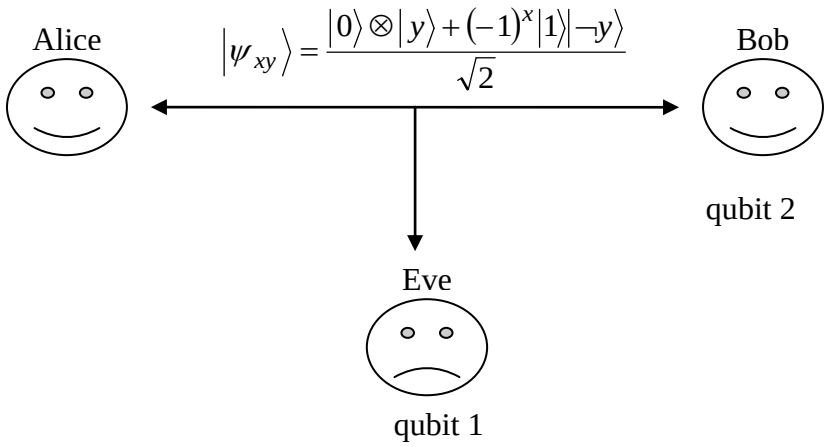


Figure 17. Eve intercepts the qubit

To achieve this, Eve will have to perform a measurement on the intercepted qubit. So, she defines a set of positive operators E_m to apply on qubit 1. Assuming the real state is $|\psi_{xy}\rangle$, the probability of obtaining result m after such a measurement is in this case:

$$p(m) = \langle \psi_{xy} | E_m \otimes I | \psi_{xy} \rangle = \left(\frac{|0\rangle \otimes |y\rangle + (-1)^x |1\rangle |\neg y\rangle}{\sqrt{2}}, \frac{E_m |0\rangle \otimes |y\rangle + (-1)^x E_m |1\rangle |\neg y\rangle}{\sqrt{2}} \right) =$$

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

So, the probabilities of obtaining each of the results are equal, regardless of the real state before the measurement. In conclusion, because Eve can't compute the information sent by Alice, this protocol is guaranteed to be secure as long as Bob's qubit is kept secret.

2.5. Quantum teleportation

Quantum teleportation [12] is a technique for transmitting quantum states that can be used even without the need of a quantum communication channel between the sender and the receiver.

Alice wants to send to Bob a qubit in an arbitrary state $|\psi\rangle$. The restrictions Alice must obey are:

- Alice doesn't know the state to be sent. She can't even find it out because she has only one copy of the qubit to be sent, and therefore she can't perform a measurement.
- Alice and Bob share only one digital classical communication channel. So no qubit can be sent on it.

The solution is based again on the assumption that Alice and Bob share an initial a pair of qubits in EPR state: $|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$.

The steps of the communication protocol are [34]:

Pas 1. Alice and Bob initialize the EPR pair. Alice takes one qubit, Bob takes the other one.

Pas 2. Alice combines the qubit to be sent, which is in state $|\psi\rangle$ with the first EPR qubit. Alice obtains then one of the following results: 00, 01, 10, or 11.

Pas 3. Alice sends these two bits to Bob.

Pas 4. Depending to the received bits, Bob performs a transformation (one of four possible) on his own EPR qubit. The final state of this qubit will be the very initial state $|\psi\rangle$.

The quantum circuit representing this protocol is presented below.

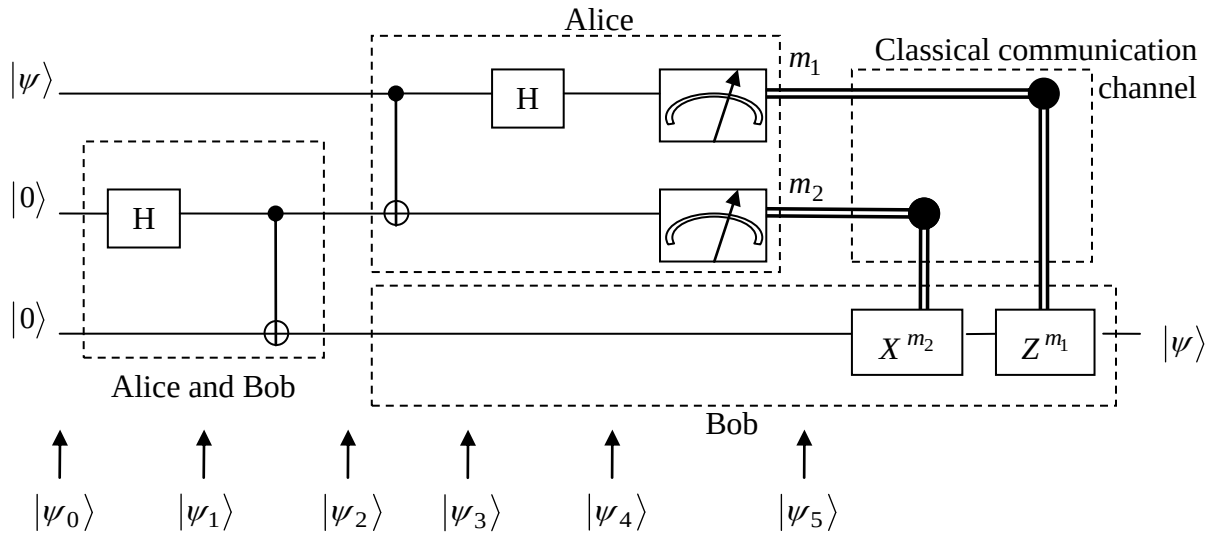


Figure 18. Quantum circuit for qubit teleportation

Initially, Alice and Bob prepare an EPR pair, using a circuit with two gates: one Hadamard, one CNOT acting on two qubits, each of them in the computational basis state $|0\rangle$. The initial state is:

$$|\psi_0\rangle = |\psi\rangle|0\rangle|0\rangle$$

After applying Hadamard gate, the state becomes:

$$|\psi_1\rangle = |\psi\rangle\frac{|0\rangle+|1\rangle}{\sqrt{2}}|0\rangle = |\psi\rangle\frac{|00\rangle+|10\rangle}{\sqrt{2}}$$

Then, CNOT gate inverts the second qubit iff the first one is in state $|1\rangle$. The next state is therefore:

$$|\psi_2\rangle = |\psi\rangle\frac{|00\rangle+|11\rangle}{\sqrt{2}} = |\psi\rangle|\beta_{00}\rangle$$

Now, Alice and Bob share an EPR pair and therefore Alice and Bob can now move away from each other, taking one qubit each. They can communicate form now only on the classical digital channel. Until now, the target qubit to be sent was left untouched, in its initial state:

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

The combined three qubit system state is:

$$|\psi_2\rangle = \left(a|0\rangle + b|1\rangle\right)\frac{|00\rangle+|11\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}\left[a|0\rangle\left(|00\rangle+|11\rangle\right)+b|1\rangle\left(|00\rangle+|11\rangle\right)\right]$$

Alice applies another CNOT gate, the new state being:

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}\left[a|0\rangle\left(|00\rangle+|11\rangle\right)+b|1\rangle\left(|10\rangle+|01\rangle\right)\right]$$

Then, the target qubit goes through a Hadamard gate:

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}\left[a\frac{|0\rangle+|1\rangle}{\sqrt{2}}\left(|00\rangle+|11\rangle\right)+b\frac{|0\rangle-|1\rangle}{\sqrt{2}}\left(|10\rangle+|01\rangle\right)\right] \Rightarrow$$

$$|\psi_4\rangle = \frac{1}{2}\left[a\left(|0\rangle+|1\rangle\right)\left(|00\rangle+|11\rangle\right)+b\left(|0\rangle-|1\rangle\right)\left(|10\rangle+|01\rangle\right)\right]$$

which is equivalent to:

$$|\psi_4\rangle = \frac{1}{2}\left[a|000\rangle+a|011\rangle+a|100\rangle+a|111\rangle+b|010\rangle+b|001\rangle-b|110\rangle-b|101\rangle\right]$$

$$|\psi_4\rangle = \frac{1}{2}\left[|00\rangle\left(a|0\rangle+b|1\rangle\right)+|01\rangle\left(a|1\rangle+b|0\rangle\right)+|10\rangle\left(a|0\rangle-b|1\rangle\right)+|11\rangle\left(a|1\rangle-b|0\rangle\right)\right]$$

Next, Alice performs a measurement on the two qubits she holds. The measurement is in the base formed from orthonormal vectors $|m_1m_2\rangle$, where $m_1, m_2 \in \{0,1\}$, i.e. $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. According to the quantum measurement principles, projectors $P_{m_1m_2} = |m_1m_2\rangle\langle m_1m_2|$ are applied on Alice's two qubits, obtaining the following results:

- 00 with probability $p(00) = \langle\psi_4|P_{00}\otimes I_2|\psi_4\rangle = \frac{1}{4}$, resulting state being

  $$|\psi_5(00)\rangle = \frac{P_{00}\otimes I_2|\psi_4\rangle}{\sqrt{p(00)}} = |00\rangle\left(a|0\rangle+b|1\rangle\right)$$

- 01 with probability $p(01) = \langle\psi_4|P_{01}\otimes I_2|\psi_4\rangle = \frac{1}{4}$, resulting state being

  $$|\psi_5(01)\rangle = \frac{P_{01}\otimes I_2|\psi_4\rangle}{\sqrt{p(01)}} = |01\rangle\left(a|1\rangle+b|0\rangle\right)$$

- 10 with probability $p(10) = \langle\psi_4|P_{10}\otimes I_2|\psi_4\rangle = \frac{1}{4}$, resulting state being

  $$|\psi_5(10)\rangle = \frac{P_{10}\otimes I_2|\psi_4\rangle}{\sqrt{p(10)}} = |10\rangle\left(a|0\rangle-b|1\rangle\right)$$

- 11 with probability $p(11) = \langle\psi_4|P_{11}\otimes I_2|\psi_4\rangle = \frac{1}{4}$, resulting state being

  $$|\psi_5(11)\rangle = \frac{P_{11}\otimes I_2|\psi_4\rangle}{\sqrt{p(11)}} = |11\rangle\left(a|1\rangle-b|0\rangle\right)$$

Alice now sends to Bob the result from her measurement, as two bits m_1m_2, using the classical communication channel. Depending on the Alice's measurement result, Bob's qubit will be in one of the following states:

- $00 \mapsto |\psi_{5B}(00)\rangle = a|0\rangle+b|1\rangle$
- $01 \mapsto |\psi_{5B}(01)\rangle = a|1\rangle+b|0\rangle$
- $10 \mapsto |\psi_{5B}(10)\rangle = a|0\rangle-b|1\rangle$
- $11 \mapsto |\psi_{5B}(11)\rangle = a|1\rangle-b|0\rangle$

Depending on the two received qubits, Bob can now re-constitute the target qubit, in its original state $|\psi\rangle$:

- If $m_1m_2 = 00$, Bob doesn't need to do anything because his qubit is already in state $|\psi\rangle$.

- If $m_1 m_2 = 01$, Bob has to apply a X gate, because $X\left(a|1\rangle + b|0\rangle\right) = a|0\rangle + b|1\rangle = |\psi\rangle$.
- If $m_1 m_2 = 10$, Bob has to apply a Z gate, because $Z\left(a|0\rangle - b|1\rangle\right) = a|0\rangle + b|1\rangle = |\psi\rangle$.
- If $m_1 m_2 = 11$, Bob has to apply first a X gate, then a Z gate, because $ZX\left(a|1\rangle - b|0\rangle\right) = Z\left(a|0\rangle - b|1\rangle\right) = a|0\rangle + b|1\rangle = |\psi\rangle$.

Or, shortly: if $m_1 m_2$, Bob has to apply first X^{m_2}, then Z^{m_1}.

Note that this protocol doesn't go against the special relativity principles because Bob can reproduce state $|\psi\rangle$ only after he receives the two bits from Alice's measurement, so the information travels only with a limited speed.

Bob can't reconstitute by himself the state $|\psi\rangle$. To prove this, consider the density operator of the whole system, just after the Alice's measurement:

$$\rho^{AB} = \frac{1}{4}|\psi_5(00)\rangle\langle\psi_5(00)| + \frac{1}{4}|\psi_5(01)\rangle\langle\psi_5(01)| + \frac{1}{4}|\psi_5(10)\rangle\langle\psi_5(10)| + \frac{1}{4}|\psi_5(11)\rangle\langle\psi_5(11)|$$

$$\rho^{AB} = \frac{1}{4}\Big[|00\rangle\langle 00|\left(a|0\rangle + b|1\rangle\right)\left(a^*\langle 0| + b^*\langle 1|\right) + |01\rangle\langle 01|\left(a|1\rangle + b|0\rangle\right)\left(a^*\langle 1| + b^*\langle 0|\right)$$
$$+ |10\rangle\langle 10|\left(a|0\rangle - b|1\rangle\right)\left(a^*\langle 0| - b^*\langle 1|\right) + |11\rangle\langle 11|\left(a|1\rangle - b|0\rangle\right)\left(a^*\langle 1| - b^*\langle 0|\right)\Big]$$

Apply a partial trace on this system to find the density matrix for the Bob's sub-system:

$$\rho^B \equiv tr_A\left(\rho^{AB}\right) = \frac{1}{4}\Big[\left(a|0\rangle + b|1\rangle\right)\left(a^*\langle 0| + b^*\langle 1|\right) + \left(a|1\rangle + b|0\rangle\right)\left(a^*\langle 1| + b^*\langle 0|\right)$$
$$+ \left(a|0\rangle - b|1\rangle\right)\left(a^*\langle 0| - b^*\langle 1|\right) + \left(a|1\rangle - b|0\rangle\right)\left(a^*\langle 1| - b^*\langle 0|\right)\Big]$$
$$= \frac{2\left(|a|^2 + |b|^2\right)0\rangle\langle 0| + 2\left(|a|^2 + |b|^2\right)1\rangle\langle 1|}{4}$$
$$= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} = \frac{I_2}{2}$$

So, the state of Bob's sub-system doesn't depend at all on the target qubit state $|\psi\rangle$.

Therefore, any measurement Bob would perform at this point will provide no information about $|\psi\rangle$.

Another fact to be noted is that the quantum teleportation protocol doesn't produce any copy of the target qubit. At any given moment, only one qubit is in state $|\psi\rangle$. After the protocol ends the target qubit, which was initially in state $|\psi\rangle$, is now in one of the computational basis states $|0\rangle$ or $|1\rangle$, because a measurement is applied on it.

The quantum teleportation shows the possibility to inter-change of different resources, demonstrating that one EPR pair together with two classical bits of communication is at least identical to a qubit of communication. This fact is used for building robust to noise quantum gates and for correcting the errors in quantum transmissions.

3. Graphical representations

3.1. Trace of an operator

Every operator A over the Hilbert vector space V of dimension n has an associated square matrix of complex numbers $a_{ij}[n \times n]$, defined for an orthonormal base $|v_1\rangle..|v_n\rangle$ like this [10]:

$$a_{ij} = \langle v_i | A | v_j \rangle, \forall i, j = \overline{1,n}.$$

For two different orthonormal bases $|v_1\rangle..|v_n\rangle$ and $|w_1\rangle..|w_n\rangle$, the associated matrices are $a_{ij}^v = \langle v_i | A | v_j \rangle$ and respectively $a_{ij}^w = \langle w_i | A | w_j \rangle$. These matrices are similar, being related by a unitary matrix $A^w = U^\dagger A^v U$ where $u_{ij} = \langle v_i | w_j \rangle$.

The trace of a square matrix of order n is defined as:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}$$

The trace is cyclic:

$$\mathrm{tr}(AB) = \sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} b_{ki} = \sum_{k=1}^{n}\sum_{i=1}^{n} b_{ki} a_{ik} = \mathrm{tr}(BA).$$

The trace is linear:

$$\mathrm{tr}(A+B) = \sum_{i=1}^{n} a_{ii} + b_{ii} = \sum_{i=1}^{n} a_{ii} + \sum_{i=1}^{n} b_{ii} = \mathrm{tr}(A) + \mathrm{tr}(B); \mathrm{tr}(cA) = \sum_{i=1}^{n} c a_{ii} = c\sum_{i=1}^{n} a_{ii} = c\,\mathrm{tr}(A)$$

and:

$$\mathrm{tr}(A^\dagger) = \sum_{i=1}^{n} a_{ii}^* = \left(\sum_{i=1}^{n} a_{ii}\right)^* = \mathrm{tr}(A)^*$$

Because trace is an invariant of the similarity transformation

$$\mathrm{tr}(U^\dagger A U) = \mathrm{tr}(U U^\dagger A) = \mathrm{tr}(A),$$

And because changing the base is equivalent to a similarity transformation [50], it means it makes sense to define the trace of an operator over the vector space V as the trace of the matrix associated to an orthonormal base.

If $|j\rangle, j = \overline{1,n}$ is an orthonormal base in V, then:

$$\forall j,k = \overline{1,n} \text{ şi } j \neq k: \ \mathrm{tr}(|j\rangle\langle j|) = 1 \text{ and } \mathrm{tr}(|j\rangle\langle k|) = 0 \text{ and } \mathrm{tr}(I_n) = n$$

3.2. The operators' vector space

The set L_V, containing all the linear operators defined over a Hilbert space V, is also a vector space over the complex numbers [33]. Also, in this space one can define a scalar product: $(A,B) \overset{\Delta}{=} \mathrm{tr}(A^\dagger B)$. An orthonormal basis in this space is defined by the Hermite operators:

$$A_{jk}=\begin{cases}\dfrac{|k\rangle\langle j|+|j\rangle\langle k|}{\sqrt{2}}, & \forall j,k=\overline{1,n} \text{ and } k>j\\ \dfrac{i|j\rangle\langle k|-i|k\rangle\langle j|}{\sqrt{2}}, & \forall j,k=\overline{1,n} \text{ and } j>k\\ |j\rangle\langle j|, & \forall j=\overline{1,n}\end{cases}$$

3.3. Pauli matrices

With the above notations, if V_2 is a bi-dimensional vector space, the vectors corresponding to the computational basis states form an orthonormal basis, and the following Hermite operators:

$$B_{00}=\frac{|0\rangle\langle 0|+|1\rangle\langle 1|}{\sqrt{2}} \quad B_{01}=\frac{|1\rangle\langle 0|+|0\rangle\langle 1|}{\sqrt{2}} \quad B_{10}=\frac{i|1\rangle\langle 0|-i|0\rangle\langle 1|}{\sqrt{2}} \quad B_{11}=\frac{|0\rangle\langle 0|-|1\rangle\langle 1|}{\sqrt{2}}$$

form an orthonormal basis in L_{V_2}.

Removing the requirement for the operators to be normal, one obtains the Pauli operators:

$$\sigma_0\equiv I_2\equiv|0\rangle\langle 0|+|1\rangle\langle 1|$$
$$\sigma_1\equiv\sigma_x\equiv X\equiv|1\rangle\langle 0|+|0\rangle\langle 1|$$
$$\sigma_2\equiv\sigma_y\equiv Y\equiv i|1\rangle\langle 0|-i|0\rangle\langle 1|$$
$$\sigma_3\equiv\sigma_z\equiv Z\equiv|0\rangle\langle 0|-|1\rangle\langle 1|$$

The Pauli matrices associated to Pauli operators for the computational basis states are:

$$\sigma_0\equiv I_2\equiv\begin{bmatrix}1&0\\0&1\end{bmatrix} \quad \sigma_1\equiv\sigma_x\equiv X\equiv\begin{bmatrix}0&1\\1&0\end{bmatrix} \quad \sigma_2\equiv\sigma_y\equiv Y\equiv\begin{bmatrix}0&-i\\i&0\end{bmatrix} \quad \sigma_3\equiv\sigma_z\equiv Z\equiv\begin{bmatrix}1&0\\0&-1\end{bmatrix}$$

Pauli matrices have the following properties:

- Hermite: $\sigma_k^\dagger=\sigma_k, \forall k=\overline{0,3}$
- unitary: $\sigma_k^\dagger\sigma_k=\sigma_k\sigma_k^\dagger=\sigma_k^2=I_2, \forall k=\overline{0,3}$
- if $\forall j,k=\overline{1,3}: \sigma_j\sigma_k=\delta_{jk}I_2+i\sum_{l=1}^{3}\varepsilon_{jkl}\sigma_l$
- $XY=iZ \quad YZ=iX \quad ZX=iY \quad YX=-iZ \quad ZY=-iX \quad XZ=-iY$
- $\{\sigma_j,\sigma_k\}=\sigma_j\sigma_k+\sigma_k\sigma_j=0_2, \quad \forall j,k=\overline{1,3}$ şi $j\neq k$
- $[\sigma_j,\sigma_k]=2i\sum_{l=1}^{3}\varepsilon_{jkl}\sigma_l$
- $\mathrm{tr}(\sigma_0)=2$ and $\mathrm{tr}(\sigma_k)=0, \ \forall k=\overline{1,3}$

3.4. Graphical representation of qubits

3.4.1. Qubits in pure state – Bloch sphere

Using the well-established Dirac notation, the pure state of a qubit, defined as a linear superposition of the computational basis states, it is represented by the following equation:

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle, \qquad \alpha,\beta\in\mathbf{C}, \quad |\alpha|^2+|\beta|^2=1$$

where $|0\rangle$ and $|1\rangle$ are the computational basis states. So, a qubit in a pure state is represented by a unit vector in a bi-dimensional complex vector space [48].
For the purpose of geometrical representation, it is more useful to use the polar coordinates for complex numbers. Considering also the measuring principle from quantum mechanics stating that measuring two quantum states that differ only by a global phase factor, provides always the same result ($|\psi\rangle \cong e^{i\phi}|\psi\rangle,\ \forall\phi \in \mathbf{R}$), the state of the qubit can be then expressed as:

$$|\psi\rangle = \cos\frac{\gamma}{2}|0\rangle + \sin\frac{\gamma}{2}e^{i\varphi}|1\rangle \qquad \gamma \in [0,\pi],\ \ \varphi \in [0,2\pi)$$

This equation represents therefore the base starting point that provides for a geometrical representation of qubit states, because it proves that there is a bijective mapping between the set of measurable pure states of a qubit and the unit sphere in the Euclidean tridimensional space. According to this mapping, each and every pure qubit state has an associated point P on the unit sphere, a point having the spherical coordinates $P(\varphi,\gamma)$. The $\overrightarrow{OP}$ vector, with the origin in the center of the unit sphere, is called "*Bloch vector*" and has the following coordinates in a tri-dimensional Euclidean space:

$$(p_x, p_y, p_z) = (\cos\varphi\sin\gamma, \sin\varphi\sin\gamma, \cos\gamma)$$

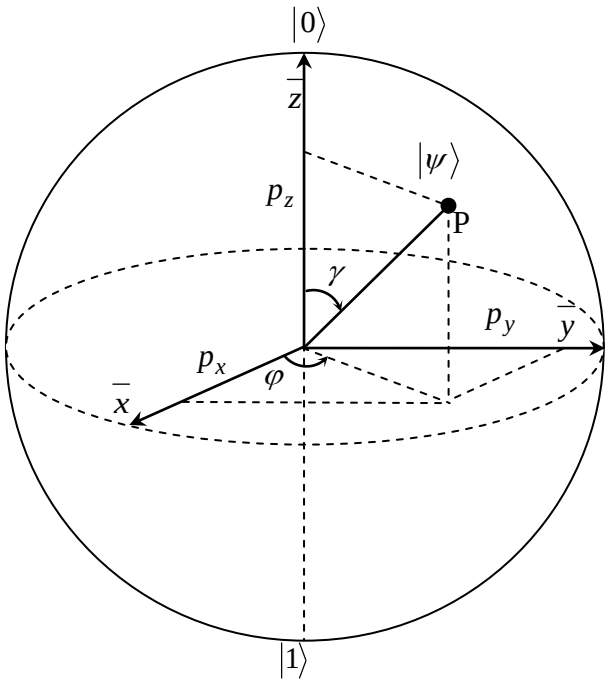


Figure 18. Bloch Sphere

3.4.2. Qubits in mixed states – Bloch ball

As opposed to pure state qubits, where a state vector is used to represent their state, when the qubit state is mixed, the density operator is used for representing that state. Actually, these two representations are mathematically equivalent, but they have different, though similar, physical interpretations. The principles that form the base for the quantum mechanics theory can be formulated using either of the two approaches: state vectors or density operators. Assuming a quantum system is in an unknown state, but the possible states form a finite and discrete set, where the pure state $|\psi_i\rangle$ occurs with the probability p_i , the density operator associated with that system is defined by the following equation:

$$\rho \equiv \sum_i p_i |\psi_i\rangle\langle\psi_i|, \quad 0 \le p_i \le 1, \quad \sum_i p_i = 1$$

The density operator is a self-adjoint (Hermite) positive operator, with unit norm:

$$\rho^\dagger \equiv \sum_i p_i^* |\psi_i\rangle\langle\psi_i| = \sum_i p_i |\psi_i\rangle\langle\psi_i| = \rho$$

$$\langle\psi|\rho|\psi\rangle = \sum_i p_i \langle\psi|\psi_i\rangle\langle\psi_i|\psi\rangle = \sum_i p_i \langle\psi|\psi_i\rangle\langle\psi|\psi_i\rangle^* = \sum_i p_i |\langle\psi|\psi_i\rangle|^2 \ge 0$$

$$\mathrm{tr}(\rho) = \mathrm{tr}\left(\sum_i p_i |\psi_i\rangle\langle\psi_i|\right) = \sum_i p_i \,\mathrm{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1$$

According to the quantum mechanics formalism based on density operators, the pure states are just a particular case that can be also represented by density operators. The density operator of a pure state is defined as:

$$\rho \equiv |\psi\rangle\langle\psi|$$

The trace of a general density operator satisfies the following inequality; with the equality happening if and only if the respective density operator represents a pure state:

$$\mathrm{tr}(\rho^2) \le 1$$

For the quantum system representing one qubit, the associated density operator ρ belongs to the complex vector space defined by the set of all operators generated by the Pauli operators. Therefore that density operator can be decomposed as a linear superposition of the Pauli operators, using complex coefficients:

$$\rho = aI_2 + b\sigma_x + c\sigma_y + d\sigma_z, \qquad a,b,c,d \in \mathbf{C}$$

The complex coefficients can be further refined by imposing the restrictions that apply to density operators: self-adjoint, positive and unitary. Hence the equation above can be transformed such that it allows for a geometrical representation:

$$\rho = \frac{1}{2}\left(I_2 + r_x\sigma_x + r_y\sigma_y + r_z\sigma_z\right) = \frac{1}{2}\left(I_2 + \bar{r}\cdot\bar{\sigma}\right)$$

whereas $\bar{r}$ is a vector in the real tri-dimensional Euclidian space. Considering the matrix associated with this density operator in the computational basis state, and by forcing its eigenvalues to be real positive numbers (this is allowed because the operator is itself positive), it follows that the $\bar{r}$ vector is restricted to a unitary ball, centered in the origin:

$$r_x^2 + r_y^2 + r_z^2 \le 1 \Leftrightarrow \|\bar{r}\| \le 1$$

Again, the equality happens if and only if the respective density operator represents a pure state. The north and south poles of the ball represent two particular pure states: the computational basis states.

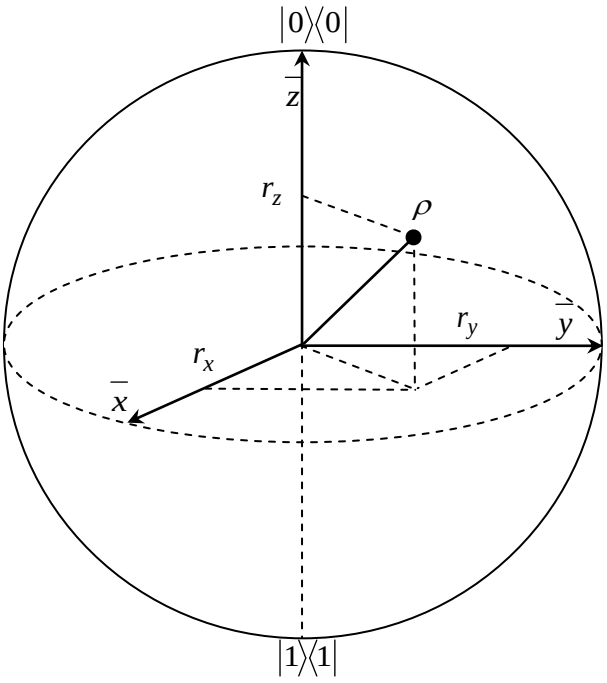


Figure 19. Bloch ball

When the density operator ρ represents a pure state, this state is represented using the real unity vector $\bar{r} = (r_x, r_y, r_z) = (\cos\varphi \sin\gamma, \sin\varphi \sin\gamma, \cos\gamma)$. Then, the representation on the Bloch sphere is a particular case of the representation on the Bloch ball.

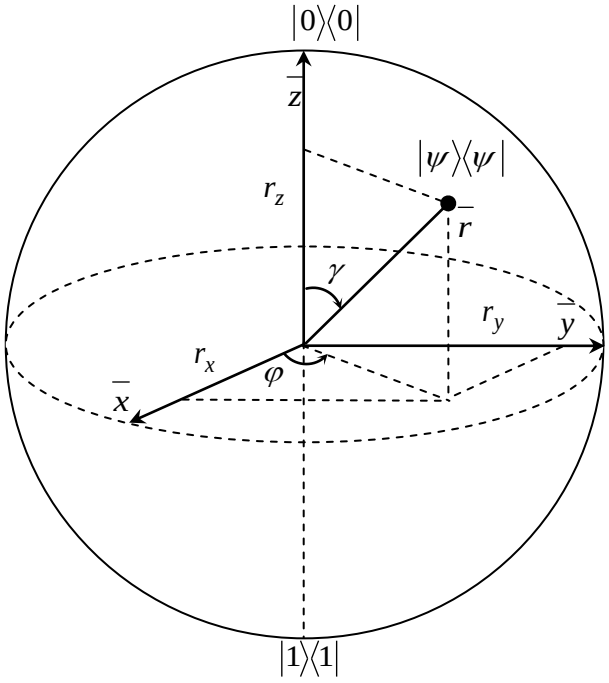


Figure 20. Bloch sphere and ball

3.5. Rotation operators

The complex exponential function defined on the space of the square matrices $n \times n$ can be factored using the Taylor-Maclaurin series [13]:

$$\exp(iAx) = \sum_{k=0}^{\infty} \frac{(iAx)^k}{k!}$$

where A is any square matrix of complex numbers and x is any real number. If additionally, A satisfies the relations:

$$A^2 = I_n \Rightarrow A^{2k} = I_n; A^{2k+1} = A, \forall k \in N,$$

the above equation becomes:

$$\exp(iAx) = \left(\sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k)!} x^{2k} \right) I_n + i \left(\sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k+1)!} x^{2k+1} \right) A = \cos(x) I_n + i \sin(x) A$$

Therefore, because the Pauli matrices satisfy $\sigma_k^2 = I_2$, one can define the following operators in the complex bi-dimensional space, which will be proved to be the very tri-dimensional rotations across the coordinate's axes:

$$R_k(\theta) \equiv \exp\left(\frac{-i\theta\sigma_k}{2} \right) = \cos\frac{\theta}{2} I_2 - i \sin\frac{\theta}{2} \sigma_k, \forall k = \overline{0,3}$$

3.5.1. Rotation operator R_z

For the k = 3case in equation (16), the operator and its associated matrix, according to the computational basis state, defined by the following equation:

$$R_z(\theta) \equiv \exp\left(\frac{-i\theta Z}{2} \right) = \cos\frac{\theta}{2} I_2 - i \sin\frac{\theta}{2} Z = \begin{bmatrix} e^{\frac{-i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{bmatrix}$$

has the following geometrical interpretation: for a qubit in a pure state, upon which an operation defined by equation above is performed such that the qubit transformed state is:

$$|\psi_1\rangle = R_z(\theta)|\psi_0\rangle = \cos\frac{\gamma_1}{2}|0\rangle + \sin\frac{\gamma_1}{2} e^{i\varphi_1}|1\rangle$$

and if the corresponding points on the Bloch sphere are P_0 and P_1, then P_1 can be deducted geometrically by rotating P_0 with angle θ around the z axis.

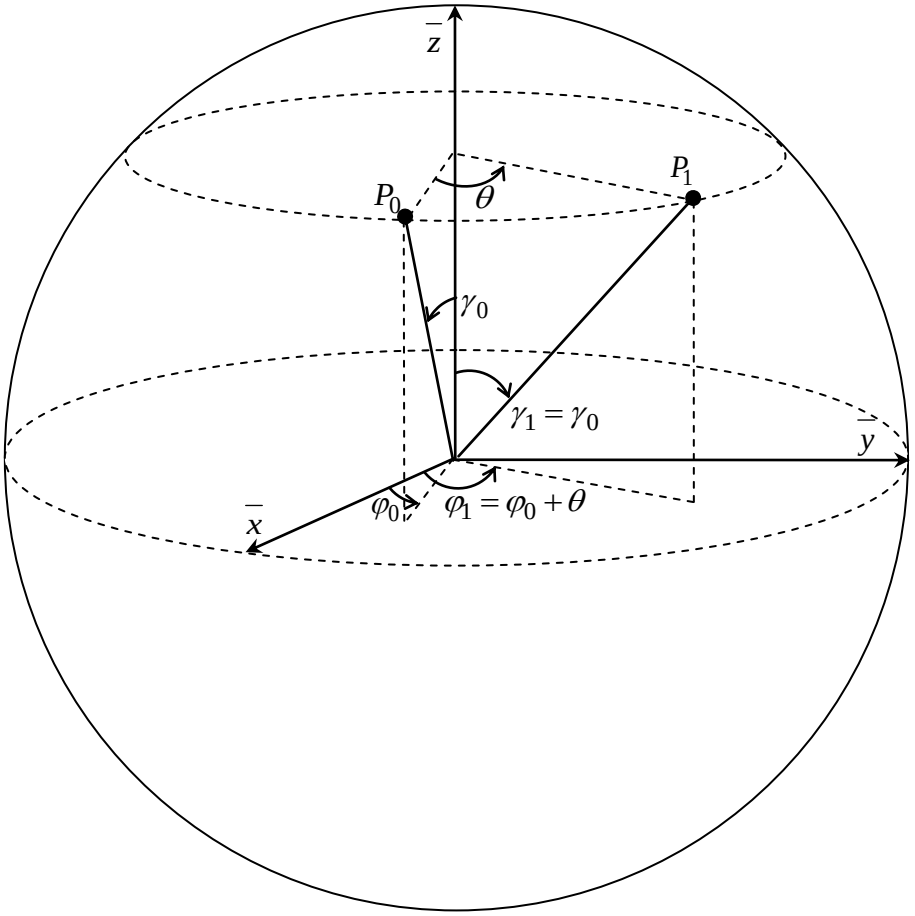


Figure 212. Geometrical representation of the rotation operator R_z

This interpretation can be validated by simply applying the operator on the qubit in initial state:

$$R_z(\theta)|\psi_0\rangle = \begin{bmatrix} e^{\frac{-i\theta}{2}} & 0 \\ 0 & e^{\frac{i\theta}{2}} \end{bmatrix} \begin{bmatrix} \cos\frac{\gamma_0}{2} \\ \sin\frac{\gamma_0}{2} e^{i\varphi_0} \end{bmatrix} = e^{\frac{-i\theta}{2}} \begin{bmatrix} \cos\frac{\gamma_0}{2} \\ \sin\frac{\gamma_0}{2} e^{i(\varphi_0+\theta)} \end{bmatrix} = e^{\frac{-i\theta}{2}} |\psi_1\rangle \cong |\psi_1\rangle$$

And from the figure above, the parameters for the transformed state:

$$P_0 \to P_1 \Leftrightarrow \gamma_1 = \gamma_0 \text{ şi } \varphi_1 = \varphi_0 + \theta$$

3.5.2. Rotation operator R_x

In the same way, for the case $k = 1 = x$, the operator whose associated matrix in the computational basis state is:

$$R_x(\theta) \equiv \exp\left(\frac{-i\theta X}{2}\right) = \cos\frac{\theta}{2} I_2 - i\sin\frac{\theta}{2} X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

has the following geometrical interpretation: for a qubit in a pure state, upon which an operation defined by equation above is performed such that the qubit transformed state is

$$|\psi_1\rangle = R_x(\theta)|\psi_0\rangle = \cos\frac{\gamma_1}{2}|0\rangle + \sin\frac{\gamma_1}{2} e^{i\varphi_1}|1\rangle,$$

and if the corresponding points on the Bloch sphere are P_0 and P_1, then P_1 can be deducted geometrically by rotating P_0 with angle θ around the x axis.

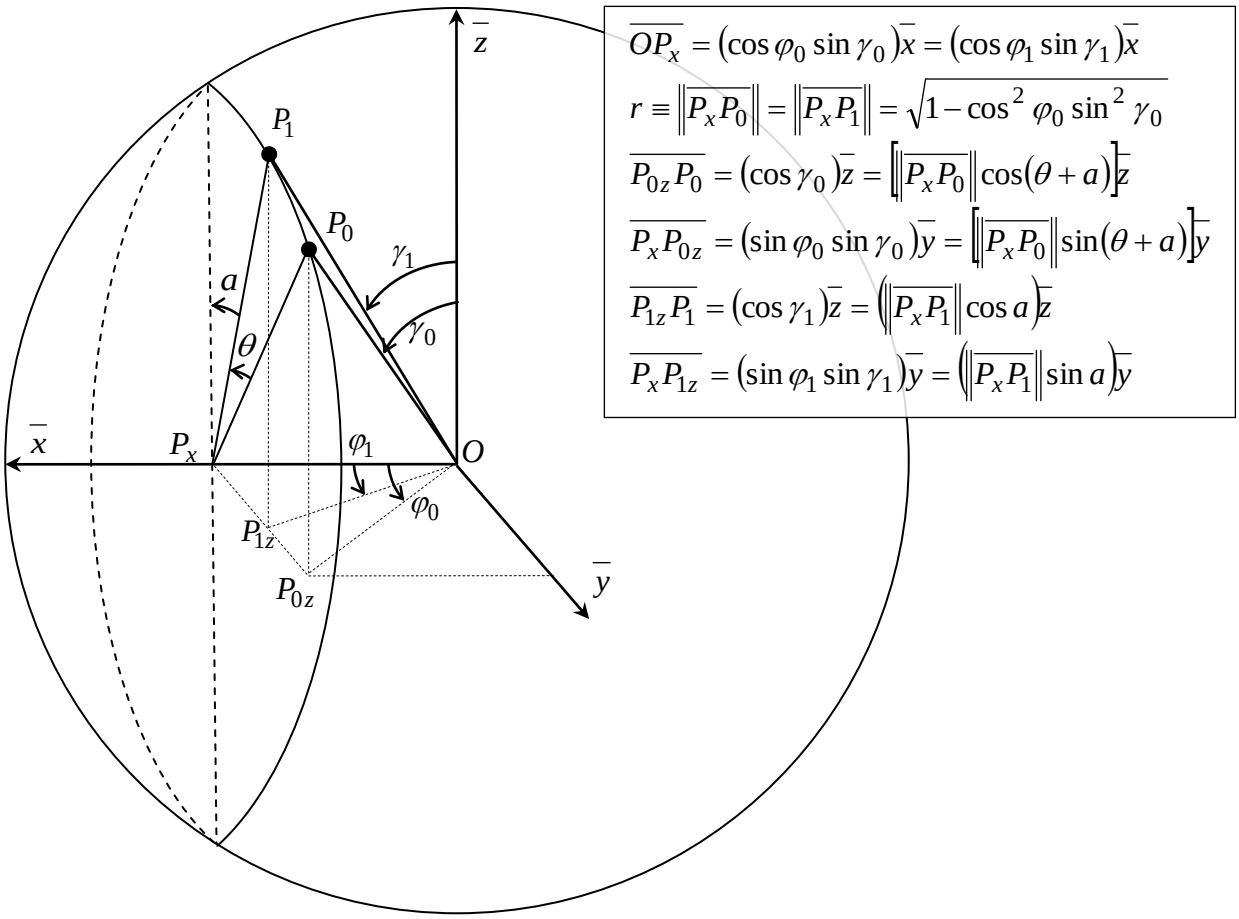


Figure 22. Geometrical representation of the rotation operator R_x

This interpretation can be validated by:

$$R_x(\theta)|\psi_0\rangle = \begin{bmatrix} \cos\frac{\theta}{2}\cos\frac{\gamma_0}{2} - i\sin\frac{\theta}{2}\sin\frac{\gamma_0}{2}e^{i\varphi_0} \\ -i\sin\frac{\theta}{2}\cos\frac{\gamma_0}{2} + \cos\frac{\theta}{2}\sin\frac{\gamma_0}{2}e^{i\varphi_0} \end{bmatrix} = e^{i\phi}\begin{bmatrix} \cos\frac{\gamma_1}{2} \\ \sin\frac{\gamma_1}{2}e^{i\varphi_1} \end{bmatrix} = e^{i\phi}|\psi_1\rangle \cong |\psi_1\rangle, \text{ unde } \phi \in [0,2\pi)$$

3.5.3. Rotation operator R_y

In the same way, for the case $k = 1 = y$, the operator whose associated matrix in the computational basis state is:

$$R_y(\theta) \equiv \exp\left(\frac{-i\theta Y}{2}\right) = \cos\frac{\theta}{2}I_2 - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

has the following geometrical interpretation: for a qubit in a pure state, upon which an operation defined by equation above is performed such that the qubit transformed state is:

$$|\psi_1\rangle = R_y(\theta)|\psi_0\rangle = \cos\frac{\gamma_1}{2}|0\rangle + \sin\frac{\gamma_1}{2}e^{i\varphi_1}|1\rangle,$$

and if the corresponding points on the Bloch sphere are P_0 and P_1, then P_1 can be deducted geometrically by rotating P_0 with angle θ around the *y* axis.

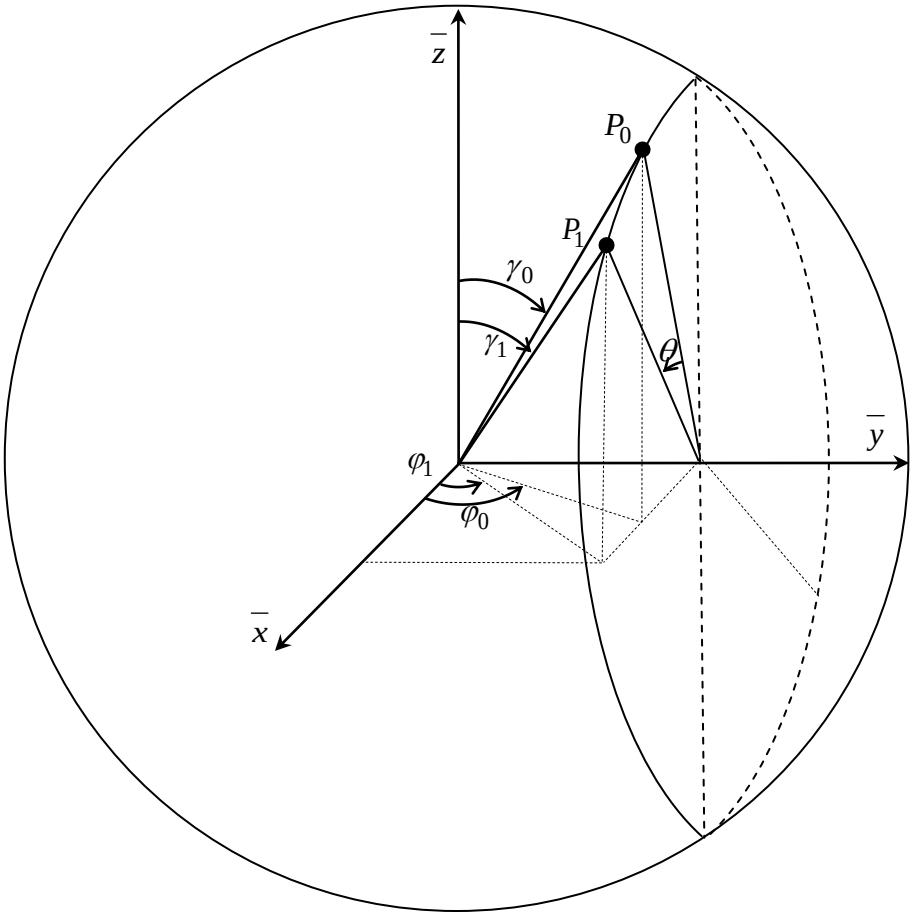


Figure 23. Geometrical representation of the rotation operator R_y

For validating this last geometrical representation, one can make use of the two representations demonstrated in the previous two chapters, for R_x and R_z. Any rotation with angle θ around the *y* axis can be factored out into three rotations made in the following order:

- one rotation with angle $-\frac{\pi}{2}$ around the $\bar{z}$ axis
- one rotation with angle θ around the $\bar{x}$ axis
- one rotation with angle $\frac{\pi}{2}$ around the $\bar{z}$ axis

Then, according to the previous two geometrical representations, these three rotations can be written using the respective operators:

$$R_z\left(\frac{\pi}{2}\right)R_x(\theta)R_z\left(-\frac{\pi}{2}\right)=\begin{bmatrix} e^{-i\frac{\pi}{4}} & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}\begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}\begin{bmatrix} e^{i\frac{\pi}{4}} & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{bmatrix}=R_y(\theta)$$

3.5.4. Generic rotation operator R_n

Considering a unitary vector $\bar{n} = (n_x, n_y, n_z)$ in the real tridimensional space, there is a matrix defined by:

$$\bar{n} \cdot \bar{\sigma} \equiv n_x \sigma_x + n_y \sigma_y + n_z \sigma_z \text{ where } (\bar{n} \cdot \bar{\sigma})^2 = I_2$$

And therefore it makes sense to define the generic rotation operator:

$$R_n(\theta) \equiv \exp\left(-i\frac{\theta}{2}\bar{n} \cdot \bar{\sigma}\right) = \cos\frac{\theta}{2} I_2 - i \sin\frac{\theta}{2} \bar{n} \cdot \bar{\sigma}$$

having the following geometrical interpretation: for a qubit transformed by the operator $R_n(\theta)$, such that

$$|\psi_1\rangle = R_n(\theta)|\psi_0\rangle = \cos\frac{\gamma_1}{2}|0\rangle + \sin\frac{\gamma_1}{2} e^{i\varphi_1}|1\rangle,$$

if P_0 and P_1 are points on the Bloch sphere corresponding to the states $|\psi_0\rangle$ and respectively $|\psi_1\rangle$, then P_1 can be obtained by rotating P_0, by the angle θ around the $\bar{n}$ axis.

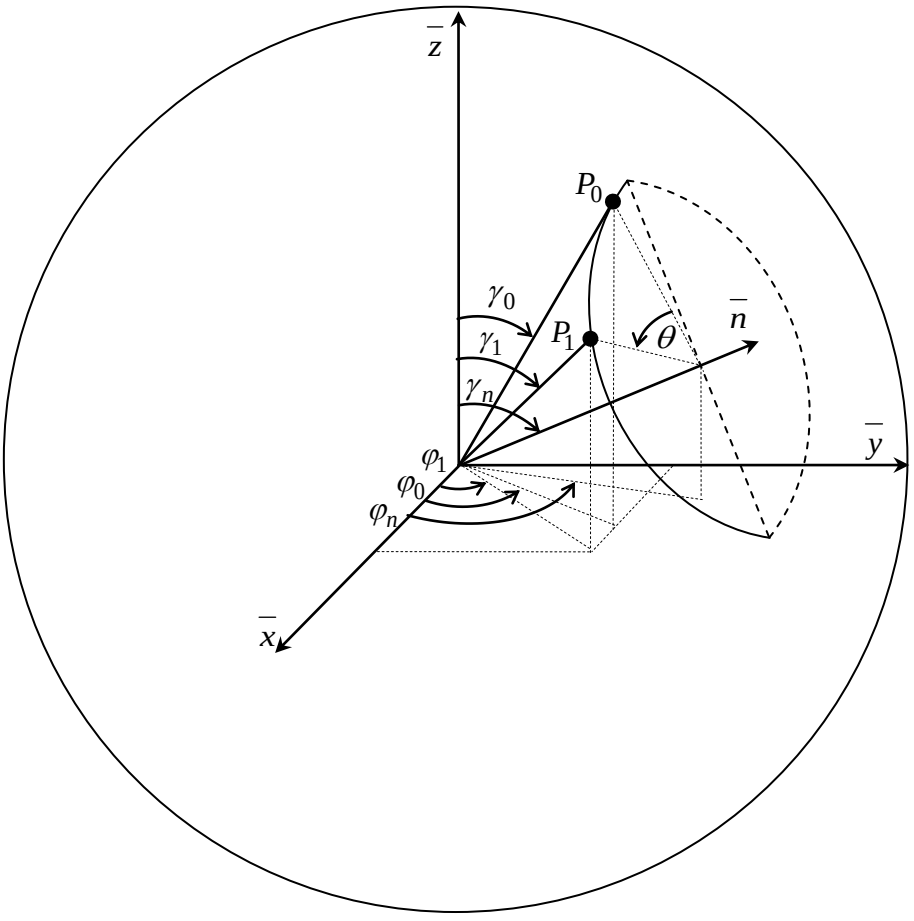


Figure 24. Geometrical representation of the generic rotation operator R_n

For demonstrating this, the rotation by angle θ around the $\bar{n}$ axis is decomposed in rotations around the coordinate axes:

- align $\bar{n}$ axis over $\bar{z}$:
  - rotation by angle $-\varphi_n$ around $\bar{z}$

- rotation by angle $-\gamma_n$ around $\bar{y}$

- perform the desired rotation:
  - rotation by angle θ around $\bar{z}$
- bring $\bar{n}$ axis back in the original position:
  - rotation by angle γ_n around $\bar{y}$
  - rotation by angle φ_n around $\bar{z}$

$$R_n(\theta) = R_z(\varphi_n)R_y(\gamma_n)R_z(\theta)R_y(-\gamma_n)R_z(-\varphi_n)$$

$$= \begin{bmatrix} \cos\frac{\theta}{2} - i\sin\frac{\theta}{2}\cos\gamma_n & \sin\frac{\theta}{2}\left(-i\cos\varphi_n\sin\gamma_n - \sin\varphi_n\sin\gamma_n\right) \\ \sin\frac{\theta}{2}\left(-i\cos\varphi_n\sin\gamma_n + \sin\varphi_n\sin\gamma_n\right) & \cos\frac{\theta}{2} + i\sin\frac{\theta}{2}\cos\gamma_n \end{bmatrix}$$

The coordinates of vector $\bar{n}$ can be deducted from the angles φ_n and γ_n:

$$\bar{n} = (n_x, n_y, n_z) = (\cos\varphi_n\sin\gamma_n, \sin\varphi_n\sin\gamma_n, \cos\gamma_n)$$

It implies that:

$$R_n(\theta) = \left(n_x\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} + n_y\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} + n_z\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\right) = \cos\frac{\theta}{2}I_2 - i\sin\frac{\theta}{2}\left(n_xX + n_yY + n_zZ\right)$$

3.6. Decomposing unitary operators on one qubit

Because the set of Pauli operators I_2, X, Y, Z form an orthogonal basis in the vector space of linear operators over the bi-dimensional complex space, any operator on one qubit U can be decomposed in:

$$U = aI_2 + bX + cY + dZ$$

where a, b, c, d are complex numbers uniquely identified.

If, additionally, the U operator has to be unitary, then: $U = e^{i\alpha}R_n(\theta)$, where $\bar{n} = (n_x, n_y, n_z)$ is a unit vector and $\theta \in [0, \pi]$.

3.6.1. Z-Y decomposition of one qubit unitary operators

If U is a unitary operator over the bi-dimensional complex space, whose associated matrix is:

$$U = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow U^\dagger = \begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix}$$

where a, b c and d are complex numbers. Then, using the polar coordinates of these complex numbers, the matrix can be decomposed like this:

- if $|b| = 0 \vee |c| = 0 \Rightarrow |a| = |d| = 1$, then $U = e^{i\frac{\alpha_a + \alpha_d}{2}}R_z(-\alpha_a)R_y(0)R_z(\alpha_d)$
- if $|a| = 0 \vee |d| = 0 \Rightarrow |b| = |c| = 1$, then $U = e^{i\frac{\alpha_b + \alpha_c + \pi}{2}}R_z(-\alpha_b + \alpha_c - \pi)R_y(\pi)R_z(0)$
- if $|a| \neq 0$ and $|b| \neq 0$ and $|c| \neq 0$ and $|d| \neq 0$, then

$$U = e^{i\left(\frac{\alpha_a + \alpha_d}{2}\right)}R_z(-\alpha_a + \alpha_c)R_y(2\arccos|a|)R_z(-\alpha_c + \alpha_d)$$

3.6.2. X-Y decomposition of unitary operators on one qubit

Considering U a unitary linear operator over the bi-dimensional complex space, it can be decomposed in rotations X-Y: $U = \mathrm{e}^{i\alpha}\, R_x(\beta)R_y(\gamma)R_x(\delta)$ where the rotation angles can be computed from:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \mathrm{e}^{i\alpha} \begin{bmatrix} \cos\frac{\gamma}{2}\cos\frac{\beta+\delta}{2} - i\sin\frac{\gamma}{2}\sin\frac{\beta-\delta}{2} & -\sin\frac{\gamma}{2}\cos\frac{\beta-\delta}{2} - i\cos\frac{\gamma}{2}\sin\frac{\beta+\delta}{2} \\ \sin\frac{\gamma}{2}\cos\frac{\beta-\delta}{2} - i\cos\frac{\gamma}{2}\sin\frac{\beta+\delta}{2} & \cos\frac{\gamma}{2}\cos\frac{\beta+\delta}{2} + i\sin\frac{\gamma}{2}\sin\frac{\beta-\delta}{2} \end{bmatrix}$$

4. Controlled quantum circuits

Using the decomposition of unitary operators on one qubit in a product of rotations, it is possible to demonstrate the following result, very important in constructing unitary operators that act on multiple qubits.

Corollary: Suppose U is a unitary operator on one qubit. Then there exist unitary operators A, B, C on one single qubit such that $ABC = I_2$ and $U = e^{i\alpha}AXBXC$ where α is a generic phase factor.

Demonstration: According to the $Z - Y$ decomposition on one qubit, any unitary operator U can be decomposed using $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ as:

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

The corollary can then be proved by using the unitary operators:

$$A \stackrel{\text{def}}{=} R_z(\beta) R_y\left(\frac{\gamma}{2}\right)$$

$$B \stackrel{\text{def}}{=} R_y\left(-\frac{\gamma}{2}\right) R_z\left(-\frac{\delta+\beta}{2}\right)$$

$$C \stackrel{\text{def}}{=} R_z\left(\frac{\delta-\beta}{2}\right)$$

4.1. Controlled-U operator on one qubit

4.1.1. Definition and notations

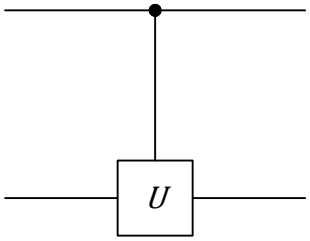
Let a unitary operator on one qubit U. A *Controlled-U* operator is defined as an operator on two qubits – one control qubit and one data (or target) qubit – that obeys the following rules:

- the control qubit always remains unchanged,
- if the control qubit is set, then operator U acts on the data qubit,
- if the control qubit is reset, the target qubit remains unchanged.

$$|0\rangle|0\rangle \longrightarrow |0\rangle|0\rangle$$
$$|0\rangle|1\rangle \longrightarrow |0\rangle|1\rangle$$
$$|1\rangle|0\rangle \longrightarrow |1\rangle U|0\rangle$$
$$|1\rangle|1\rangle \longrightarrow |1\rangle U|1\rangle$$

$$|c\rangle|d\rangle \xrightarrow{U-Controlat} |c\rangle U^c|d\rangle$$

The circuit that implements the $Controlled - U$ operator is represented as [61]:



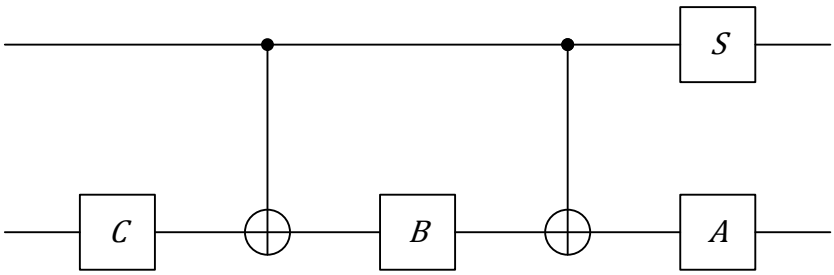
4.1.2. Implementing the Controlled-U operator on one qubit

Using the decomposition $U = e^{i\alpha}AXBXC$, it can be proved that:

Theorem: The $Controlled - U$ circuit can be implemented using only one qubit gates and the CNOT gate.

Demonstration: The circuit is the following one, where the phase gate is defined as:

$$|0\rangle \xrightarrow{S} S|0\rangle = |0\rangle$$
$$|1\rangle \xrightarrow{S} S|1\rangle = e^{i\alpha}|1\rangle$$



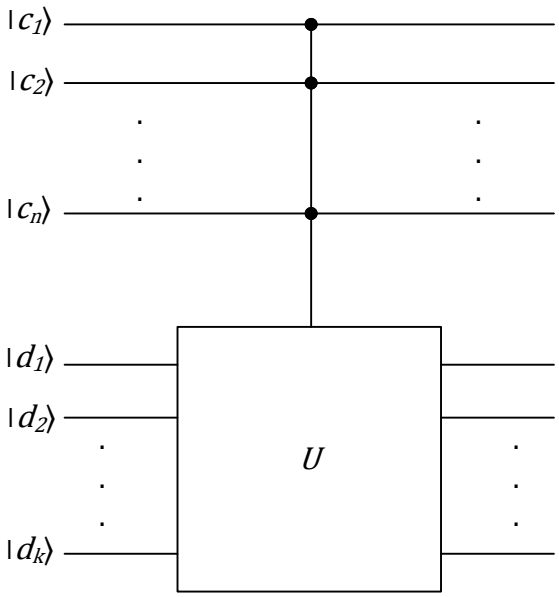
4.2. The Controlled-U operator on multiple qubits

A particular example of such operator is implemented by the Toffoli gate.
In the general case, considering the computational basis state, given an operator on $n + k$ qubits where the U operator acts on k qubits, the controlled U operator by n qubits is defined as:

$$C_k^n(U)|c_1c_2 \dots c_n\rangle|d_1d_2 \dots d_k\rangle \stackrel{\text{def}}{=} |c_1c_2 \dots c_n\rangle U^{c_1c_2\dots c_n}|d_1d_2 \dots d_k\rangle,$$

where the exponent is the binary product of the control bits $c_1c_2 \dots c_n$. So, the controlled U operator has the following properties:

- the control qubits $c_1c_2 \dots c_n$ always remain unchanged,
- if all the control qubits are set $\forall i \in \{1,2 \dots n\} \Rightarrow c_i = 1$, then the U operator acts on the data qubits $d_1d_2 \dots d_k$,
- if at least one control qubit is reset $\exists i \in \{1,2 \dots n\} \Rightarrow c_i = 0$, then all the data qubits remain unchanged.

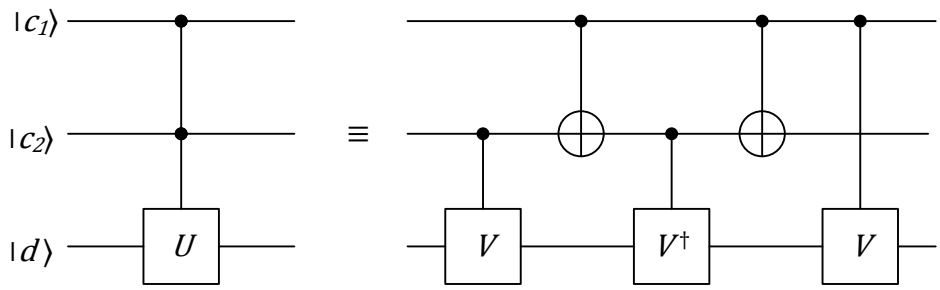


4.3. The Controlled-U operator with two control qubits

4.3.1. Implementation using controlled gates on 1 qubit

Theorem: $C_1^2(U)$ can be decomposed in a product of $C_1^1(U)$ operators.

Demonstration: The following circuit demonstrates this decomposition, where $V^2 = U$.



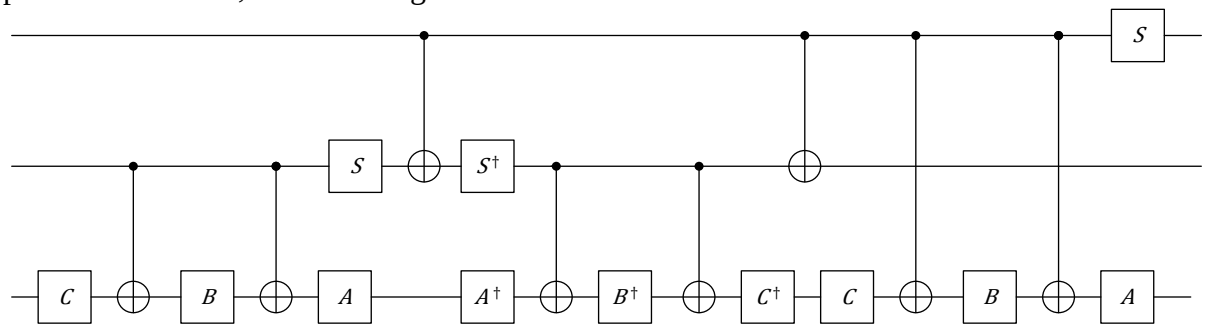
■

For the Toffoli gate: $U \equiv X$ and $V \equiv \frac{1-i}{2}(I_2 + iX)$. Then it follows that reversible gates on one qubit together with reversible gates on two qubits are enough to implement the Toffoli gate. This proposition doesn't hold for classical computing.

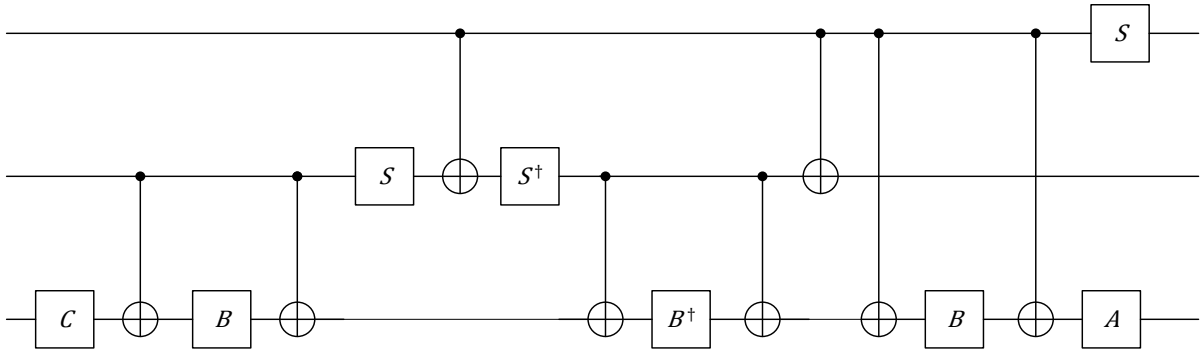
4.3.2. Implementation using only CNOT gates and one qubit gates

Theorem: Any operator $C_1^2(U)$ can be implemented using at most 8 gates on one qubit and 6 CNOT gates.

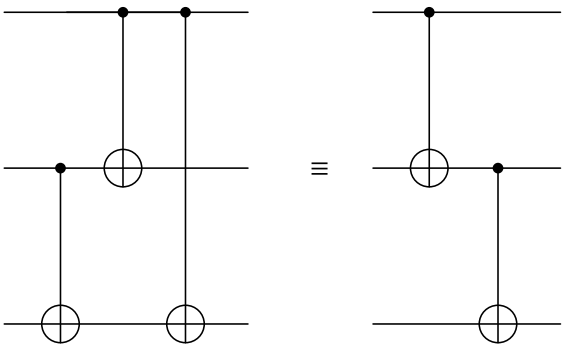
Demonstration: Using the decomposition $V = e^{i\alpha}AXBXC \Rightarrow V^\dagger = e^{-i\alpha}C^\dagger XB^\dagger XA^\dagger$, from the previous theorems, the following circuit can be built



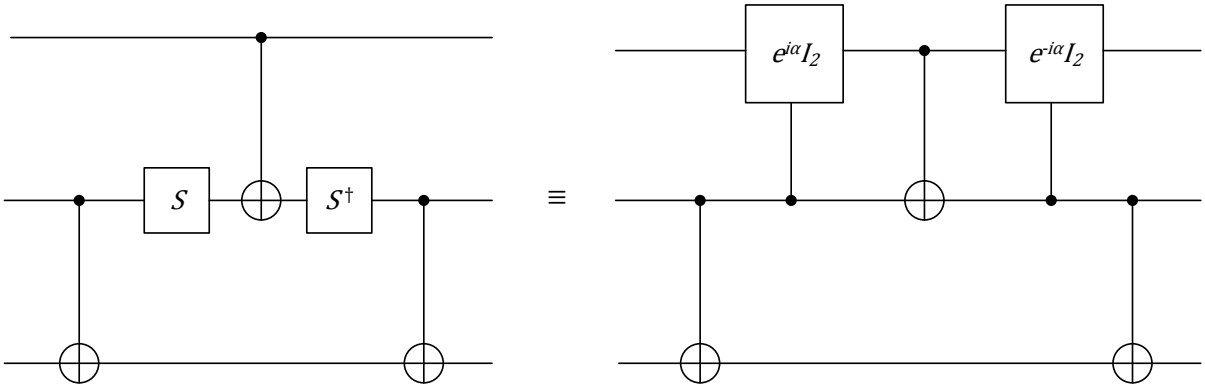
This circuit can be further simplified because $A^\dagger A = I_2$, $CC^\dagger = I_2$ and $S^\dagger S = I_2$:



The three cascaded CNOT gates produce: $|c_1c_2c_3\rangle \xrightarrow{CNOT} |c_1c_2\rangle|c_2 \oplus c_3\rangle \xrightarrow{CNOT} |c_1\rangle|c_1 \oplus c_2\rangle|c_2 \oplus c_3\rangle \xrightarrow{CNOT} |c_1\rangle|c_1 \oplus c_2\rangle|c_1 \oplus c_2 \oplus c_3\rangle$, and this state can be computed by using just two CNOT gates:



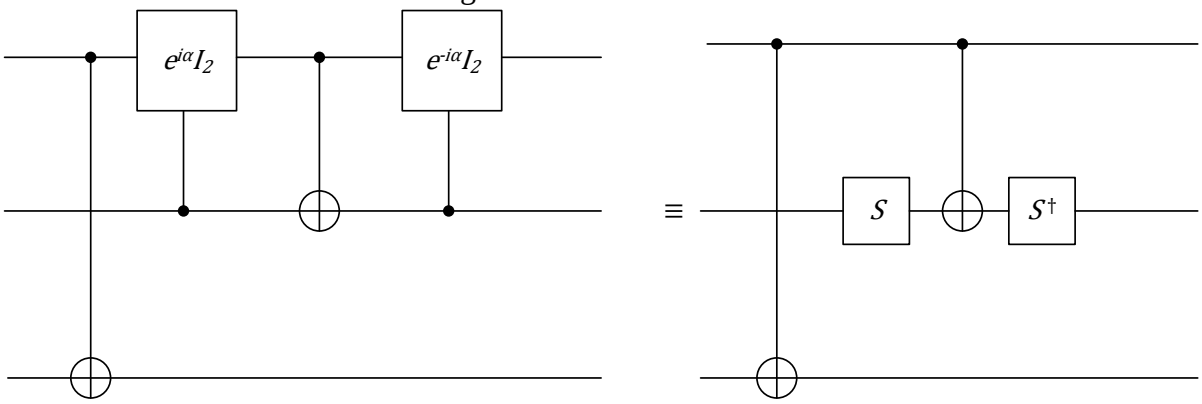
The circuit can be further simplified. According to the previous theorem the following circuit identity holds:



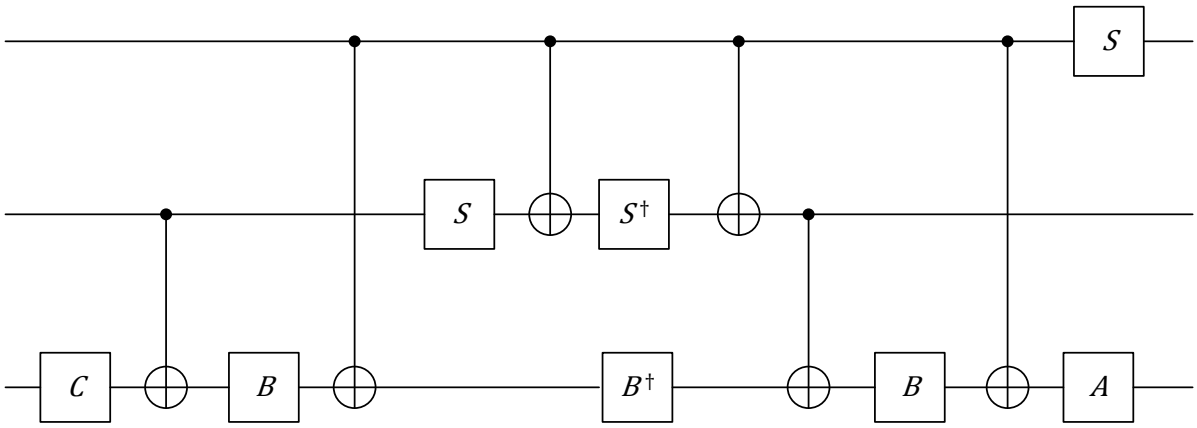
And the effect of this circuit is:

$$|c_1c_2c_3\rangle \xrightarrow{CNOT} |c_1c_2\rangle|c_2 \oplus c_3\rangle \xrightarrow{e^{i\alpha}I_2} e^{i\alpha c_2}|c_1\rangle|c_2\rangle|c_2 \oplus c_3\rangle$$
$$\xrightarrow{CNOT} e^{i\alpha c_2}|c_1\rangle|c_1 \oplus c_2\rangle|c_2 \oplus c_3\rangle \xrightarrow{e^{-i\alpha}I_2} e^{i\alpha c_2}e^{i\alpha(c_1\oplus c_2)}|c_1\rangle|c_1 \oplus c_2\rangle|c_2 \oplus c_3\rangle$$
$$\xrightarrow{CNOT} e^{i\alpha c_2}e^{i\alpha(c_1\oplus c_2)}|c_1\rangle|c_1 \oplus c_2\rangle|c_1 \oplus c_2 \oplus c_2 \oplus c_3\rangle$$
$$= e^{i\alpha c_2}e^{i\alpha(c_1\oplus c_2)}|c_1\rangle|c_1 \oplus c_2\rangle|c_1 \oplus c_3\rangle$$

The same state can be obtained using a more efficient circuit:



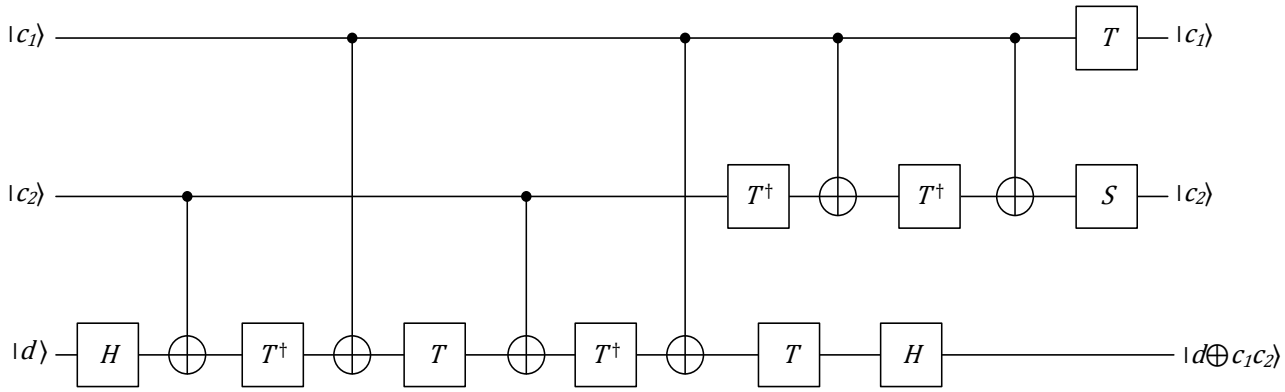
With these last simplifications, the desired circuit is:



4.4. Quantum implementation of universal reversible classical gates

4.4.1. Implementing Toffoli gate

Toffoli gate is a reversible gate which is universal for classical computation. It can be implemented using only the following types of gates: Hadamard, phase, CNOT and T. The circuit is:



4.4.2. Implementing the Fredkin gate by using Toffoli gates

The Fredkin gate, also called controlled swap gate, has a special signification in the classical theory of reversible computing. It is defined as a gate on 3 qubits that satisfies the following conditions:

- the control bit remains always unchanged
- the data bits are swapped iff the control bit is set

The Fredkin gate has two significant properties:

- reversible: applying the gate twice all the bits are switched to their original states.
- conservative: the number of set bits is conserved when they go through the gate.

In quantum computing, the Fredkin operator is unitary, his matrix for the computational basis state is:

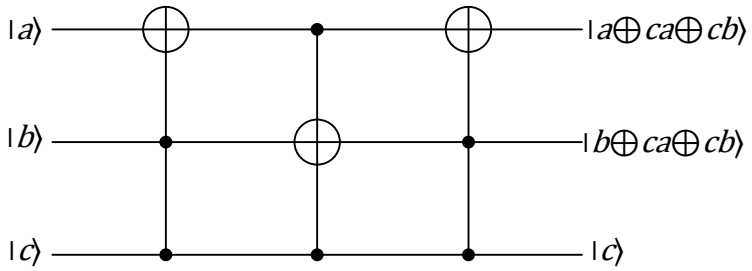
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Fredkin gate is represented as:

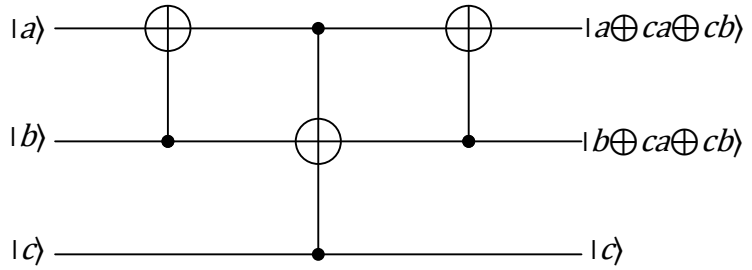
From its definition, one can deduct its algebraic description, where a, b, c are binary numbers each on 1 bit:

$$|abc\rangle \xrightarrow{Fredkin} |a \oplus ca \oplus cb\rangle|b \oplus ca \oplus cb\rangle|c\rangle$$

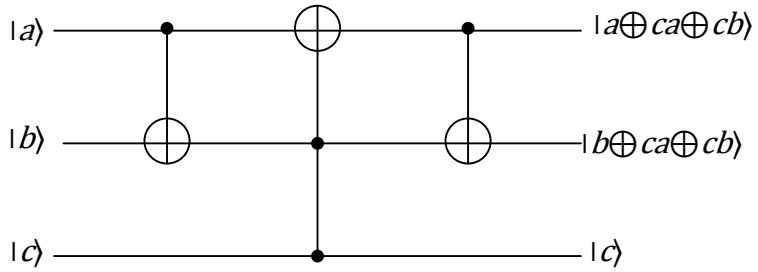
The Fredkin gate can be implemented using 3 Toffoli gates:



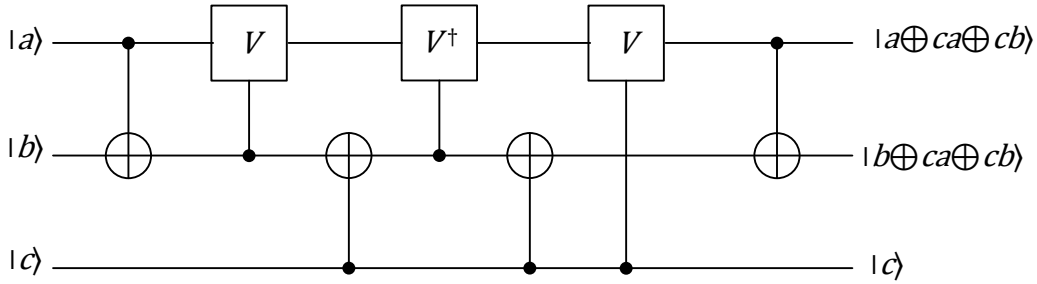
It can be noticed that the first and the last Toffoli gates can be replaced by CNOT gates, the new circuit having exactly the same effect:



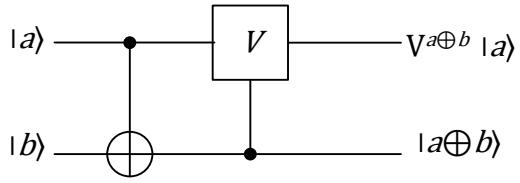
Because the Fredkin operator is symmetric in a, b, the following gate is also equivalent to the precedent one:



Next, replacing the Toffoli gate with the circuit from 4.3.1. that implements the Toffoli gate, one finds a circuit that implements the Fredkin operator using only 2 qubit gates:



where $V \equiv \frac{1-i}{2}(I_2 + iX)$. The first two gates can be grouped together into a single 2 qubit gate:

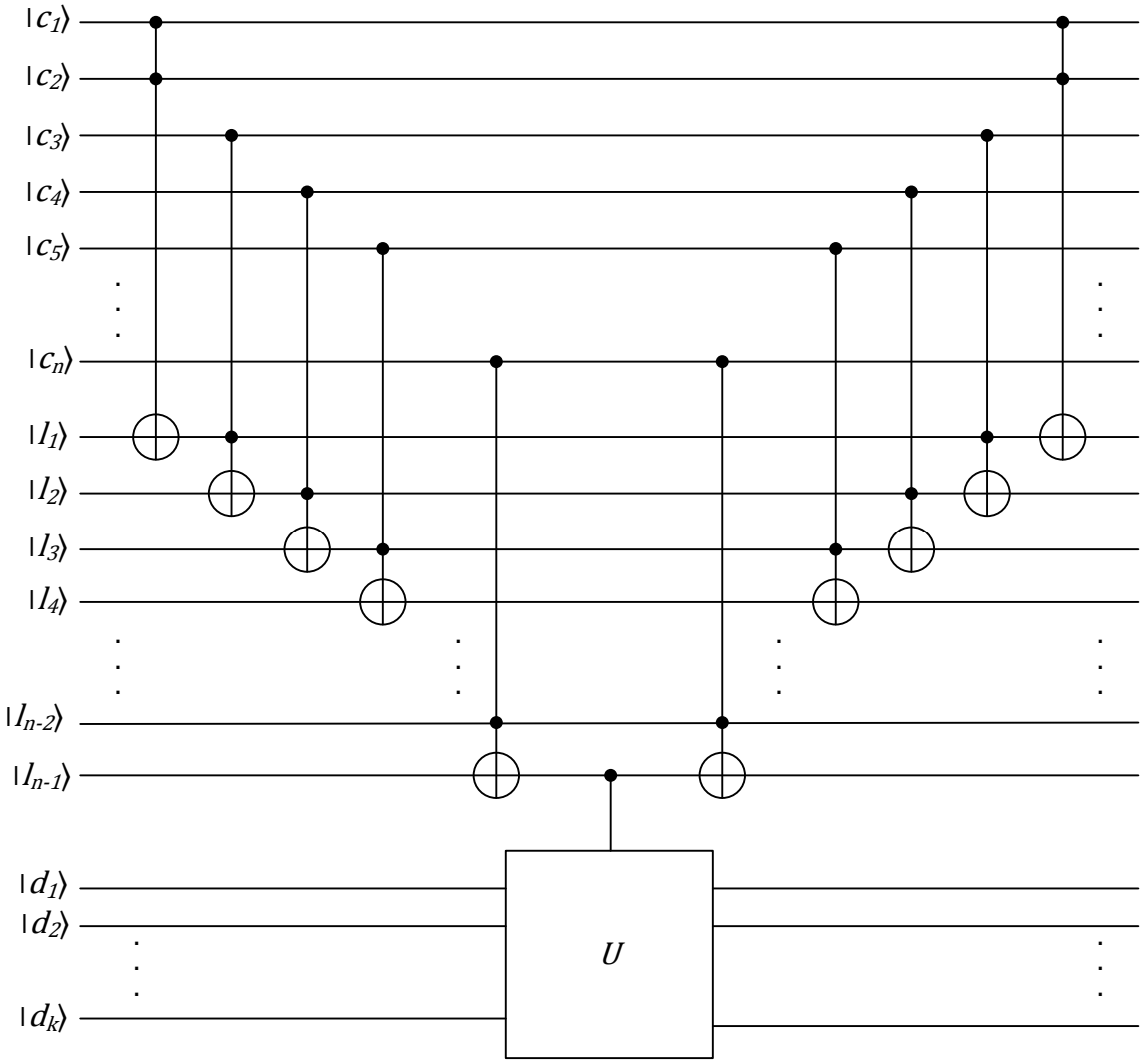


Thus it results an implementation of the Fredkin gate using only 6 gates on 2 qubits.

5. Controlled operators implementation

5.1. The linear implementation of the controlled operators

A simple circuit for implementing the controlled operators $C_k^n(U)$ in the generic case is presented below. The circuit is logically split in three steps and uses $n-1$ working qubits, set initially to the computational basis state$|0\rangle$.



Considering the control qubits in the computational basis state $|c_1c_2 \dots c_n\rangle$, the circuit implements in the first step the logical product between all the control bits: $c_1 \cdot c_2 \cdot \dots \cdot c_n$. For this, the circuit uses $n-1$ Toffoli gates and the $n-1$ working qubits. The first Toffoli gate implements the logical product between the first two control qubits, using as result the first working qubit. The second Toffoli gate adds the third control qubit to the previous product, using as result the second working qubit. And so on, until the last working qubit contains the logical product of all the n control qubits.
During the second step, the circuit implements the desired operator $C_k^n(U)$ using a controlled gate with just one control qubit $C_k^1(U)$. In the third step, the circuit implements the reverse operation that corresponds to the transformation from the first step in order to reset all the working qubits to their initial computational basis state $|0\rangle$.

So, in the end, all the control qubits remain unchanged, while the operator U is applied to the data qubits $|d_1 d_2 \ldots d_k\rangle$ iff the all the control qubits are set to their computational basis state $|1\rangle$.

5.2. Exponential implementation of the controlled operators

5.2.1. Implementing controlled operators on 3 qubits

The implementation of the controlled operators $C_k^n(U)$ can be realized using only operators $C_k^1(V)$, without using any working qubit. The method for implementing the controlled operator with 2 control qubits $C_1^2(U)$, described above, can be generalized starting from the logical operations performed with $C_1^2(U)$:

- V iff $c_2 = 1$
- $V^\dagger$ iff $c_1 \oplus c_2 = 1$
- V iff $c_1 = 1$

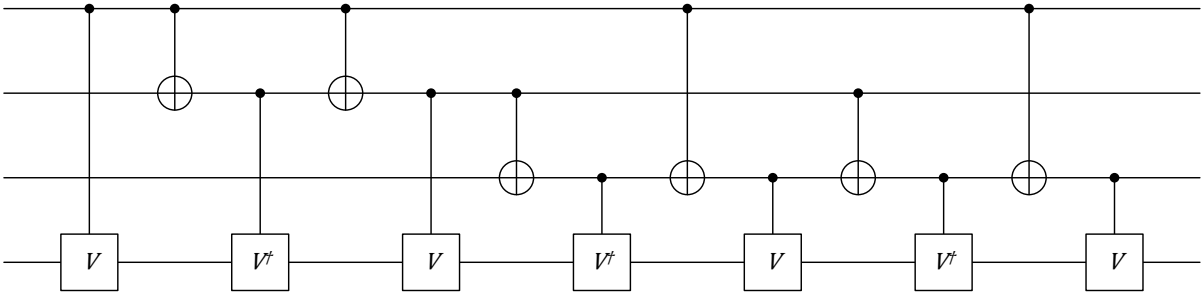
Because $c_1 - c_1 \oplus c_2 + c_2 = 2 \cdot (c_1 \wedge c_2)$, this sequence of operations is equivalent to applying $V^2 = U$ to the third qubit.
In a similar mode, for implementing $C_1^3(U)$ the operator V is required, such that $V^4 = U$. The sequence of operations on the fourth qubit is:

- (100) V iff $c_1 = 1$
- (110) $V^\dagger$ iff $c_1 \oplus c_2 = 1$
- (010) V iff $c_2 = 1$
- (011) $V^\dagger$ iff $c_2 \oplus c_3 = 1$
- (111) V iff $c_1 \oplus c_2 \oplus c_3 = 1$
- (101) $V^\dagger$ iff $c_1 \oplus c_3 = 1$
- (001) V iff $c_3 = 1$

For each of the above operations, the sets of bits on the left column indicate on which qubits it is required to check the set condition ($= 1$). The parity of each triplet of bits indicates the type of the operator to be applied: for triplets with odd number of set bits V is applied, while for the triplets with an even number of set bits $V^\dagger$ is applied.
By comparing this sequence of operations with the terms in the equation:

$$c_1 - c_1 \oplus c_2 + c_2 - c_2 \oplus c_3 + c_1 \oplus c_2 \oplus c_3 - c_1 \oplus c_3 + c_3 = 4 \cdot (c_1 \wedge c_2 \wedge c_3)$$

It can be verified that the sequence of operations above is equivalent to applying the operator V^4 on the fourth qubit iff $c_1 \wedge c_2 \wedge c_3 = 1$; that is the very definition of the controlled operator $C_1^3(U)$.
The circuit that implements the sequence of operations above is presented below. For an efficient implementation, the triplets above have to form a Gray sequence of codes.



5.2.2. Implementing controlled operators. Generalization

In a similar way to the construction of the above circuit, by using induction, the following generalization can be applied:
Theorem: For any $n \geq 2$, and any unitary operator U on k qubits, the controlled gate $C_k^n(U)$ can be implemented by a circuit on $n+k$ qubits composed of $2^n - 1$ gates $C_k^1(V)$ and $C_k^1(V^\dagger)$, together with $2^n - 2$ CNOT gates, where $V^{2^{n-1}} = U$ is a unitary.
The circuit can be obtained by using the following equation:

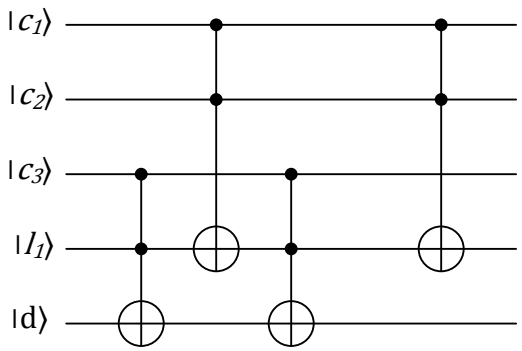
$$\sum_{i_1=1}^{n} c_i - \sum_{i_1<i_2}^{n} (c_{i_1} \oplus c_{i_2}) + \sum_{i_1<i_2<i_3}^{n} (c_{i_1} \oplus c_{i_2} \oplus c_{i_3}) - \ \ldots + (-1)^{n-1}(c_1 \oplus c_2 \oplus \ldots \oplus c_n)$$
$$= 2^{n-1} \cdot (c_1 \wedge c_2 \wedge \ldots \wedge c_n)$$

It should be observed that this methodology of implementing generic controlled operators has the advantage that doesn't require any working qubits. But the main disadvantage is that the number of required gates is rising exponentially with the number of control qubits. In contrast, when implementing by using the working qubits, the number of gates raises only linearly with the number of control qubits.
Here is again the old and well known compromise between the processing speed (i.e. number of gates connected sequentially) and the size of the working memory (i.e. the number of working qubits).

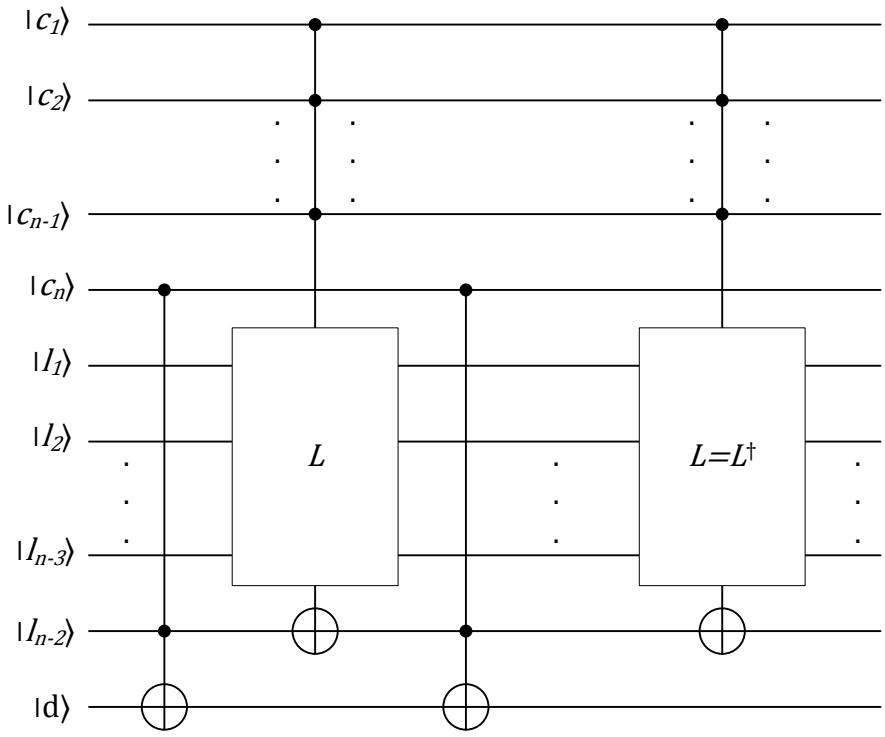
5.3. Quadratic implementation of controlled operators

5.3.1. Implementing the generic CNOT by using Toffoli

As a compromise between the exponential number of gates and the necessity of working qubits, it is possible to design a circuit that implements $C_k^n(U)$ using a single working qubit.
Lemma: For $n \geq 3$, $C^n(X)$ can be implemented using $2n-3$ Toffoli gates and $n-2$ working qubits that doesn't require any special initial state. By adding another $2n-5$ Toffoli gates, the circuit leaves the working qubits in the same state as it found them.
Demonstration: induction after n. For $n = 3$, the circuit below satisfies the requirements:
$|c_1c_2c_3\rangle|l_1\rangle|d\rangle \xrightarrow{4 \cdot Toffoli} |c_1c_2c_3\rangle|l_1\rangle|d \oplus c_1c_2c_3\rangle$



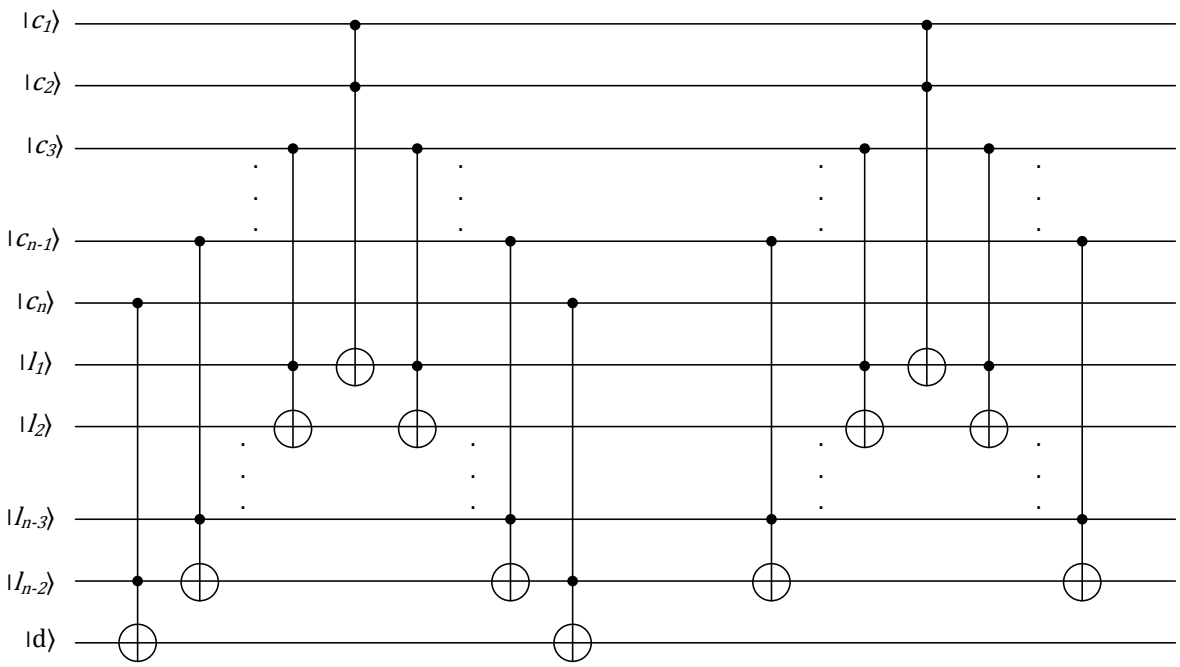
Assuming the statement is true for $n-1$, then it is also true for n:
$|c_1c_2 \ldots c_{n-1}\rangle|l_1l_2 \ldots l_{n-3}\rangle|d\rangle \xrightarrow{(4n-12) \cdot Toffoli} |c_1c_2 \ldots c_{n-1}\rangle|l_1l_2 \ldots l_{n-3}\rangle|d \oplus c_1c_2 \ldots c_{n-1}\rangle$
$\Rightarrow$
$|c_1c_2 \ldots c_n\rangle|l_1l_2 \ldots l_{n-2}\rangle|d\rangle \xrightarrow{(4n-8) \cdot Toffoli} |c_1c_2 \ldots c_n\rangle|l_1l_2 \ldots l_{n-2}\rangle|d \oplus c_1c_2 \ldots c_n\rangle$



$$|c_1c_2\dots c_n\rangle|l_1l_2\dots l_{n-2}\rangle|d\rangle \xrightarrow{Toffoli} |c_1c_2\dots c_n\rangle|l_1l_2\dots l_{n-2}\rangle|d\oplus l_{n-2}c_n\rangle$$
$$\xrightarrow{(2n-5)\cdot\ Toffoli} |c_1c_2\dots c_n\rangle|l_1\oplus c_1c_2\rangle|l_2\oplus c_1c_2c_3\rangle\dots|l_{n-2}\oplus c_1c_2\dots c_{n-1}\rangle|d\oplus l_{n-2}c_n\rangle$$
$$\xrightarrow{Toffoli} |c_1c_2\dots c_n\rangle|l_1\oplus c_1c_2\rangle|l_2\oplus c_1c_2c_3\rangle\dots|l_{n-2}\oplus c_1c_2\dots c_{n-1}\rangle|d$$
$$\oplus l_{n-2}c_n\oplus l_{n-2}c_n\oplus c_1c_2\dots c_{n-1}c_n\rangle$$
$$= |c_1c_2\dots c_n\rangle|l_1\oplus c_1c_2\rangle|l_2\oplus c_1c_2c_3\rangle\dots|l_{n-2}\oplus c_1c_2\dots c_{n-1}\rangle|d\oplus c_1c_2\dots c_{n-1}c_n\rangle$$
$$\xrightarrow{(2n-5)\cdot\ Toffoli} |c_1c_2\dots c_n\rangle|l_1\oplus c_1c_2\oplus c_1c_2\rangle|l_2\oplus c_1c_2c_3\oplus c_1c_2c_3\rangle\dots|l_{n-2}\oplus c_1c_2\dots c_{n-1}$$
$$\oplus c_1c_2\dots c_{n-1}\rangle|d\oplus c_1c_2\dots c_{n-1}c_n\rangle$$
$$= |c_1c_2\dots c_n\rangle|l_1l_2\dots l_{n-2}\rangle|d\oplus c_1c_2\dots c_n\rangle$$

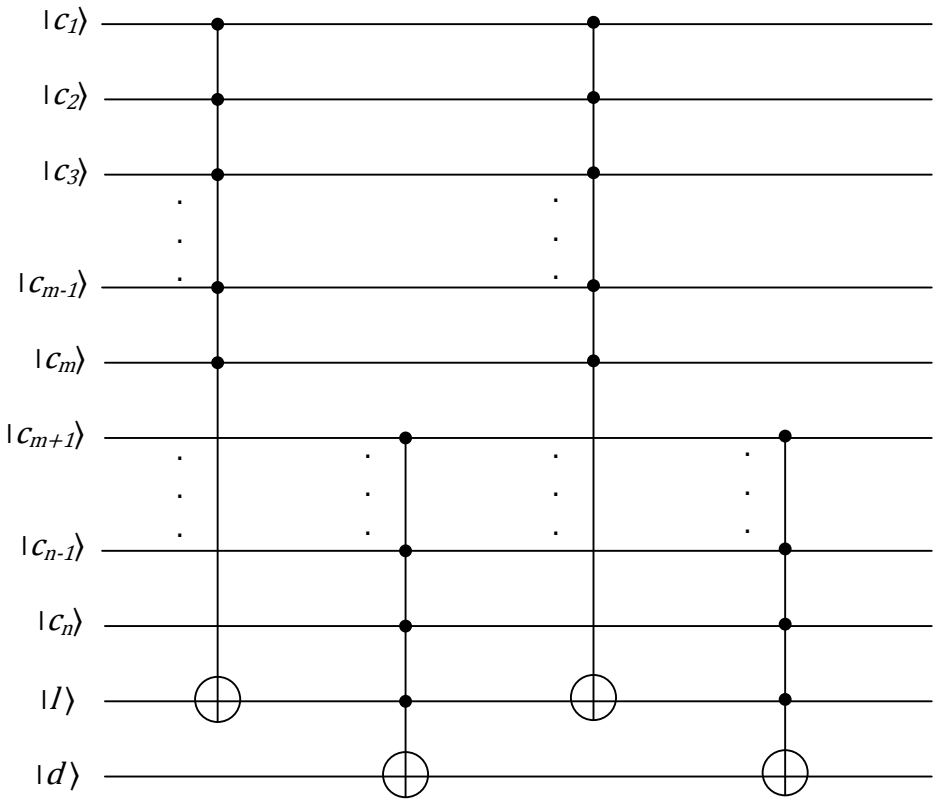
So, the total number of the Toffoli gates is: $1+(2n-5)+1+(2n-5)=4n-8$

The circuit for the generic case is:



Lemma: For any $n \geq 3$ and $1 \leq m \leq n-1$ the operator $C^n(X)$ can be implemented using a circuit having only two $C^m(X)$ gates, two $C^{n-m+1}(X)$ gates and a single working qubit.
Demonstration: is made by construction, see the following circuit. The qubits evolution is:

$$|c_1c_2 \dots c_n\rangle|l\rangle|d\rangle \xrightarrow{C^m(X)} |c_1c_2 \dots c_n\rangle|l \oplus c_1c_2 \dots c_m\rangle|d\rangle$$
$$\xrightarrow{C^{n-m+1}(X)} |c_1c_2 \dots c_n\rangle|l \oplus c_1c_2 \dots c_m\rangle|d \oplus lc_{m+1}c_{m+2} \dots c_n \oplus c_1c_2 \dots c_mc_{m+1}c_{m+2} \dots c_n\rangle$$
$$\xrightarrow{C^m(X)} |c_1c_2 \dots c_n\rangle|l \oplus c_1c_2 \dots c_m \oplus c_1c_2 \dots c_m\rangle|d \oplus lc_{m+1}c_{m+2} \dots c_n \oplus c_1c_2 \dots c_n\rangle$$
$$= |c_1c_2 \dots c_n\rangle|l\rangle|d \oplus lc_{m+1}c_{m+2} \dots c_n \oplus c_1c_2 \dots c_n\rangle$$
$$\xrightarrow{C^{n-m+1}(X)} |c_1c_2 \dots c_n\rangle|l\rangle|d \oplus lc_{m+1}c_{m+2} \dots c_n \oplus c_1c_2 \dots c_n \oplus lc_{m+1}c_{m+2} \dots c_n\rangle$$
$$= |c_1c_2 \dots c_n\rangle|l\rangle|d \oplus c_1c_2 \dots c_n\rangle$$



Corollary: For any $n \geq 5$, operator $C^n(X)$ can be implemented using a circuit containing $8n - 24$ Toffoli gates and a single working qubit. So, the temporal complexity is linear and the spatial complexity is constant.

Demonstration: Let $m = \left\lceil \frac{n}{2} \right\rceil$ and according the previous Lemma, the desired operator can be implemented using two $C^{\left\lceil \frac{n}{2} \right\rceil}(X)$ gates, two $C^{n-\left\lceil \frac{n}{2} \right\rceil+1}(X)$ gates and a single working qubit. According to the lemma above operator $C^{\left\lceil \frac{n}{2} \right\rceil}(X)$ applied to the working qubit $|l\rangle$ can be implemented using $4\left\lceil \frac{n}{2} \right\rceil - 8$ Toffoli gates and, as working qubits: $\left|c_{\left\lceil \frac{n}{2} \right\rceil+1} c_{\left\lceil \frac{n}{2} \right\rceil+2} \ldots c_n\right\rangle$, i.e. $n - \left\lceil \frac{n}{2} \right\rceil$ working qubits. The respective lemma hypothesis is satisfied because $-\left\lceil \frac{n}{2} \right\rceil \geq \left\lceil \frac{n}{2} \right\rceil - 2 \Leftrightarrow \frac{n}{2} + 1 \geq \left\lceil \frac{n}{2} \right\rceil$.

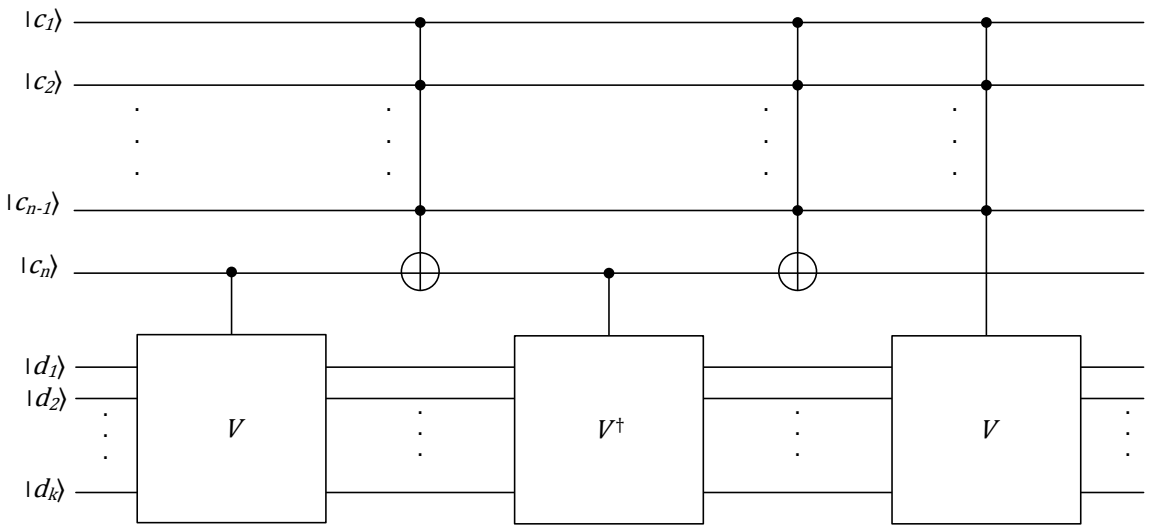
According to the same lemma, operator $C^{n-\left\lceil \frac{n}{2} \right\rceil+1}(X)$ applied to the data qubit $|d\rangle$ can be implemented using $4\left(n - \left\lceil \frac{n}{2} \right\rceil + 1\right) - 8 = 4\left(n - \left\lceil \frac{n}{2} \right\rceil - 1\right)$ Toffoli gates and, as working qubits: $\left|c_1 c_2 \ldots c_{\left\lceil \frac{n}{2} \right\rceil}\right\rangle$, i.e. $\left\lceil \frac{n}{2} \right\rceil$ working qubits. The respective lemma hypothesis is satisfied because $\left\lceil \frac{n}{2} \right\rceil \geq n - \left\lceil \frac{n}{2} \right\rceil + 1 - 2 \Leftrightarrow \left\lceil \frac{n}{2} \right\rceil \geq \frac{n}{2} - \frac{1}{2} \Leftarrow \left\lceil \frac{n}{2} \right\rceil \geq \frac{n}{2} \geq \frac{n}{2} - \frac{1}{2}$

In conclusion, the total number of the required Toffoli gates is:

$$\left(4\left\lceil \frac{n}{2} \right\rceil - 8\right) + 4\left(n - \left\lceil \frac{n}{2} \right\rceil - 1\right) + \left(4\left\lceil \frac{n}{2} \right\rceil - 8\right) + 4\left(n - \left\lceil \frac{n}{2} \right\rceil - 1\right) = 8\text{n} - 24$$

5.3.2. Implementing controlled operators without working qubits

Lemma: For any operator U, the controlled operator $C_k^n(U)$ can be implemented with the quantum circuit below, where $V^2 = U$.



The $C_k^n(U)$ definition is verified:

$$|c_1c_2\dots c_n\rangle|d_1d_2\dots d_k\rangle \xrightarrow{C_k^1(V)} |c_1c_2\dots c_n\rangle V^{c_n}|d_1d_2\dots d_k\rangle$$
$$\xrightarrow{C^{n-1}(X)} |c_1c_2\dots c_{n-1}\rangle|c_n\oplus c_1c_2\dots c_{n-1}\rangle V^{c_n}|d_1d_2\dots d_k\rangle$$
$$\xrightarrow{C_k^1(V^\dagger)} |c_1c_2\dots c_{n-1}\rangle|c_n\oplus c_1c_2\dots c_{n-1}\rangle V^{\dagger c_n\oplus c_1c_2\dots c_{n-1}}V^{c_n}|d_1d_2\dots d_k\rangle$$
$$\xrightarrow{C^{n-1}(X)} |c_1c_2\dots c_{n-1}\rangle|c_n\oplus c_1c_2\dots c_{n-1}\oplus c_1c_2\dots c_{n-1}\rangle V^{\dagger c_n\oplus c_1c_2\dots c_{n-1}}V^{c_n}|d_1d_2\dots d_k\rangle$$
$$= |c_1c_2\dots c_{n-1}\rangle|c_n\rangle V^{\dagger c_n\oplus c_1c_2\dots c_{n-1}}V^{c_n}|d_1d_2\dots d_k\rangle$$
$$\xrightarrow{C_k^{n-1}(V)} |c_1c_2\dots c_n\rangle V^{c_1c_2\dots c_{n-1}}V^{\dagger c_n\oplus c_1c_2\dots c_{n-1}}V^{c_n}|d_1d_2\dots d_k\rangle$$
$$= \begin{cases} |c_1c_2\dots c_n\rangle V^{c_1c_2\dots c_{n-1}}V^{\dagger c_1c_2\dots c_{n-1}}|d_1d_2\dots d_k\rangle & , c_n=0 \\ |c_1c_2\dots c_n\rangle V^{c_1c_2\dots c_{n-1}}V^{\dagger\overline{c_1c_2\dots c_{n-1}}}V|d_1d_2\dots d_k\rangle & , c_n=1 \end{cases}$$
$$= \begin{cases} |c_1c_2\dots c_n\rangle|d_1d_2\dots d_k\rangle & , c_1c_2\dots c_n=0 \\ |c_1c_2\dots c_n\rangle U|d_1d_2\dots d_k\rangle & , c_1c_2\dots c_n=1 \end{cases}$$

Theorem: Any operator $C^n(U)$ can be implemented using $O(n^2)$ elementary gates: one qubit gates together with Toffoli and CNOT.

Demonstration: by applying the previous lemma for $k=1$ and defining $cost(U)$ as the number of necessary gates for implementing operator U:

$$cost\big(C^n(U)\big) = cost\big(C(V)\big) + cost\big(C^{n-1}(X)\big) + cost\left(C(V^\dagger)\right) + cost\big(C^{n-1}(X)\big) + cost\big(C^{n-1}(V)\big)$$

According to the previous corollary, considering that the data qubit $|d\rangle$ is temporarily used as a working qubit, the number of necessary Toffoli gates for implementing operator $C^{n-1}(X)$ is:

$$cost\big(C^{n-1}(X)\big) = 8(n-1) - 24 = 8n - 32$$

According to a previous theorem, the two gates $C(V)$ and $C(V^\dagger)$ coupled like the circuit in the corollary above, require together 6 gates on one qubit and 4 CNOT gates:

$$cost\big(C(V)\big) + cost\left(C(V^\dagger)\right) = 6 + 4 = 10$$

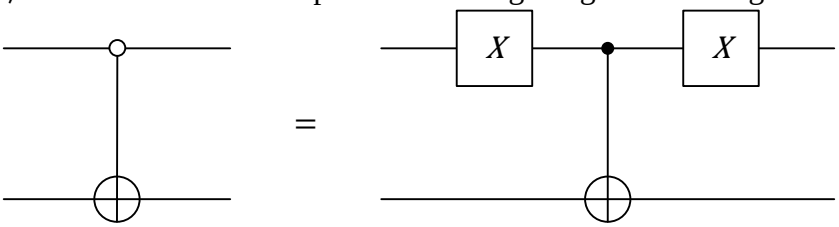
So: $cost\big(C^n(U)\big) = 10 + 2(8n-32) + cost\big(C^{n-1}(V)\big) = O(n) + cost\big(C^{n-1}(V)\big)$

And from this recursive relation it follows that: $cost\big(C^n(U)\big) = O(n^2)$.

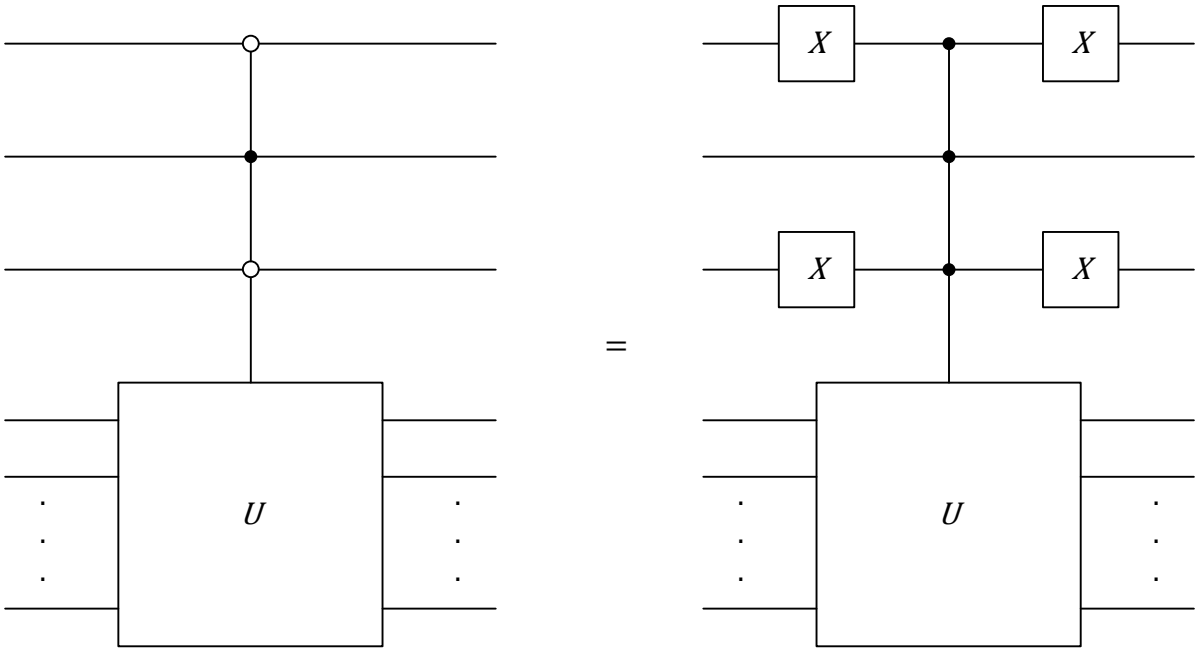
6. Universal quantum gates

6.1. Gates controlled by $|0\rangle$ qubits

In the previous circuits, the unitary operators from controlled gates are activated when the control qubits are set to $|1\rangle$, and they are deactivated when at least one control qubit is reset to $|0\rangle$. It is possible to build the controlled gates the other way around: the respective unitary operators are activated when all the control qubits are reset to $|0\rangle$. It is also possible to build gates that require for activation a mixed type of the control qubits, some set to $|1\rangle$, the others reset to $|0\rangle$.
For example, similarly to the CNOT gate, it is considered the simplest such circuit on two qubits – one control qubit and one data qubit, where the data qubit is flipped iff the control qubit is reset to $|0\rangle$. The general transformation for such a gate is: $|c\rangle|t\rangle \longrightarrow |c\rangle|1 \oplus c \oplus t\rangle = |c\rangle|\overline{c} \oplus t\rangle$. This circuit can be implemented using a regular CNOT gate:



In the generic case, consider a circuit on any number of qubits, which is controlled by a mixture of $|1\rangle$ and $|0\rangle$ control qubits. This type of circuit can be implemented using one qubit X gates and one regular controlled-U gate. The X gates act on the control qubits that activate the U gate by their $|0\rangle$ state.



$$|c_1c_2c_3\rangle|t_1t_2 \dots t_n\rangle \xrightarrow{X} |\overline{c_1}c_2\overline{c_3}\rangle|t_1t_2 \dots t_n\rangle \xrightarrow{U-controlat}$$
$$\xrightarrow{U-controlat} |\overline{c_1}c_2\overline{c_3}\rangle|t_1t_2 \dots t_n \oplus \overline{c_1}c_2\overline{c_3}\rangle \xrightarrow{X} |c_1c_2c_3\rangle|t_1t_2 \dots t_n \oplus \overline{c_1}c_2\overline{c_3}\rangle$$

6.2. Infinite sets of universal quantum gates

In classical computing, the set of following gates is universal, that is any operation can be implemented by using only these gates: $\{AND, NOT, FANOUT\}$. Other such universal sets for

classical computing are the one gate set $\{Toffoli\}$ and the one gate set $\{Fredkin\}$ [23]. This means that, because the Toffoli operator can be implemented by using a finite set of quantum gates, the set of all classical circuits is a sub-set in the set of all possible quantum circuits. The same conclusion can be reached by using the Fredkin operator.
A set of quantum gates is considered to be exactly universal for quantum computing iff any unitary operator acting on a finite number of qubits can be exactly implemented by a quantum circuit containing only gates from the given set [4]. For this type of universality only infinite sets have been discovered [5].
If using a probabilistic paradigm, a set of quantum gates is said to be approximately universal for quantum computing iff any unitary operator acting on a finite number of qubits can be approximated with an arbitrary big accuracy by a circuit containing only the gates from a given set. For this type of universality, finite and discrete sets have been discovered.

6.2.1. Level 2 matrices

Let a square unitary matrix of dimension $d \times d$, acting on a d-dimensional Hilbert space. By definition, a matrix U of order d that acts on any vector v having d components, is called a level 2 matrix iff the matrix modifies two components from v, leaving all the other components unchanged.
For example the matrix below is unitary and of level 2:

$$\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & u_{ii} & \cdots & u_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & u_{ji} & \cdots & u_{jj} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_j \\ \vdots \\ v_d \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ u_{ii}v_i + u_{ij}v_j \\ \vdots \\ u_{ji}v_i + u_{jj}v_j \\ \vdots \\ v_d \end{bmatrix}$$

$v_i \xrightarrow{U} v'_i = u_{ii}v_i + u_{ij}v_j$ and $v_j \xrightarrow{U} v'_j = u_{ji}v_i + u_{jj}v_j$ and $\forall k \notin \{i,j\} \Rightarrow v_k \xrightarrow{U} v_k$.
This is because the only non-trivial elements in the matrix are: $u_{ii}, u_{ij}, u_{ji}, u_{jj}$. The rest of the elements on the principal diagonal are 1, and the others elements in the matrix are all 0.
Furthermore, because the matrix has to be unitary, the non-trivial elements have to satisfy:

$$\begin{cases} |u_{ii}|^2 + |u_{ij}|^2 = 1 \\ |u_{ji}|^2 + |u_{jj}|^2 = 1 \\ u_{ii}^* u_{ji} + u_{ij}^* u_{jj} = 0 \\ u_{ii}^* u_{ij} + u_{ji}^* u_{jj} = 0 \end{cases}$$

Note that the inverse of such a unitary, level 2 matrix is also unitary and of level 2.

6.2.2. Decomposing matrices using level 2 factors

Theorem: Any unitary matrix U of order d can be decomposed in a finite product of unitary matrices, each of them of dimension $d \times d$ and level 2.
Demonstration: Explicitly construct a set of level 2 matrices $U_i^{(j)}$ where $i \geq j \wedge \{i,j\} \subset \{1, \dots, d-1\}$ such that:

$$\left[\prod_{1}^{j=d-1} \left(\prod_{j}^{i=d-1} U_i^{(j)} \right) \right] U = I_d$$

Then, by multiplying to the left by level 2 matrices: $U_i^{(j)\dagger}$, the desired decomposition results:

$$U = \prod_{j=1}^{d-1} \left(\prod_{i=j}^{d-1} U_i^{(j)\dagger} \right)$$

The requested unitary, level 2 matrices are chosen in the following way: $U_1^{(1)}$ is chosen such that it acts only on lines 1 and 2: $\begin{bmatrix} u_{11} \\ u_{21} \\ \vdots \end{bmatrix}, \begin{bmatrix} u_{12} \\ u_{22} \\ \vdots \end{bmatrix}$ ... $\begin{bmatrix} u_{1d} \\ u_{2d} \\ \vdots \end{bmatrix}$. Then by multiplying $U_1^{(1)} U$ one gets a matrix U' that has $u'_{21} = x_{21} u_{11} + x_{22}\, u_{21} = 0$. So,

$$U_1^{(1)} = I_d, if\ u_{21} = 0$$

$$U_1^{(1)} = \begin{bmatrix} x_{11} & x_{12} & 0 & \cdots & 0 & \cdots & 0 \\ x_{21} & x_{22} & 0 & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, if\ u_{21} \neq 0$$

This implies resolving the system:

$$\begin{cases} x_{11}^* x_{11} + x_{12}^* x_{12} = 1 \\ x_{21}^* x_{21} + x_{22}^* x_{22} = 1 \\ x_{11}^* x_{21} + x_{12}^* x_{22} = 0 \\ x_{11}^* x_{12} + x_{21}^* x_{22} = 0 \\ x_{21} u_{11} + x_{22}\, u_{21} = 0 \end{cases} \Rightarrow \begin{cases} x_{12}^* = x_{21} = \dfrac{u_{21}}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} \\ -x_{11}^* = x_{22} = \dfrac{-u_{11}}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} \end{cases}$$

So, for the general case, the first matrix in the factorization is:

$$U_1^{(1)} = \begin{bmatrix} \dfrac{u_{11}^*}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} & \dfrac{u_{21}^*}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} & 0 & \cdots & 0 & \cdots & 0 \\ \dfrac{u_{21}}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} & \dfrac{-u_{11}}{\sqrt{u_{11} u_{11}^* + u_{21} u_{21}^*}} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, if\ u_{21} \neq 0$$

And:

$$U_1^{(1)} U = \begin{bmatrix} u'_{11} & u'_{12} & \cdots & u'_{1j} & \cdots & u'_{1d} \\ 0 & u'_{22} & \cdots & u'_{2j} & \cdots & u'_{2j} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u_{i1} & u_{i2} & \cdots & u_{ij} & \cdots & u_{id} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ u_{d1} & u_{d2} & \cdots & u_{dj} & \cdots & u_{dd} \end{bmatrix}$$

In a similar way, all the elements $u_{i1}, i \in \{2, \ldots, d\}$ from the first column can be turned into 0, by using the unitary level 2 matrices $U_i^{(1)}$, acting on lines 1 şi i. So, after such $d - 1$ steps, the product of unitary matrices gives:

$$U_{d-1}^{(1)} \dots U_1^{(1)} U = \begin{bmatrix} u''_{11} & u''_{12} & \cdots & u''_{1j} & \cdots & u''_{1d} \\ 0 & u'_{22} & \cdots & u'_{2j} & \ddots & u'_{2j} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & u'_{i2} & \cdots & u'_{ij} & \cdots & u'_{id} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & u'_{d2} & \cdots & u'_{dj} & \cdots & u'_{dd} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & u'_{22} & \cdots & u'_{2j} & \ddots & u'_{2j} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & u'_{i2} & \cdots & u'_{ij} & \cdots & u'_{id} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & u'_{d2} & \cdots & u'_{dj} & \cdots & u'_{dd} \end{bmatrix}$$

Next, the same procedure is applied for the sub-matrix obtained from U by eliminating the first row and the first column. Using the unitary level 2 matrices $U_2^{(2)} \dots U_{d-1}^{(2)}$, acting on lines 2 and i, $i \in \{3, \dots, d\}$, all elements $u_{i2}, i \in \{3, \dots, d\}$ are turned into 0. It must be noted that by multiplying by the matrices $U_i^{(2)}$, first column remains unchanged because neither $U_i^{(2)}$ acts on the first row. Therefore the 0 values obtained at the previous step are preserved:

$$0 = u_{21} \xrightarrow{U_i^{(2)}} 0x_{22} + 0x_{2i} = 0 \quad \text{and} \quad 0 = u_{i1} \xrightarrow{U_i^{(2)}} 0x_{i2} + 0x_{ii}$$

After multiplying to the left with the second set of matrices, the product gives:

$$U_{d-1}^{(2)} \dots U_2^{(2)} U_{d-1}^{(1)} \dots U_1^{(1)} U = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & \ddots & 0 \\ 0 & 0 & u''_{33} & \cdots & u''_{3j} & \cdots & u''_{3d} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & u''_{i3} & \cdots & u''_{ij} & \cdots & u''_{id} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & u''_{d3} & \cdots & u''_{dj} & \cdots & u''_{dd} \end{bmatrix}$$

This procedure is repeated in a similar way, at each step the set of matrices required for annulling the respective column is smaller and smaller, and the already annulled columns remain unchanged.

■

So, the total number of unitary matrices of order d and level 2 required for factoring a unitary matrix of order d is therefore at most:

$$nr_d \leq (d-1) + (d-2) + \cdots + 1 = \frac{d(d-1)}{2}$$

Hence, because the matrix associated to an operator of a quantum circuit on n qubits has order $d = 2^n$, this matrix will be decomposed using an exponential number of unitary level 2 matrices: $2^{n-1}(2^n - 1)$. But, for some special matrices it is possible to find more efficient decompositions.

Yet, it is very important to prove that there are unitary matrices U of order d that cannot be decomposed into a product of unitary level 2 matrices containing less than $d - 1$ terms. Therefore, by using this type of level 2 decomposition, some circuits can be implemented efficiently, by using a polynomial number of gates corresponding to those level 2 matrices, but there are also are circuits which require an exponential number of gates.

This observation can be proved by contradiction. Suppose $\exists\, d > 1$ such that any unitary matrix of order d can be decomposed in $U = U_1 U_2 \dots U_{d-2}$, where U_i are unitary matrices of level 2. But $\forall d > 1$ there is at least one unitary matrix U of order d that doesn't contain any 0 element on at least one of its columns. If for this matrix, there was a level 2 decomposition, then it would result that, considering only the column k that doesn't have any 0 element, $u_{ik} \neq 0, \forall i \in \{1, \dots, d\}$:

$$\begin{bmatrix} u_{1k} \\ u_{2k} \\ \vdots \\ u_{kk} \\ \vdots \\ u_{dk} \end{bmatrix} = U_1 U_2 \dots U_{d-2} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

In order for a unitary, level 2 matrix U_i to have any effect in the above product, it must necessarily act on at least one line that was modified before by one of the matrices U_j where $j > i$. So, the number of affected lines by the $U_1 U_2 \dots U_{d-2}$ product is at most $d-2$. But because the initial column (on which U_{d-2} acts) contains $d-1$ elements which are 0, it means that the resulted column contains at least a 0 element, which is in contradiction with a supposition above.

6.2.3. Implementing unitary matrices of level 2

Considering a unitary matrix U of order $d = 2^n$ and level 2, the problem is to implement this matrix using a quantum circuit on n qubits.

Considering the computational basis states $|b^i\rangle$, with $i \in \{1, \dots, d\}$, any vector can be decomposed as: $|v\rangle = \sum_{i=1}^{d} \langle b^i | v \rangle |b^i\rangle$. If the matrix U_{ij} acts only on the components i and j, it will affect only the vectors from the sub-space generated by the two vectors $|b^i\rangle = |b_1^i b_2^i \dots b_n^i\rangle$ and $|b^j\rangle = |b_1^j b_2^j \dots b_n^j\rangle$, where $b_k^i \in \{0,1\}, k \in \{1, 2, \dots, n\}$ and $b_l^j \in \{0,1\}, l \in \{1, 2, \dots, n\}$. So, the effect of the sub-circuit implementing U_{ij} has to be:

$$\left(|b^i\rangle \xrightarrow{U_{ij}} U_{ij}|b^i\rangle\right) \wedge \left(|b^j\rangle \xrightarrow{U_{ij}} U_{ij}|b^j\rangle\right) \wedge \left(\forall t \in \{1, 2, \dots, d\} - \{i, j\} \Rightarrow |b^t\rangle \xrightarrow{U_{ij}} |b^t\rangle\right)$$

The required circuit is built in 4 major steps:

Step 1. Build the set of Grey codes that connects $b_1^i b_2^i \dots b_n^i$ and $b_1^j b_2^j \dots b_n^j$: $\{g^1, \dots g^m\}$, where $g^k, k \in \{1, 2, \dots, m\}$ are binary numbers on n bits such that

$g^1 = b^i$ and $g^m = b^j$ and $\forall k \in \{1, 2, \dots, m-1\}, \exists l_k \in \{1, 2, \dots, m\} \Rightarrow g^k \oplus g^{k+1} = 2^{l_k}$.

This means the binary numbers g^k and g^{k+1} differ only by exactly one bit. So,

$$|g^k\rangle = |g_1^k g_2^k \dots g_{l_k-1}^k g_{l_k}^k g_{l_k+1}^k \dots g_{n-1}^k g_n^k\rangle \Rightarrow |g^{k+1}\rangle = |g_1^k g_2^k \dots g_{l_k-1}^k \overline{g_{l_k}^k} g_{l_k+1}^k \dots g_{n-1}^k g_n^k\rangle$$

Step 2. For each $k \in \{1, 2, \dots, m-2\}$, build a controlled circuit on n qubits that swaps only the computational basis states $|g^k\rangle \xleftrightarrow{C^{n-1}(X)_{l_k}} |g^{k+1}\rangle$, preserving all the others unchanged. The effect of this circuit is:

$$|b^i\rangle = |g^1\rangle \xrightarrow{C^{n-1}(X)_{l_1}} |g^2\rangle \xrightarrow{C^{n-1}(X)_{l_2}} \dots \xrightarrow{C^{n-1}(X)_{l_{m-3}}} |g^{m-2}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-2}}} |g^{m-1}\rangle$$

$$|g^2\rangle \xrightarrow{C^{n-1}(X)_{l_1}} |g^1\rangle$$
$$|g^3\rangle \xrightarrow{C^{n-1}(X)_{l_2}} |g^2\rangle$$
$$\vdots$$
$$|g^{m-1}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-2}}} |g^{m-2}\rangle$$

Step 3. From the original matrix U_{ij} of order d and level 2, build the matrix $\widetilde{U}_{ij}$, of order 2.

$$U_{ij} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & u_{ii} & \cdots & u_{ij} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & u_{ji} & \cdots & u_{jj} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \qquad \widetilde{U}_{ij} = \begin{bmatrix} u_{ii} & u_{ij} \\ u_{ji} & u_{jj} \end{bmatrix}$$

Serially add to the end of the circuit obtained above, another circuit composed of a controlled gate $C^{n-1}\left(\widetilde{U}_{ij}\right)_{l_{m-1}}$. Gate $\widetilde{U}_{ij}$ acts on the qubit l_{m-1}, which corresponds to the bit that differentiates g^{m-1} and g^m, and it is controlled by the other $n-1$ qubits, which are the same in g^{m-1} and g^m. Note that:

$$|g^m\rangle = \left|b^j\right\rangle = \left|b_1^j b_2^j \dots b_{l_{m-1}-1}^j b_{l_{m-1}}^j b_{l_{m-1}+1}^j \dots b_{n-1}^j b_n^j\right\rangle$$

$$|g^{m-1}\rangle = \left|b_1^j b_2^j \dots b_{l_{m-1}-1}^j \overline{b_{l_{m-1}}^j} b_{l_{m-1}+1}^j \dots b_{n-1}^j b_n^j\right\rangle$$

So, the effect of the circuit is therefore:

$$|g^{m-1}\rangle \xrightarrow{C^{n-1}\left(\widetilde{U}_{ij}\right)_{l_{m-1}}} \left|b_1^j\right\rangle\left|b_2^j\right\rangle \dots \left|b_{l_{m-1}-1}^j\right\rangle \widetilde{U}_{ij} \left|\overline{b_{l_{m-1}}^j}\right\rangle \left|b_{l_{m-1}+1}^j\right\rangle \dots \left|b_{n-1}^j\right\rangle\left|b_n^j\right\rangle = U_{ij}|g^{m-1}\rangle$$

$$\left|b^j\right\rangle = |g^m\rangle \xrightarrow{C^{n-1}\left(\widetilde{U}_{ij}\right)_{l_{m-1}}} \left|b_1^j\right\rangle\left|b_2^j\right\rangle \dots \left|b_{l_{m-1}-1}^j\right\rangle \widetilde{U}_{ij} \left|b_{l_{m-1}}^j\right\rangle \left|b_{l_{m-1}+1}^j\right\rangle \dots \left|b_{n-1}^j\right\rangle\left|b_n^j\right\rangle = U_{ij}\left|b^j\right\rangle$$

Step 4. To the circuit obtained in the previous steps, add the reverse of the sub-circuit built at Step 2, which is composed of the same gates in the reverse order (because it contains only $C^{n-1}(X)$ gates). The only effect of this circuit is:

$$|g^{m-1}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-2}}} |g^{m-2}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-3}}} \dots \xrightarrow{C^{n-1}(X)_{l_2}} |g^2\rangle \xrightarrow{C^{n-1}(X)_{l_1}} |g^1\rangle = \left|b^i\right\rangle$$

$$|g^{m-2}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-2}}} |g^{m-1}\rangle$$

$$|g^{m-3}\rangle \xrightarrow{C^{n-1}(X)_{l_{m-3}}} |g^{m-2}\rangle$$

$$\vdots$$

$$\left|b^i\right\rangle = |g^1\rangle \xrightarrow{C^{n-1}(X)_{l_1}} |g^2\rangle$$

In the end, the final circuit, after the four steps above has the desired effect:

$$\left|b^i\right\rangle = |g^1\rangle \xrightarrow{Step\ 2} |g^{m-1}\rangle \xrightarrow{Step\ 3} U_{ij}|g^{m-1}\rangle \xrightarrow{Step\ 4} U_{ij}\left|b^i\right\rangle$$

$$|g^2\rangle \xrightarrow{Step\ 2} |g^1\rangle \xrightarrow{Step\ 3} |g^1\rangle \xrightarrow{Step\ 4} |g^2\rangle$$

$$|g^3\rangle \xrightarrow{Step\ 2} |g^2\rangle \xrightarrow{Step\ 3} |g^2\rangle \xrightarrow{Step\ 4} |g^3\rangle$$

$$\vdots$$

$$|g^{m-1}\rangle \xrightarrow{Step\ 2} |g^{m-2}\rangle \xrightarrow{Step\ 3} |g^{m-2}\rangle \xrightarrow{Step\ 4} |g^{m-1}\rangle$$

$$\left|b^j\right\rangle = |g^m\rangle \xrightarrow{Step\ 2} \left|b^j\right\rangle \xrightarrow{Step\ 3} U_{ij}\left|b^j\right\rangle \xrightarrow{Step\ 4} U_{ij}\left|b^j\right\rangle$$

■

6.2.4. Complexity evaluation

The maximum number of gates required for implementing a unitary matrix of 2^n and level 2 is:

$$cost\left(U_{ij}\right) = (n-1)cost\left(C^{n-1}(X)\right) + cost\left(C^{n-1}\left(\widetilde{U}_{ij}\right)\right) + (n-1)cost\left(C^{n-1}(X)\right)$$

As shown before, $C^n(U)$ can be implemented using only one qubit gates and CNOT gates, the required number of gates being $cost\left(C^n(U)\right) = O(n^2)$. Therefore, U_{ij} can be also implemented using only one qubit gates and CNOT gates, the number of required gates being:

$$cost\left(U_{ij}\right) = (n-1)O(n^2) + O(n^2) + (n-1)O(n^2) = O(n^3)$$

Next, because a generic operator U on n qubits can be decomposed in a product of level 2 matrices, with at most $2^{n-1}(2^n-1)$ factors, it follows that U can be implemented using only one qubit gates and CNOT gates, the total number of gates being:

$$cost(U) = 2^{n-1}(2^n-1)O(n^3) = O(n^3 4^n)$$

It is obvious that this construction is not the most efficient one in all the cases, as it contains an exponential number of gates. For this reason, finding efficient quantum algorithms for specific problems requires a different kind of construction for the implementing circuit.
■

6.3. Discrete sets of universal quantum gates

In order to use the quantum computing model in practice, a few problems must be solved first. Most important, especially because the processes at quantum level are extremely susceptible to noisy interferences, the model must provide for fault-tolerant implementations. Therefore, it is mandatory that the unitary operators (which are used for modeling quantum computations) are proved to have implementations that are based only on fault tolerant quantum gates. There are several well-established results on the universality of quantum bases, which rest primarily on using a non-elementary gate, i.e. a gate that performs a single qubit rotation by an irrational multiple of 2π. However, a direct, fault tolerant realization of such gate is not really possible, therefore they can't be easily used in noisy quantum environments.
There are quantum codes that can be used to show that a small sub-set of elementary quantum gates: Hadamard, phase, CNOT, called the normalizer group, can be implemented in a fault-tolerant manner. But this set of gates it is not enough for universality as it doesn't spawn the whole set of unitary operators. This fact led to the suggestion of adding of a new elementary gate to the normalizer group: Toffoli, a gate which can be also implemented fault-tolerantly. But a direct proof of the universality of the new set was not provided.
There are some indirect proofs which follow an indirect approach by demonstrating the direct equivalence between the Shor's basis and other universal bases. That is, these bases provide simple and accurate circuits that implement the operators in Shor's basis. There are also other categories of bases which were proved to not be directly equivalent to Shor's basis; that is, they can only be approximated by gates in the Shor's basis, and not implemented exactly.

6.3.1. Basic circuit for the non-elementary rotation

The quantum circuit below implements a basic rotation operator, around the z axis, with a specific angle: $R_z(\theta)$, where $\cos\theta = 3/5$. This angle θ was chosen so that it is an irrational multiple of 2π. To prove the circuit, we use the definitions of the gates involved: Hadamard, phase and Toffoli.

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

$$\xrightarrow{Toffoli} \frac{1}{2}[(|00\rangle + |01\rangle + |10\rangle)|\psi\rangle + |11\rangle X|\psi\rangle]$$

$$\xrightarrow{phase} \frac{1}{2}[(|00\rangle + |01\rangle + |10\rangle)S|\psi\rangle + |11\rangle SX|\psi\rangle]$$

$$\xrightarrow{Toffoli} \frac{1}{2}[(|00\rangle + |01\rangle + |10\rangle)S|\psi\rangle + |11\rangle XSX|\psi\rangle]$$

$$\xrightarrow{Hadamard} \frac{1}{4}[|00\rangle(3S + XSX)|\psi\rangle + (|01\rangle + |10\rangle - |11\rangle)(S - XSX)|\psi\rangle]$$

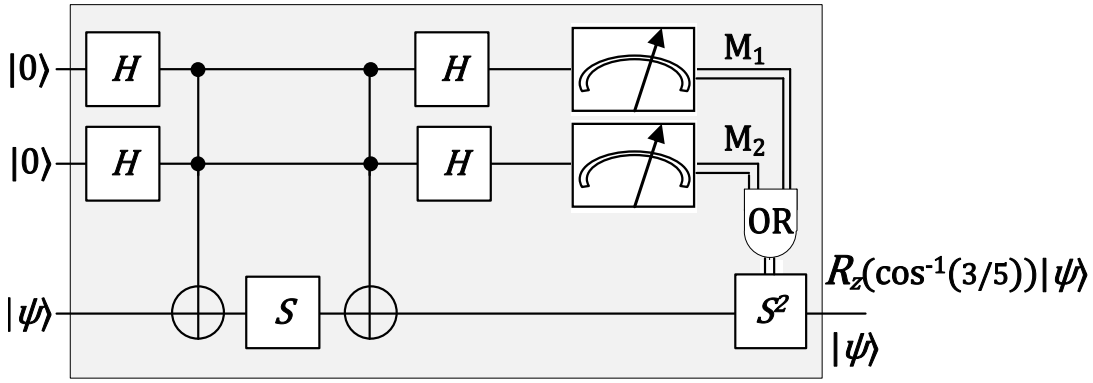
Considering the following identities for quantum gates:

$$3S + XSX = \sqrt{10}e^{i\frac{\pi}{4}}R_z(\theta), \quad \text{where } \cos\theta = \frac{3}{5}$$

$$S - XSX = (1 - i)Z = (1 - i)S^2$$

The circuit state just before the measurements are performed becomes then:

$$\longrightarrow \frac{\sqrt{10}}{4} e^{i\frac{\pi}{4}} |00\rangle R_z(\theta) + \frac{1-i}{4}(|01\rangle + |10\rangle - |11\rangle) S^2 |\psi\rangle$$



The above circuit implements the elementary rotation operator with probability $P_{00} = 5/8$.

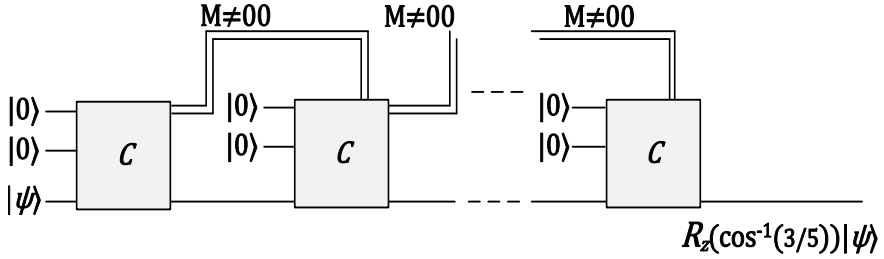
6.3.2. Circuit for the elementary rotation, with unitary probability

The quantum circuit above applies the $R_z(\theta)$ operation to the target qubit if the measurement outcomes on the control qubits are both 0. Otherwise, if at least one measurement returns 1, the target qubit will be left unchanged, in the same state. This decision is implemented by the classic OR gate at the end, which takes as input the classic bits from the measurement and then controls the application of the final quantum gate $S^2 = Z$. The probabilities of these four different outcomes, given by the two control qubits, can be easily calculated as following:

$$P_{00} = \left| \frac{\sqrt{10}}{4} e^{i\frac{\pi}{4}} \right|^2 = \frac{5}{8}$$

$$P_{01} \equiv P_{10} \equiv P_{11} = \left| \frac{1-i}{4} \right|^2 = \frac{1}{8}$$

As the above equations indicate, the probability of the circuit to actually apply the desired rotation operator is much higher than the probability of performing a no-op. Still, it is possible to improve this probability, in order to make it approach 1, by successively applying the very same quantum circuit "*C*" until the rotation operation is performed. This process is schematically presented in the next figure.



The above circuit implements the elementary rotation operator with a probability that approaches 1. The process runs as follows: if, at the current step, the measurement outcome on at least one control qubit is 1, then apply the circuit again, by using two new control qubits set to $|0\rangle$ and the same target qubit as returned by the circuit. Else, if at any step *n* the measurement outcomes on the control qubits are both 0, then the target qubit has been

transformed with $R_z(\theta)$, and the process stops. The probability for the process to stop at step n is therefore:

$$P(n) = P_{00}[\sum_{k=0}^{n-1}(P_{01} + P_{10} + P_{11})^k] = \frac{5}{8}\sum_{k=0}^{n-1}\left(\frac{3}{8}\right)^k$$

This is because in all the previous steps (1..n-1) the measurement outcome was either $|01\rangle$ or $|10\rangle$ or $|11\rangle$, and at the current step (n) the measurement outcome was $|00\rangle$. And it can be easily observed that the sum above is the sum of a geometrical series, therefore as n raises, the probability $P(n)$ approaches 1:

$$\lim_{n\to\infty}\big(P(n)\big) = \frac{5}{8}\frac{1}{1-\frac{3}{8}} = 1$$

6.3.3. Approximating unitary operators

Since the set of unitary operations is continuous, it is clear that a discrete set of gates is not sufficient to implement an arbitrary unitary operation. Rather, a discrete set can be used to only approximate any unitary operation. Considering U and V are two unitary operators on the same state space, U being the required target operator and V being the operator that is actually implemented, the error in approximation is defined by:

$$E(U,V) \equiv \max_{|\psi\rangle}\|(U-V)|\psi\rangle\|$$

This definition guarantees that if the respective error is small, then a measurement performed on the actually implemented operator, using any initial state and any measurement operator, gives similar statistics as if the same measurement were to be performed on the required target operator. Furthermore, if a sequence of gates is used to approximate another sequence of gates, the errors add up at most linearly:

$$E(U_mU_{m-1}\dots U_1, V_mV_{m-1}\dots V_1) \le \sum_{j=1}^{m} E\big(U_j, V_j\big)$$

6.3.4. Approximating the rotation operator

With respect to the two above relations, if the operators involved are rotation operators, the error can be expressed in terms of the rotation angles. In the equalities below, z axis can be replaced by any arbitrary axis.

$$\begin{aligned}
E\big(R_z(\alpha), R_z(\alpha+\beta)\big) &= \max_{|\psi\rangle}\big\|\big(R_z(\alpha) - R_z(\alpha+\beta)\big)|\psi\rangle\big\| \\
&= \max_{|\psi\rangle}\big(\langle\psi|\big(R_z(-\alpha) - R_z(-\alpha-\beta)\big)\big(R_z(\alpha) - R_z(\alpha+\beta)\big)|\psi\rangle\big) \\
&= \max_{|\psi\rangle}\big(\langle\psi|\big(2I - R_z(-\beta) - R_z(\beta)\big)|\psi\rangle\big) = \max_{|\psi\rangle}\left(\left\langle\psi\middle|2I\left(1-\cos\frac{\beta}{2}\right)\middle|\psi\right\rangle\right) \\
&= 2\left(1-\cos\frac{\beta}{2}\right)\max_{|\psi\rangle}(\langle\psi|I|\psi\rangle) = 2\left(1-\cos\frac{\beta}{2}\right)
\end{aligned}$$

This relation can be generalized to approximations by successive identical rotations:

$$E(R_z(\alpha), R_z(\theta)^n) = 2\left(1-\cos\frac{\big((n\theta)\mathrm{mod}2\pi\big)-\alpha}{2}\right)$$

The pigeonhole principle implies that if θ is an irrational multiple of 2π, then, for any α and any desired accuracy $\delta > 0$ it is possible to find n, such that $\big((n\theta)\mathrm{mod}2\pi\big) - \alpha < \delta$. But, as the angle $\big((n\theta)\mathrm{mod}2\pi\big)$ tends to approach α, in the same time $\cos\frac{\big((n\theta)\mathrm{mod}2\pi\big)-\alpha}{2}$ tends to approach 1. And as a consequence, $E(R_z(\alpha), R_z(\theta)^n)$ in the equation above tends to

approach 0. In conclusion, for any α and any desired accuracy $\epsilon > 0$, there is a number n_ε depending on both α and the desired accuracy, such as:

$$E(R_z(\alpha), R_z(\theta)^{n_{\alpha,\varepsilon}}) < \frac{\epsilon}{3}$$

Now, it can be shown that the discrete set of quantum gates formed by the normalizer group, plus the Toffoli gate is universal for quantum computation; that is an arbitrary unitary operation on *d* qubits can be approximated to an arbitrary accuracy using a circuit composed only from these gates. The circuit obtained will most likely have to be applied several times, the number of applications being direct proportional with the desired accuracy in the approximation. Firstly, because Pauli operators satisfy $HZH = X$, any single qubit unitary operation can be factored into a product of rotations around z axis and Hadamard operators:

$$U \cong R_z(\alpha)R_x(\beta)R_z(\gamma) = R_z(\alpha)HR_z(\beta)HR_z(\gamma)$$

Then, from the last two equations, it follows that the circuit above, together with two more Hadamard gates, can be used to successfully approximate any single qubit unitary operation:

$$E(U, R_z(\theta)^{n_\alpha}HR_z(\theta)^{n_\beta}HR_z(\theta)^{n_\gamma}) < 3\frac{\epsilon}{3} + 2E(H,H) = \epsilon$$

Furthermore, because any unitary operator on *d* qubits can be factored into a product of two-level unitary operators on *d* qubits, and because these two-level unitary operators on *d* qubits can in turn be exactly (i.e. no approximation needed) implemented using only single qubit gates and CNOT gates, this implies that any unitary operator on *d* qubits can be approximately implemented, with arbitrary accuracy ϵ, using only Hadamard, phase, CNOT and Toffoli gates, i.e. the gates from Shor's basis.

Performance considerations

The direct proof provided for the universality of the Shor basis raises a few questions regarding the efficiency of the quantum circuit models and the amount of computing resources required to approximate unitary operations. Unfortunately it is not possible to approximate generic unitary operators on *d* qubits using a circuit of size polynomial in *d*. Yet, the search for universal fault-tolerant bases must always consider the efficiency aspect. Although most of the unitary transformations can only be implemented by approximation very inefficiently, that is the number of fault-tolerant gates is exponential in the number of qubits of the operator, it may be possible that some universal bases are more efficient than others to approximate some specific set of unitary operators.

7. The Fourier Transform

A quantum computer can factor natural numbers in polynomial time, more precisely by performing only $O(n^2 \log n \log\log n)$ operations, with any desired accuracy. This represents an exponential gain compared to the known classical algorithms. The question that comes naturally next is: what kinds of other algorithms could be exponentially improved by the quantum computing paradigm?
One of the key ingredients suggesting an answer to this question is the quantum Fourier transform, which is used also in natural numbers factorization, but also in many other interesting problems. The quantum Fourier transformation is a quantum algorithm that computes the Fourier transform on the set of amplitudes describing quantum mechanics states. By itself, this algorithm doesn't bring any worthy efficiency improvement with respect to computing the Fourier transform on classical data sets. But this algorithm provides an efficient phase estimation that is a more efficient way of approximating the singular values of a unitary operator in given specific circumstances.
The phase estimation algorithm can be used in solving other problems:
- factoring natural numbers
- finding the order of an element in a finite group
- counting the solutions of a search problem; this is achieved by combining it with the quantum search algorithm

Additionally, QFT can be used for solving other problems, which are considered to be intractable on classical computers:
- hidden subgroups
- discrete logarithm

7.1. Quantum Fourier Transform (QFT)

One of the most used ways of solving a hard computing problem is to transform it into another problem that can be easier solved, or that already has been solved. So, this kind of transformations became a field of study by themselves. A very important discovery for quantum computing was that one of such transformations can be more efficiently implemented, compared with the classical implementation. This opened the door for finding efficient solutions to a whole class of problems that are based on the respective transformation.
This transformation was the Fourier transformation, together with its more relevant variant for digital computing – discrete Fourier transformation (DFT) [35]. Using usual mathematical notations, DFT transforms a vector of complex numbers $x_0, x_1 \dots x_{N-1}$ into another vector of complex numbers $y_0, y_1 \dots y_{N-1}$, with the same dimension:

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}}$$

QFT is similar with the definition above, but its data and significance is a bit different [32]. QFT is by definition a unitary linear operator on a vector space of dimension N that transforms the set of orthonormal vectors in the computational basis state $|0\rangle, |1\rangle, \dots, |N-1\rangle$ according to the relation:

$$|j\rangle \xrightarrow{QFT} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

The set of obtained vectors is still orthonormal.

$$\||j\rangle\| = \left\|\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}}|k\rangle\right\| = \sqrt{\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{-2\pi ijk}{N}}\langle k|\right)\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}}|k\rangle\right)} =$$

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

$$\forall j \neq l, \langle j|l\rangle = \left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{-2\pi ijk}{N}}\langle k|\right)\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi ilk}{N}}|k\rangle\right) = \frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{2\pi i(l-j)k}{N}}\langle k|k\rangle$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{2\pi i(l-j)k}{N}} = \frac{1}{N}\frac{1-e^{\frac{2\pi i(l-j)}{N}N}}{1-e^{\frac{2\pi i(l-j)}{N}}} = \frac{1}{N}\frac{1-e^{2\pi i(l-j)}}{1-e^{\frac{2\pi i(l-j)}{N}}} = 0$$

Any vector $|x\rangle$ from the respective space can be also transformed by QFT as:

$$|x\rangle = \sum_{j=0}^{N-1} x_j|j\rangle \xrightarrow{QFT} |y\rangle = \sum_{k=0}^{N-1} y_k|k\rangle$$

where the set of complex numbers y_k is the DFT of the set x_j.

$$|x\rangle = \sum_{j=0}^{N-1} x_j|j\rangle \xrightarrow{QFT}$$

$$\sum_{j=0}^{N-1}\left(x_j\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{\frac{2\pi ijk}{N}}|k\rangle\right) = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} x_j e^{\frac{2\pi ijk}{N}}|k\rangle = \sum_{k=0}^{N-1}\left(\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} e^{\frac{2\pi ijk}{N}}x_j\right)|k\rangle$$

$$= \sum_{k=0}^{N-1} y_k|k\rangle = |y\rangle$$

Because QFT is a unitary operator [38], it can be implemented by a quantum circuit. In order to achieve this, a more refined version of the transformation formula is needed. For this assume:

- $N = 2^n$, where n is a natural number
- the set of orthonormal vectors $|0\rangle, |1\rangle, \dots, |2^n - 1\rangle$ that constitute a basis in the vector space where QFT is defined, is also the computational basis state for a quantum circuit on n qubits.

For each vector $|k\rangle$ in computational basis state, represent it in binary format $k = k_1k_2 \dots k_n$, $k_l \in \{0, 1\}$:

$$k = k_1 2^{n-1} + k_2 2^{n-2} + \cdots + k_n 2^0 = \sum_{l=1}^{n} k_l 2^{n-l}$$

Then QFT becomes:

$$|j\rangle \xrightarrow{QFT} \frac{1}{2^{\frac{n}{2}}}\sum_{k=0}^{2^n-1} e^{2\pi ijk2^{-n}}|k\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{k_1=0}^{1}\sum_{k_2=0}^{1}\dots\sum_{k_n=0}^{1}\left(e^{2\pi ij\sum_{l=1}^{n}(k_l 2^{-l})}|k_1k_2\dots k_n\rangle\right)$$

$$= \frac{1}{2^{\frac{n}{2}}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \dots \sum_{k_n=0}^{1} \left(\prod_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right) = \frac{1}{2^{\frac{n}{2}}} \prod_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right) =$$

$$= \frac{1}{2^{\frac{n}{2}}} \prod_{l=1}^{n} \left(|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right)$$

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

Where by writing j in the binary format and because $l \in \{1, 2 \dots n\}$:

$$e^{2\pi i j 2^{-l}} = e^{2\pi i 2^{-l} (j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0)}$$

$$= e^{2\pi i j_1 2^{n-l-1}} e^{2\pi i j_2 2^{n-l-2}} \dots e^{2\pi i j_{n-l-1} 2^1} e^{2\pi i j_{n-l} 2^0} e^{2\pi i j_{n-l+1} 2^{-1}} \dots e^{2\pi i j_n 2^{-l}}$$

$$= e^{2\pi i j_{n-l+1} 2^{-1}} \dots e^{2\pi i j_n 2^{-l}} = e^{2\pi i (j_{n-l+1} 2^{-1} + j_{n-l+2} 2^{-2} + \dots + j_n 2^{-l})}$$

So, for each vector $|j\rangle = |j_1 j_2 \dots j_n\rangle$ from the computational basis state, QFT on each qubit now becomes:

$$|j_1\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i j_n 2^{-1}} |1\rangle \right)$$

$$|j_2\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i (j_{n-1} 2^{-1} + j_n 2^{-2})} |1\rangle \right)$$

...

$$|j_{n-1}\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i (j_2 2^{-2} + \dots + j_n 2^{-n-1})} |1\rangle \right)$$

$$|j_n\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i (j_1 2^{-1} + j_2 2^{-2} + \dots + j_{n-1} 2^{-n-1} + j_n 2^{-n})} |1\rangle \right)$$

7.2. Implementing the quantum Fourier transform

Starting from the equations above, the circuit for implementation QFT is built using Hadamard gates and controlled gates on one qubit S_l, where this unitary operator of phase transformation is defined as:

$$S_l \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i 2^{-l}} \end{bmatrix}$$

The effect of the operator S_l on one qubit is:

$$|\psi\rangle = a|0\rangle + b|1\rangle \xrightarrow{S_l} a|0\rangle + b e^{2\pi i 2^{-l}} |1\rangle$$

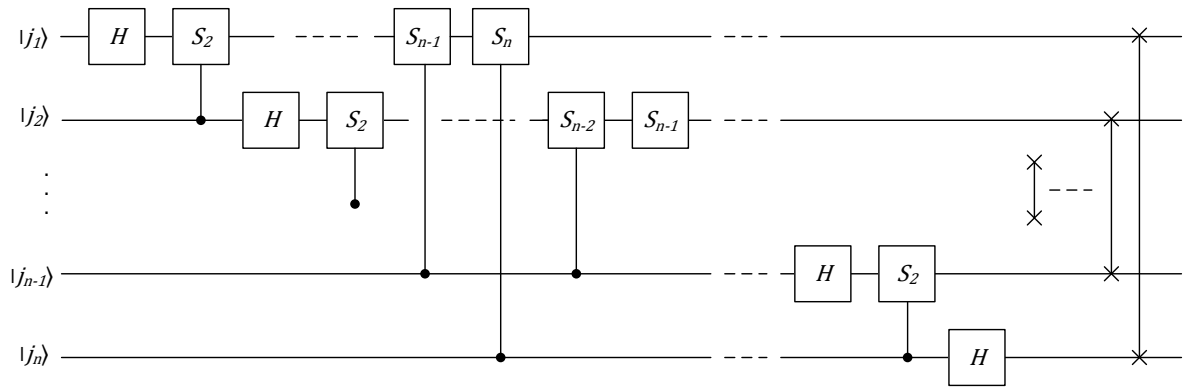
Whereas the controlled operator $C^1(S_l)$, with the control qubit in the computational basis state $|j_l\rangle$, acts as:

$$|j_l\rangle|\psi\rangle = |j_l\rangle(a|0\rangle + b|1\rangle) \xrightarrow{C^1(S_l)} |j_l\rangle \left(a|0\rangle + b e^{2\pi i j_l 2^{-l}} |1\rangle \right)$$

The Hadamard acts according to:

$$\begin{cases} |0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \\ |1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \end{cases} \equiv |j_l\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i j_l 2^{-1}} |1\rangle \right)$$

The circuit for implementing QFT is:



Adding the swapping gates at the circuit exit is actually optional because reading the result could be done in the reverse order.

For verifying the circuit above, the gates are applied in sequential order. In reality, the gates act in parallel where possible.

$$|j_1 j_2 \dots j_n\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i j_1 2^{-1}}|1\rangle\right)|j_2 \dots j_n\rangle$$

$$\xrightarrow{C^1(S_2)} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_1 2^{-1} + j_2 2^{-2})}|1\rangle\right)|j_2 \dots j_n\rangle$$

...

$$\xrightarrow{C^1(S_{n-1})} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_1 2^{-1} + j_2 2^{-2} + \dots + j_{n-1} 2^{-n-1})}|1\rangle\right)|j_2 \dots j_n\rangle$$

$$\xrightarrow{C^1(S_n)} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_1 2^{-1} + j_2 2^{-2} + \dots + j_{n-1} 2^{-n-1} + j_n 2^{-n})}|1\rangle\right)|j_2 \dots j_n\rangle = (QFT|j_n\rangle)|j_2 \dots j_n\rangle$$

$$\xrightarrow{H} (QFT|j_n\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i j_2 2^{-1}}|1\rangle\right)|j_3 \dots j_n\rangle$$

$$\xrightarrow{C^1(S_2)} (QFT|j_n\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_2 2^{-1} + j_3 2^{-2})}|1\rangle\right)|j_3 \dots j_n\rangle$$

...

$$\xrightarrow{C^1(S_{n-1})} (QFT|j_n\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_2 2^{-1} + j_3 2^{-2} + \dots + j_n 2^{-n-1})}|1\rangle\right)|j_3 \dots j_n\rangle$$

$$= (QFT|j_n\rangle)(QFT|j_{n-1}\rangle)|j_3 \dots j_n\rangle$$

...

$$(QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)|j_{n-1} j_n\rangle$$

$$\xrightarrow{H} (QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i j_{n-1} 2^{-1}}|1\rangle\right)|j_n\rangle$$

$$\xrightarrow{C^1(S_2)} (QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i (j_{n-1} 2^{-1} + j_n 2^{-2})}|1\rangle\right)|j_n\rangle$$

$$= (QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)(QFT|j_2\rangle)|j_n\rangle$$

$$\xrightarrow{H} (QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)(QFT|j_2\rangle)\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i j_n 2^{-1}}|1\rangle\right)$$

$$= (QFT|j_n\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)(QFT|j_2\rangle)(QFT|j_1\rangle)$$

$$\xrightarrow{Swap(j_n, j_1)} (QFT|j_1\rangle)(QFT|j_{n-1}\rangle) \dots (QFT|j_3\rangle)(QFT|j_2\rangle)(QFT|j_n\rangle)$$

$$\xrightarrow{Swap(j_{n-1}, j_2)} (QFT|j_1\rangle)(QFT|j_2\rangle)(QFT|j_{n-2}\rangle) \dots (QFT|j_3\rangle)(QFT|j_{n-1}\rangle)(QFT|j_n\rangle)$$

...

$$\xrightarrow{Swap(j_{\lfloor \frac{n}{2} \rfloor}, j_{\lfloor \frac{n}{2} \rfloor + 1})} (QFT|j_1\rangle)(QFT|j_2\rangle) \dots (QFT|j_{n-1}\rangle)(QFT|j_n\rangle) = QFT|j_1 j_2 \dots j_n\rangle$$

7.3. Complexity evaluation

For computing the number of gates used, start from the first qubit: one Hadamard gate followed by $n-1$ controlled gates $S_2 \dots S_n$; so in total n gates. Acting on the second qubit, there were one Hadamard gate and $n-2$ controlled gates $S_2 \dots S_{n-1}$; in total $n-1$ gates. The counting progresses in a similar way until the end, on the last qubit acting just 1 Hadamard gate. Then, the circuit contains $\left\lfloor \frac{n}{2} \right\rfloor$ swapping gates. The number of used gates is therefore:

$$C\big(QFT(n)\big) = n + (n-1) + \cdots + 2 + 1 + \left\lfloor \frac{n}{2} \right\rfloor = \frac{n(n+1)}{2} + \left\lfloor \frac{n}{2} \right\rfloor \leq \frac{n^2}{2} + n.$$

The complexity of the QFT implementation is therefore $O\left(\frac{n^2}{2}\right)$. This is an exponential improvement compared to the classical algorithm for Fast Fourier Transform, which has complexity: $O(n2^n)$.

At a first glance, it would look as a bunch of applications that use FFT, like signal processing, speech recognition, etc., could also benefit from an exponential improvement. Unfortunately there is no known way to make these applications use of QFT, the main problem being that the amplitudes of the signals can't be directly accessed via measurements in the quantum computer. So, applying the Fourier transform on the amplitudes cannot be determined as efficiently. To make things worse, there is no known efficient way of preparing the original quantum state. So, finding ways of practically using the QFT improvements is more subtle than it looks [52].

The good news is that the construction of the quantum circuit that implements QFT doesn't require exponential precision on the implementation of the gates contained within. For example, considering U as the ideal QFT on n qubits and V as the real one that results if the controlled gates $C^1(S_k)$ are built with limited polynomial precision $\Delta = \frac{1}{p(n)}$, then, the total error, defined as $E(U,V) \equiv max_{|\psi\rangle} \|(U-V)|\psi\rangle\|$ is a function $\Theta\left(\frac{n^2}{p(n)}\right)$. Such polynomial precision in implementation of each gate is sufficient for guarantying a polynomial accuracy of the entire circuit [17].

8. Phase estimation

8.1. Quantum procedure for phase estimation

Considering a unitary operator U that has a singular vector $|u\rangle$ with the corresponding singular value $e^{2\pi i\varphi}$:

$$U|u\rangle = e^{2\pi i\varphi}|u\rangle$$

The problem is to estimate the unknown phase, a sub-unitary real number [53].
For the estimation consider two types black boxes (oracles):

- ones capable to prepare the state represented by singular vector $|u\rangle$
- ones that are capable of implementing controlled operations $C\left(U^{2^j}\right)$, for any $j \in \mathbb{N}$

The implementation uses two quantum registers. The first one is a control register with t qubits initially in the computational basis state $|0\rangle$. Number t is chosen depending on:

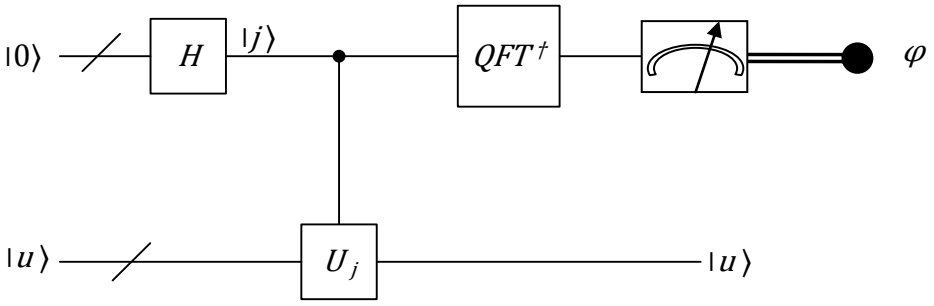
- number of decimals required to approximate phase φ, i.e. the number of bits used for reading the result
- the probability with which the algorithm returns the correct result, i.e. the number φ

The second register, the data register, has a number of qubits determined by the dimension of the vector space and initially is in the singular state $|u\rangle$.

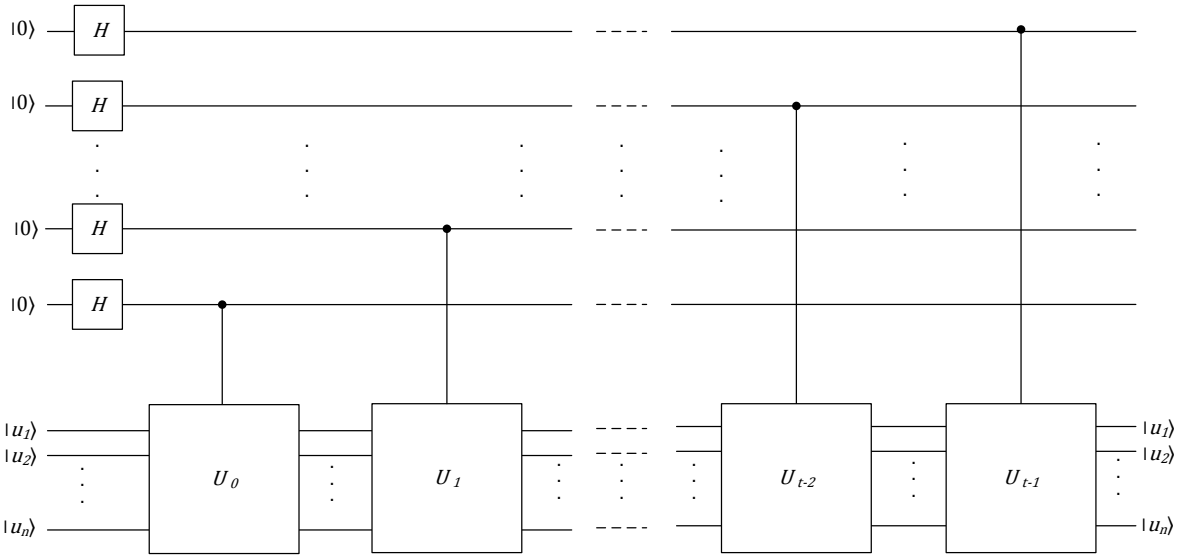
8.2. Quantum circuit for phase estimation

The algorithm comprises four principal phases [54]:

- apply Hadamard operators on the control qubits
- apply the black boxes on the data qubits
- apply inverse QFT on the control qubits
- read the result by performing a measurement in the computational basis state on the control qubits.



The circuit for the first two phases is presented below, where $U_j \equiv U^{2^j}$:



Because $|u\rangle$ is a singular vector for operator U, the state of the data register remains practically unchanged. For the control register, the qubit with order $j, j \in \{1 \dots t\}$ is transformed as:

$$|0\rangle_j|u\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|u\rangle \xrightarrow{C^1\left(U^{2^{t-j}}\right)} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 2^{t-j}\varphi}|1\rangle\right)|u\rangle$$

So, the state of the control register is:

$$\frac{1}{2^{\frac{t}{2}}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i 2^{1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right)$$

$$= \frac{1}{2^{\frac{t}{2}}}\sum_{k=0}^{2^t-1} e^{2\pi i k\varphi}|k\rangle$$

Intuitively, it is assumed the desired phase is represented exactly by a binary fraction $\varphi = \frac{1}{2^t}\sum_{l=1}^{t}\varphi_l 2^{t-l}$, with $\varphi_l \in \{0, 1\}$, the equation above becomes QFT:

$$\frac{1}{2^{\frac{t}{2}}}\left(|0\rangle + e^{2\pi i \varphi_t 2^{-1}}|1\rangle\right)\left(|0\rangle + e^{2\pi i\left(\varphi_{t-1}2^{-1} + \varphi_t 2^{-2}\right)}|1\rangle\right) \dots \left(|0\rangle + e^{2\pi i\left(\varphi_1 2^{-1} + \dots + \varphi_t 2^{-t}\right)}|1\rangle\right)$$

Therefore, by applying the reverse QFT, one gets exactly the computational basis state corresponding to the desired phase: $|\varphi_1\varphi_2 \dots \varphi_t\rangle$. And performing a measurement in the computational basis state will return exactly (i.e. with probability 1) the value φ.
In reality, if φ is a real number, the algorithm returns an approximation:

$$\frac{1}{2^{\frac{t}{2}}}\left(\sum_{k=0}^{2^t-1} e^{2\pi i k\varphi}|k\rangle\right)|u\rangle \xrightarrow{QFT^\dagger} |\tilde{\varphi}\rangle|u\rangle$$

where $|\tilde{\varphi}\rangle$ is a state that by measurement gives a good estimation of the phase φ.

8.3. Performance evaluation

In the ideal case considered above, it was considered that the desired phase φ can be represented exactly on t bits. In the generic case, a real number can only be represented on a number of fixed bits by using an error margin. And the algorithm for phase estimation produces a good approximation, with high probability, of the real value in this generic case.

Define b as the best maximal integer approximation of φ on t bits. So, b is an integer such that $b/2^t$ is the best approximation of φ:

$$0 \leq b \leq 2^t - 1, \qquad \varphi - \frac{1}{2^t} \leq \frac{b}{2^t} \leq \varphi$$

By defining the error in approximation as $\delta \equiv \varphi - b/2^t$, it satisfies:

$$0 \leq \delta \leq \frac{1}{2^t}$$

It is proved that the algorithm for phase estimation produces a small δ with high probability. For obtaining φ aproximatted on n bits with a probability of success of at least $1 - \varepsilon$, the circuit needs a number of qubits equal to:

$$t = n + \left\lceil log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$$

8.4. Quantum algorithm for phase estimation

Algorithm: Quantum phase estimation
Input:

1. A black box that implements $C^1(U^j)$, for any j integer
2. A quantum register on $t = n + \left\lceil log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$ qubits initialized to $|0\rangle$
3. A quantum register prepared in state $|u\rangle$, a singular state of U, with corresponding singular value $e^{2\pi i\varphi_u}$

Output:

1. Integer on n bits $\tilde{\varphi}_u$, an approximation on n bits of φ_u

Running time:

1. $O(t^2)$ unitary operations and one call to each $C^1(U^j)$ black box.
2. The probability of obtaining the correct answer is at least $1 - \varepsilon$.

Procedure:

1. $\|0\rangle\|u\rangle$	initial state
2. $\longrightarrow \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} \|k\rangle\|u\rangle$	building superposition
3. $\longrightarrow \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} \|k\rangle U^k\|u\rangle$	application of black boxes
4. $\longrightarrow \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} e^{2\pi ik\varphi}\|k\rangle\|u\rangle$	result after applying black boxes
5. $\longrightarrow \|\tilde{\varphi}_u\rangle\|u\rangle$	applying QFT
6. $\longrightarrow \tilde{\varphi}_u$	measurement of the control register

9. Applications for quantum algorithms

9.1. Order finding and factorization

The procedure for phase estimation can be used for solving two concrete problems: order finding and numbers factorization. Actually these two problems are algorithmically equivalent. These algorithms are important for at least three reasons:

1. The most important, they represent the uncontestable proof that quantum computers are intrinsically more efficient in solving some problems, and therefore they are used to challenge the strong variant of the Church-Turing conjecture.
2. They are both theoretically and practically valuable. This encourages the search for other algorithms that could bring the same benefits.
3. Purely from a practical perspective, they could be used, as soon as they are physically implemented, to challenge some cryptographic systems based on public keys.

9.2. Order finding

The problem is formulated in the numbers theory as: for positive integer numbers $x < N$ that are co-prime, i.e. $\gcd(x, N) = 1$, the order of x modulo N is defined as the smallest positive integer r, such that $x^r = 1(\mathrm{mod}\ N)$. In number theory, this number is proved to exist and $r \leq N$. The problem can be more generically expressed in the language of cyclic finite groups.
This problem is believed to be difficult to resolve on classical computers, i.e. there is no known classical algorithm for solving this using polynomial resources $O(L)$, where L is the number of bits necessary to store the input, $L \equiv \lceil \log_2(N) \rceil$.
At an abstract layer, the quantum algorithm for solving this problem is actually the algorithm for phase estimation, applied to operator U that replaces the black box:

$$\begin{cases} U|y\rangle \equiv |xy(\mathrm{mod}\ N)\rangle & \forall y \in \{0,1\}^L, \quad 0 \leq y \leq N-1 \\ U|y\rangle \equiv |y\rangle & \forall y \in \{0,1\}^L, \quad N \leq y < 2^L - 1 \end{cases}$$

Because x and N are co-prime it follows that:

$$\exists x^{-1} \in \mathbb{N}, \quad x^{-1} \leq N, \quad xx^{-1} = x^{-1}x = 1(\mathrm{mod}\ N)$$

And therefore the operator defined above is unitary, with its adjoint operator being:

$$\begin{cases} U^\dagger|y\rangle \equiv |x^{-1}y(\mathrm{mod}\ N)\rangle & \forall y \in \{0,1\}^L, \quad 0 \leq y \leq N-1 \\ U^\dagger|y\rangle \equiv |y\rangle & \forall y \in \{0,1\}^L, \quad N \leq y < 2^L - 1 \end{cases}$$

The algorithm for phase estimation also requires a singular state for the respective operator. The s singular states for the operator defined above are:

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi i s k}{r}} |x^k(\mathrm{mod}\ N)\rangle \right] \quad \forall s \in \mathbb{N}, \ 0 \leq s \leq r-1$$

because:

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi i s k}{r}} U|x^k(\mathrm{mod}\ N)\rangle \right] = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi i s k}{r}} |x^{k+1}(\mathrm{mod}\ N)\rangle \right]$$
$$= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi i s (k+1)}{r}} e^{\frac{2\pi i s}{r}} |x^{k+1}(\mathrm{mod}\ N)\rangle \right]$$

$$= e^{\frac{2\pi is}{r}} \frac{1}{\sqrt{r}} \left\{ \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi isk}{r}} |x^k (\text{mod } N)\rangle \right] - e^{\frac{-2\pi is0}{r}} |x^0 (\text{mod } N)\rangle + e^{\frac{-2\pi isr}{r}} |x^r (\text{mod } N)\rangle \right\}$$

$$= e^{\frac{2\pi is}{r}} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left[e^{\frac{-2\pi isk}{r}} |x^k (\text{mod } N)\rangle \right] = e^{\frac{2\pi is}{r}} |u_s\rangle$$

So, to each singular state $|u_s\rangle$, its singular value is $e^{\frac{2\pi is}{r}}$.
For applying the quantum phase estimation algorithm, it is also needed to:

1. find an efficient procedure for implementing $C^1(U^{2^j})$, for any integer j.
2. find an efficient way of preparing singular state $|u_s\rangle$, with non-trivial singular value, or a superposition of such singular states.

If these conditions are satisfied, the order r can be computed as $r = \frac{s}{\tilde{\varphi}_s}$.
The first condition can be satisfied by considering a procedure known as modular exponentiation, which can implement the entire sequence of controlled operators $C^1(U^{2^j})$ called in the procedure for quantum phase estimation, by using $O(L^3)$ gates.
The first part from the algorithm for phase estimation computes the transformation:

$$|z\rangle|y\rangle \longrightarrow |z\rangle U^{z_t 2^{t-1}} U^{z_{t-1} 2^{t-2}} \dots U^{z_2 2^1} U^{z_1 2^0} |y\rangle$$
$$= |z\rangle \left| x^{z_t 2^{t-1}} x^{z_{t-1} 2^{t-2}} \dots x^{z_2 2^1} x^{z_1 2^0} y (\text{mod } N) \right\rangle$$
$$= |z\rangle |x^z y (\text{mod } N)\rangle$$

For achieving this, three major steps are required:

1.1. reversible compute function $f(z) = x^z (\text{mod } N)$ into a third register
1.2. reversible multiply modulo N this third register with the second register y
1.3. to keep the entire step reversible, bring the third register to its initial state

The first step can be further divided in smaller steps:

1.1.1. sequentially compute values $x^2 (\text{mod } N)$, $x^{2^2} (\text{mod } N)$,... $x^{2^j} (\text{mod } N)$,... $x^{2^{t-1}} (\text{mod } N)$, by successively squaring
1.1.2. apply successive multiplications according to the binary representation of z: $x^z (\text{mod } N) = \left(x^{z_t 2^{t-1}} (\text{mod } N)\right)\left(x^{z_{t-1} 2^{t-2}} (\text{mod } N)\right) \dots \left(x^{z_1 2^0} (\text{mod } N)\right)$

For evaluating the complexity of modular exponentiation, note that the third step is the reverse of the first one. Considering the multiplication modulo N is done with the classical algorithm of complexity $O(L^2)$, there is a total polynomial complexity [18]:

$$cost(1.1.) + cost(1.2.) + cost(1.3.) = 2cost(1.1.) + cost(1.2.) =$$
$$= 2cost(1.1.1.) + 2cost(1.1.2.) + cost(1.2.) + cost(1.3.)$$
$$= 2(t-1)O(L^2) + 2(t-1)O(L^2) + O(tL) = O(tL^2)$$

For satisfying the second condition, note that the definition of $|u_s\rangle$ can't be used for computing it, because this would require knowing r. But it is possible to use a superposition of singular states:

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = \frac{1}{\sqrt{r}}\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\left(\sum_{k=0}^{r-1}\left[e^{\frac{-2\pi isk}{r}}|x^k(\mathrm{mod}\ N)\rangle\right]\right) = \frac{1}{r}\sum_{k=0}^{r-1}\left(\sum_{s=0}^{r-1}\left[e^{\frac{-2\pi isk}{r}}|x^k(\mathrm{mod}\ N)\rangle\right]\right)$$
$$= \frac{1}{r}\sum_{k=0}^{r-1}\left(\left(\sum_{s=0}^{r-1}e^{\frac{-2\pi isk}{r}}\right)|x^k(\mathrm{mod}\ N)\rangle\right)$$

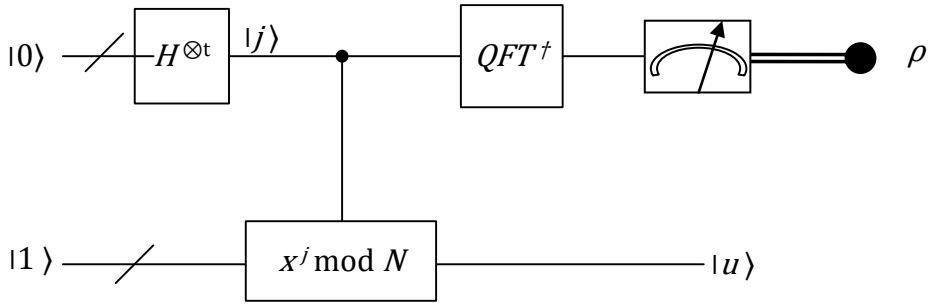
The interior sum is a geometrical progression with rate $e^{\frac{-2\pi ik}{r}}$:

$$\sum_{s=0}^{r-1}e^{\frac{-2\pi isk}{r}} = \frac{1-e^{\frac{-2\pi ik}{r}r}}{1-e^{\frac{-2\pi ik}{r}}} = \begin{cases}0, & 1\le k\le r-1\\ r, & k=0\end{cases}$$

So, the superposition becomes, which is very easy to prepare into the data register:

$$\frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$$

In the control register, it is necessary to use $t = 2L+1+\left\lceil log\left(2+\frac{1}{2\varepsilon}\right)\right\rceil$ qubits. With this data, the algorithm for phase estimation is applied for each $0\le s\le r-1$, obtaining an approximation $\tilde{\varphi}_s = \frac{s}{r}$ with an accuracy of $2L+1$ bits and the probability of at least $\frac{1-\varepsilon}{r}$. Schematically, the entire quantum circuit is represented as.



9.2.1. Result interpretation from the quantum phase estimation algorithm

The reduction of the order finding problem to the problem of the phase estimation is complete only if it is possible to efficiently obtain the desired result r, from the result returned by the algorithm of quantum phase estimation: $\varphi \cong \frac{s}{r}$. This result is returned as a number on $2L+1$ bits. It is very important to note that it is a priori known that this is a rational number. So, if it is possible to compute this fraction, the closest to φ, it would be then possible to find the desired result r.

Remarkably, such a polynomial algorithm for computing the closest fraction is known: continued fractions algorithm. Its base idea uses the representation of real numbers as continued fractions:

$$[a_0,\dots,a_M] \equiv a_0 + \cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{\dots+\cfrac{1}{a_M}}}}$$

where $a_0,\dots,a_M$ are all positive integers, with a_0 being allowed to be 0 so that sub-unitary numbers are also represented. The complexity of this algorithm is polynomial: if x, y are represented on L bits, the algorithm performs $O(L)$ divisions each division with $O(L^2)$ operations. So, it totals $O(L^3)$ operations.

Formally, the continued fractions algorithm is defined by the following theorem which can be proved by induction:

Theorem: Let $a_0, \dots, a_n$ a sequence of positive numbers. Then $[a_0, \dots, a_n] = \frac{p_n}{q_n}$, where p_n and q_n are positive integers defined recursively as:

$$\begin{cases} p_0 = a_0, & p_1 = 1 + a_0 a_1, \;\; p_n = a_n p_{n-1} + p_{n-2} \\ q_0 = 1, & q_1 = a_1, \;\; q_n = a_n q_{n-1} + q_{n-2} \end{cases}$$

Corollary: $q_n p_{n-1} - p_n q_{n-1} = (-1)^n$ for $n \geq 1$ and $\gcd(p_n, q_n) = 1$.

For being able to apply the continued fractions algorithm, it is important to note the following theorem:

Theorem: If $\frac{p}{q}$ and x are rational numbers where $\left|\frac{p}{q} - x\right| \leq \frac{1}{2q^2}$, then $\frac{p}{q}$ is a convergent from x's continued fraction.

Because, according to the phase estimation algorithm, the result after the final measurement φ is an approximation of number $\frac{s}{r}$, with an accuracy of $2L + 1$ bits, it results that

$$\left|\frac{s}{r} - \varphi\right| \leq \frac{1}{2^{2L+1}} = \frac{1}{2(2^L)^2} \leq \frac{1}{2r^2}$$

Therefore, $\frac{s}{r}$ is a convergent from φ's continued fraction. So, $\frac{s}{r}$ can be computed in $O(L^3)$ operations, by using the continued fractions algorithm.

9.2.2. The performance of the order finding algorithm

This is a probabilistic algorithm which could return a wrong result for two reasons:

1. The procedure for phase estimation could produce an erroneous approximation of $\frac{s}{r}$. This is mitigated by increasing the probability through the increase of the control register.
2. It is possible that s and r are not co-prime. In this case, r', the integer returned by the continued fractions algorithm is only a factor of the desired r, instead of being r itself.

Fortunately, there are three ways for avoiding or mitigating the second cause:

2.1. If define $\pi(r)$ the number of prime numbers less than r: $\pi(r) = \|\{s,\ s < r,\ s\ prime\}\|$, then $\frac{r}{2\log(r)} \leq \pi(r)$. And from this, if $0 \leq s < r$ is a random number, the probability of s being prime is:

$$p(s\ prim) = \frac{\pi(r)}{r} \geq \frac{1}{2\log(r)} > \frac{1}{2\log(N)}$$

So, by repeating the algorithm $2\log(N) = 2L$ time, there is a high chance to get s prime, and hence s and r to be co-prime.

2.2. If the desired number r is not returned, and the algorithm returns r' instead, then r' is a factor of r, if $s \neq 0$. But $s = 0$ has the probability $\frac{1}{r} < \frac{1}{2}$. So, this case can be mitigated by repeatedly running the algorithm a few times. So, if r' factor of r was

returned, one can substitute $x \leftarrow x' \equiv x^{r'} (\text{mod } N)$ and the order of the new x' is $\frac{r}{r'}$ because

$$(x')^{\frac{r}{r'}} = x^{r'\frac{r}{r'}} = x^r = 1(\text{mod } N)$$

This procedure may be necessary to be repeated at most $\log_2(r) \in O(L)$ steps.

2.3. The third modality is more efficient as it requires only a constant time. The idea is to repeat twice both the procedure of phase estimation and the procedure for continued fractions. So, the first run returns r_1' and s_1', and the second returns r_2' and s_2'. If s_1' and s_2' are co-prime, r can be computed in $O(L^2)$ using formula:

$$r = \frac{r_1' r_2'}{\gcd(r_1', r_2')}$$

The probability of s_1' and s_2' being co-prime:

$$p(gcd(s_1', s_2') \equiv 1) = 1 - \sum_{q=2,\ q\ prime}^{r} p(q|s_1')p(q|s_2') \geq \frac{1}{4}$$

So, the probability of returning the desired order r is at least $\frac{1}{4}$.

The time complexity is evaluated by considering each step in the algorithm. Hadamard transformations require $O(L)$ gates; QFT requires $O(L^2)$ gates. The biggest cost comes from the modular exponentiation step: $O(L^3)$ gates. The continued fractions algorithm adds other $O(L^3)$ gates. So, the total complexity is $O(L^3)$ for obtaining r'. Using the third method (see 2.3 above) it may be required to run the whole procedure a few times to obtain r from r'.

Algorithm: Quantum order finding
Input:

(1) Black box $U_{x,N}$ that implements transformation $|z\rangle|y\rangle \longrightarrow |z\rangle|x^z y(\text{mod } N)\rangle$ where x is co-prime co-prim to N, and is on L bits.

(2) $t = 2L + 1 + \left\lceil log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$ qubits initialized to $|0\rangle$

(3) L qubits initialized to $|1\rangle$.

Output:

The smallest integer $r > 0$ such that $x^r = 1 \ (\text{mod } N)$

Execution time:

$O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1. $|0\rangle|1\rangle$ — Initial state
2. $\longrightarrow \frac{1}{2^{\frac{t}{2}}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ — Create superposition

3. $\longrightarrow \frac{1}{2^{\frac{t}{2}}} \sum_{j=0}^{2^t-1} |j\rangle \left|x^j (\text{mod } N)\right\rangle$

$$\cong \frac{1}{\sqrt{r} 2^{\frac{t}{2}}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{\frac{2\pi i s j}{r}} |j\rangle |u_s\rangle$$

Apply operator $U_{x,N}$

4. $\longrightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\frac{\tilde{s}}{r}\right\rangle |u_s\rangle$ Apply inverse QFT on control register

5. $\longrightarrow \frac{\tilde{s}}{r}$ Measurement on the control register

6. $\longrightarrow r$ Apply the continued fractions algorithm

9.3. Applications: factoring natural numbers

Problem: given a positive number N, find its prime factors. This problem is actually equivalent to the order finding problem, i.e. the algorithm for order finding can be efficiently transformed into a factorization algorithm.

9.3.1. Steps for factorization

For reducing this to the order finding algorithm, there are two main steps:

1. A factor of N can be determined by finding a non-trivial solution $x \neq \pm 1 (\text{mod } N)$ to the equation $x^2 = 1 (\text{mod } N)$
2. Prove that a random number y, co-prime with N is very likely to have an order r which is an even number that satisfies $y^{\frac{r}{2}} \neq \pm 1 (\text{mod } N)$.

Then combine these steps by considering $\equiv y^{\frac{r}{2}} (\text{mod } N)$. These two steps are based on the following two theorems.

Theorem: Let N integer. If $1 < x < N - 1$ is an integer, a non-trivial solution of $x^2 = 1 (\text{mod } N)$, then either $\gcd(x - 1, N)$ or $\gcd(x + 1, N)$ is a non-trivial factor of N.

So, if N is on L bits, knowing x, one can obtain a non-trivial factor of N by performing the $O(L^3)$ steps from the Euclid's algorithm.

Lemma: Consider $p > 2$ prime. Let 2^d the biggest power of 2 that divides $\varphi(p^\alpha)$. If x is a random number in group $\mathbb{Z}_{p^\alpha}^* = \{x \in \mathbb{Z}_{p^\alpha}, \text{cmmdc}(x, p) = 1\}$, let r be its order. Then 2^d divides r with probability of exactly $\frac{1}{2}$.

Theorem: Let $N = p_1^{\alpha_1} \ldots p_m^{\alpha_m}$ be the factoring of an odd number. If x is a random number in $\mathbb{Z}_N^*$, using a uniform distribution, and if r is the order of x modulo N; then the probability

$$p\left(r = 2k \wedge x^{\frac{r}{2}} \neq -1 (\text{mod } N)\right) \geq 1 - \frac{1}{2^{m-1}}$$

9.3.2. The quantum factoring algorithm

Almost all the steps can be efficiently implemented also on a classical computer. The only exception is the subroutine of order finding. By calling the algorithm repeatedly, all the factors can be found.

Algorithm: Reducing factorization to order finding

Input:

Integer N

Output:

A non-trivial factor of N

Execution time:

$O((\log_2 N)^3)$. Succeeds with probability $O(1)$.

Procedure:

1. if N even, return 2
2. for each $2 \leq b \leq \lceil \log_2 N \rceil$
   2.1. if $N = a^b$, return a
3. choose random $1 \leq x \leq N - 1$
4. if $f = \gcd(x, N) > 1$ then return f
5. call finding order r of x, modulo N
6. if r even and $x^{\frac{r}{2}} \neq -1 (\mathrm{mod}\ N)$ then compute $f_1 = \gcd(x^{\frac{r}{2}} - 1, N)$ $f_2 = \gcd(x^{\frac{r}{2}} + 1, N)$
7. if $f_1 \neq 1$ return f_1
8. if $f_2 \neq 1$ return f_2
9. return error

The first two steps return a factor or make sure N has at least two factors. They require $O(L^3)$ steps. The next two steps return a factor or a random number in $\mathbb{Z}_N^*$, with complexity $O(L^2)$. The last three steps apply the two theorems above to find a factor. They succeed with probability at least $\frac{1}{2}$.

These are the main applications with great practical applicability. But by using the quantum Fourier transform a much larger set of problems can be addressed. The most general problem that includes, as particular cases, all the applications exponentially efficient of the quantum Fourier transform, is the hidden subgroup problem. Intuitively, this problem can be thought of as a generalization of the problem of finding the period of a periodical function, when its source and the target domains are very complicated. Using the algebraic language from the groups' theory, this problem can be expressed in a general way as [41]:

Let a function f defined on a group G finitely generated, taking values from a finite set X such that f is constant on any coset defined by an unknown subgroup K, those constant values being different. If there is a quantum black box that implements the unitary transformation $U|g\rangle|h\rangle = |g\rangle|h \oplus f(g)\rangle$, where $g \in G$, $h \in X$ and $\oplus$ is a binary operation in X, carefully chosen; find a set that generates K.

9.4. Languages for quantum programming

Experimenting with known quantum algorithms and, even more, designing new ones is a difficult enterprise if the proper programming languages and development tools are not in place. Quantum circuits presented above are a great first step in presenting detailed aspects of quantum algorithms. Furthermore, they can be easily mapped onto specific quantum

computing hardware, after they are reduced to the respective set of elementary quantum gates. But in order to implement more advanced quantum programs, a higher level of abstraction is required.

9.4.1. Quantum programs

Lately, great progress has been made in abstracting away the quantum hardware by following a path that resembles in many aspects the approach taken for classical computing, decades ago. In classical terms, programming a computer basically means telling it the actions to perform and the data to be used as input for those actions, in a language that the computer understands. Therefore, leaving aside all the complexities related to specific computer architectures and different flavors of paradigms for programming languages, a classical computer program can be summarized as:

PROGRAM = DATA + INSTRUCTIONS

Here, data means all kinds of information that can be physically stored on the computer, or communicated to another computer, and that can be manipulated by the instructions statements. Usually, in its most basic form, data is represented by bits. The instructions are composed of a basic set (for example, logical and arithmetic instructions) and a set of control structures (such as conditionals, loops, jumps, etc.).
Things start to get more complicated as we consider a specific programming paradigm or specific computer hardware architecture. For example, if one assumes the existence of an instruction pointer, instructions become nothing more than a specific set of data.
This same basic scheme was extended to quantum programming world, by supplementing it with the elements specific to quantum aspects:

QUANTUM PROGRAM = QUANTUM DATA +
QUANTUM OPERATIONS +
INSTRUCTIONS

In the conceptual relation above, a quantum computer running such quantum program is assumed to be comprised of a quantum device, able to manipulate the quantum data – represented as qubits which can be addressed, by performing a predefined set of quantum operations, which come in two types:

- unitary operations, which transform quantum data by maintaining its quantum properties
- measurements, which inspect the quantum data and turns it into classical data

For practical implementations, the set of predefined unitary operations has to have the following properties:

- be finite set, so that it can be implemented by physical hardware
- be an universal set, so that all the possible quantum programs can be created
- each operator acts on a finite Hilbert space

One example of such predefined set of unitary operators can be formed from the quantum gates composing the Shor basis: Hadamard, phase, CNOT, Toffoli. Note that this predefined set doesn't necessarily need to be a minimal set, but their number will most likely be kept low because of economical and practical considerations.
Then, this basic predefined set of unitary operators can be used to define increasingly complicated unitary operators, in the same way quantum gates are composed together into quantum circuits. Since for the time being all the known sets of operators which are both

finite and universal can only approximate some more complicated quantum operators, it follows that some operators won't have a precise implementation, but only an approximate one.
Measurements are usually considered to be performed using projection operators built around the computational basis states. The outcomes of the measurements are expressed therefore as classic bits, which can be read out from the quantum device.

9.4.2. Quantum programming languages

As in the classical case, the first level of abstraction used in programming is the assembler language. And as for the classical case, the quantum computation model will also have to be built around a specific architecture of the respective quantum machine. Yet, it is not necessary to specify the underlying quantum hardware.
In the same way the classical assembler (and even higher level languages) are built around abstract concepts such as stacks, heaps, etc., the quantum assembler must consider a set of abstract concepts.

Quantum circuits

Quantum circuits were the first quantum computation model introduced. This model assumes the following ingredients used in computation:

- an input device that can prepare initial set of qubits, to start the computation
- a finite set of quantum gates, acting on a finite number of qubits, that are connected in serial or in parallel, forming an directed acyclic graph – a quantum circuit
- a measurement device that outputs a sequence of classical bits that can be read off. These measurements can be always performed at the exit of the quantum circuit, after all the gates finished processing the quantum state.

Quantum Turing machine

This model, which adds the quantum elements to the standard probabilistic Turing machine model, is particularly useful for discussing complexity classes, but it is difficult to use it to build quantum algorithms or bigger quantum programs. This model assumes the well-known Turing hardware architecture based on an infinite, one dimensional band that contains characters from a finite set and that can be moved left or right, and a device that can read or write the current character.
The main differences between the probabilistic and the quantum machines are:

- the transition function for probabilistic machine gives probabilities, whilst the transition function for the quantum machines gives complex numbers, whose module squared represent probabilities.
- at each step, the probabilistic machine picks randomly one transition, whilst the quantum machine carries out all the possible transitions in parallel.

Quantum Random Access Memory Model

This is the quantum computing model that is most suited to programmers, because it offers a programming paradigm closer to the classical one and allows easy specification of quantum algorithms and even higher level quantum computing languages. This model assumes a hardware architecture that is basically a quantum extension of the classical one. The hardware is composed of:

- a classical device that:
    - passes classical bits to the quantum device
    - performs classical computations
    - reads measurement results (i.e. classical bits) from the quantum device

- a quantum device that:
    - takes a set of bits and initializes a set of qubits with the corresponding computational basis state
    - performs the specified basic unitary transformation on the current quantum state, or on a subset of it
    - performs the measurement using projection operators on the specified qubits
    - returns measurement results as a set of classical bits to the classical device

This model is closely related to a typical programmer's approach to solving a problem: implement the problem classically, as much as possible, and each time the quantum power is needed, delegate the task to the quantum device, and read the results of the final measurements. Then, as necessary continue in classical mode, or delegate again to the quantum device.
The communication between the two devices, the classical one and the quantum one, has to be implemented very carefully, by the means of a quantum hardware interface. This interface would have to ensure that the processing happening inside the quantum device is not affected in an unwanted way by the classical device, causing the quantum states to collapse when they shouldn't or become entangled with the environment.
As before, these basic unitary transformations have to be only from a predefined finite set, which is universal for quantum computation. Usually this set is chosen so that it matches the underlying quantum hardware capabilities.
So, there are a few types of quantum instructions that this model has to provide:

- initialize a quantum register with a given computational basis state. There is no need to provide a more complicated initialization instruction, which would initialize the quantum register with a superposition of computational basis states. This is because such superpositions can be obtained by applying unitary transformations.
- select a subset of qubits from a quantum register in order to use them for further processing, either unitary transformations or measurements
- apply a unitary transformation on a quantum register (or a subset of its qubits)
- compose two unitary transformations, i.e. execute them sequentially
- tensor two unitary transformation, i.e. execute them in parallel
- measure a quantum register (or a subset of its qubits) and put the results in classical register

For example, the following set of instructions builds an EPR pair and measures its first qubit, obtaining either 0 or 1, with equal probability:

```
qbit q[ 2 ] = {0, 0};              // initialization
let U1 = tensor( H, I2 );          // tensor product
let Epr = concat( U1, CNOT );      // operators composition
Epr( q );                          // applying operators
bit r[ 1 ] = measure( q[ 0 ] );    // measurement
```

It has to be noted that copying qubits it is not permitted, so there is no point to provide an instruction for assigning one set of qubits to another.
Also, there is no need to provide a special set of quantum instructions for controlling the flow of execution in the quantum device, such as for example a quantum if-then-else type of statement. This is because of two reasons:

- any such conditional statements can be implemented in the quantum world by using conditional operators
- conditional statements can be implemented by using only classical bits obtained by performing measurements whenever required

Assembly language based on strings manipulation

This low level language is a good compromise between the Turing machine and QRAM. It is easy to use with most real world algorithms, and it abstracts the hardware internals well enough to make it useful in theoretical analysis of complexity classes. In this model, define a fixed, finite character set C. Let there be an (possible infinite) number of storage locations, each labeled by some string over C. Initially, all the locations are set to empty strings. If s is a string over C, the usual C++ notation *s is used to represent the string stored at address s. The string *s is also composed only from characters in C.

For the quantum part, define the following fixed objects [40]:

- a finite-dimensional Hilbert space H.
- a unit vector $|\psi_0\rangle$ in H
- the Hilbert space H can be tensored with itself for any finite number of times, each instance in this tensor product is labeled by strings over C. So, such a tensor product will be represented as $H_{s_1} \otimes H_{s_2} \otimes \ ... \otimes H_{s_k}$.
- a finite set of unitary operators, each of which acts on some finite tensor product $H_{s_1} \otimes \ ... \otimes H_{s_k}$
- a finite set of projection operators, each of which acts on some finite tensor product $H_{s_1} \otimes \ ... \otimes H_{s_k}$

Each operator from the sets above (either unitary or projection) are labeled by a string over C. So, the unitary operators will be represented as U_s and the project operators will be represented as P_s. At any point of the program execution, the state of the quantum system is represented by a state $|\psi\rangle$ in $H_{s_1} \otimes \ ... \otimes H_{s_k}$.

A quantum assembly language can then be composed from the following commands.

- Commands for classical operations:
  - InputTo *s: allows the user to enter any string over C, which is then placed in the memory location pointed to by s
  - OutputFrom *s: returns to the user the string from memory location s
  - AppendTo x, *s: appends character x to the string stored in location s
  - DeleteLast *s: deletes last character (if any) from the string stored in location s
  - ConditionalJump x, *s, n: if the last character (if any) of the string stored in location s is equal to x, then skip forward n program lines, if n is positive, or skip backward -n program lines, if n is negative. Otherwise, do nothing and continue as usual. If the line indicated to by n doesn't exist, then do nothing and continue as usual.
- Commands for quantum operations:
  - Apply s, s1, s2, … sk: this command actually performs several steps:
    - make sure the current state space contains all the H factors labelled by the strings in the command $H_{s_1} \otimes H_{s_2} \otimes \ ... \otimes H_{s_k}$. If it doesn't, then:
      - expand the current state space with the missing H factors

- tensor to the current state $|\psi\rangle$ the corresponding number of $|\psi_0\rangle$, obtaining a new $|\psi\rangle$.

- if the current state space contains tensor factors that are not among the specified s1, s2, … sk, those subspaces will remain unchanged. That is, U_s is tensored with the corresponding number of identity operators over H, obtaining a new U_s.
- apply unitary operator U_s to the current state $|\psi\rangle$, obtaining a new state $|\psi\rangle$

o Observe s, s1, s2, … sk, *s': this command too actually performs several steps:

- make sure the current state space contains all the H factors labelled by the strings in the command $H_{s_1} \otimes H_{s_2} \otimes \ ... \otimes H_{s_k}$. If it doesn't, then:
  - expand the current state space with the missing H factors
  - tensor to the current state $|\psi\rangle$ the corresponding number of $|\psi_0\rangle$, obtaining a new $|\psi\rangle$.
- if the current state space contains tensor factors that are not among the specified s1, s2, … sk, those subspaces will remain unchanged. That is, U_s is tensored with the corresponding number of identity operators over H, obtaining a new U_s.
- apply projection operator P_s to the current state $|\psi\rangle$, obtaining a string of length k, containing only 0 or 1 characters.
- store the resulted string at the location pointed by string s'.

This language is closely related to the QRAM model as it also offers both classical and quantum commands. In this model, it is considered that a program computes a problem π if, for any input string s, the program run on that string and the probability of obtaining the correct string is greater than any of the probabilities of obtaining a different string. Furthermore, the probability that the program running never halts must be 0.

9.4.3. High level quantum programming languages

The models of quantum computation described above represent a first level of abstraction that can be used to develop quantum algorithms or to simulate them on classical machines. However, these languages are quite limited in their capabilities. For example, the QRAM model operates only with arrays of qubits, whilst the model based on strings manipulation operates only with strings. As with classical computing, there is an obvious need to provide built in data types offering a higher level of abstraction: structures, classes, functions, etc. The vast majority of current classical programs require these kinds of types, and there is no reason to believe the quantum programs wouldn't.
When defining quantum programming languages at a higher level, there are a few design decisions that have to be made:

- choosing between one of two options:
  - a language that builds on top of a classical language, probably an existent one.

This classical language would be augmented with the requirements for quantum programming. The programmer developing in this language will use the classical features of the language to implement the classical aspects of the program, and delegate only parts of the implementation to quantum specific language features.

This approach will usually imply that the program flow control (if, while, jump, etc.) will be realized using only classical language features, eventually using measurements when necessary.

- a self-contained quantum language, a language that contains only quantum programming feature. In such a program, programs must be implemented based on the reversible computing approach, using the quantum scratch pad when necessary to accommodate irreversible classical functions. Such language is possible because any classical Turing machine can be simulated by a quantum Turing machine, but is more difficult to use by programmers who are not use to approaching problems from this perspective.

- choosing the programming paradigm: imperative, functional, pattern matching.

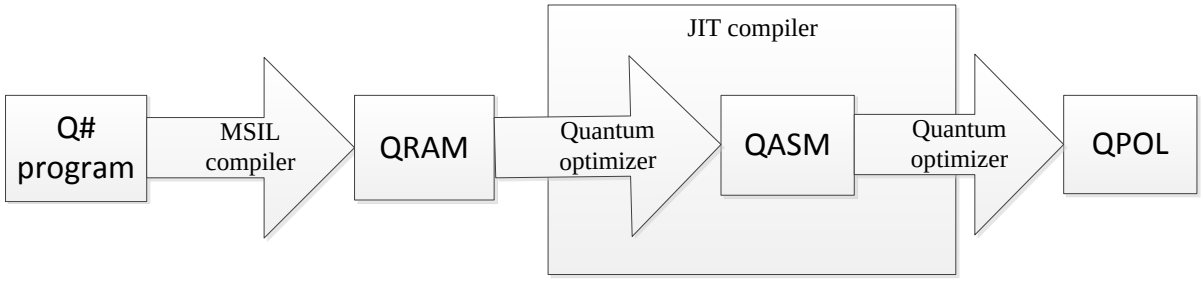
Several such high level programming languages for quantum computing have been defined. Some of them are using the imperative paradigm, like QCL [55] – which is also self-contained, or Q [14] – which was built as an extension to C++. Other approaches were based on functional programming paradigm, like for example QFC [61] – which uses classical flow control, or QML – where both data and control are quantum.
Yet, in classical programming languages the programming paradigms are starting to merge together. A very good example of this is offered by the recent evolutions of the languages in the .Net space. C# started as an imperative language but then, from version 3.0 onwards, by the addition of the LINQ extension it started providing features specific to functional programming, such as function closers, meta-programming patterns (i.e. programs that manipulate other programs), etc. On top of this, a standalone functional programming language was added to .Net: F#. In this way, by using the easy interoperability between C# and F# provided by the .Net framework, both imperative and functional paradigms can be merged in the same program.
Considering these aspects, a promising approach for quantum programming would be to define a language (Q#) based on top of C#. This language:

- would use all the current C# language features for classical computing (both functional and imperative)
- provides an extra layer of features for quantum programming. Since these features are based around applying operators, the most obvious approach would be to use a sort of functional programming syntax.

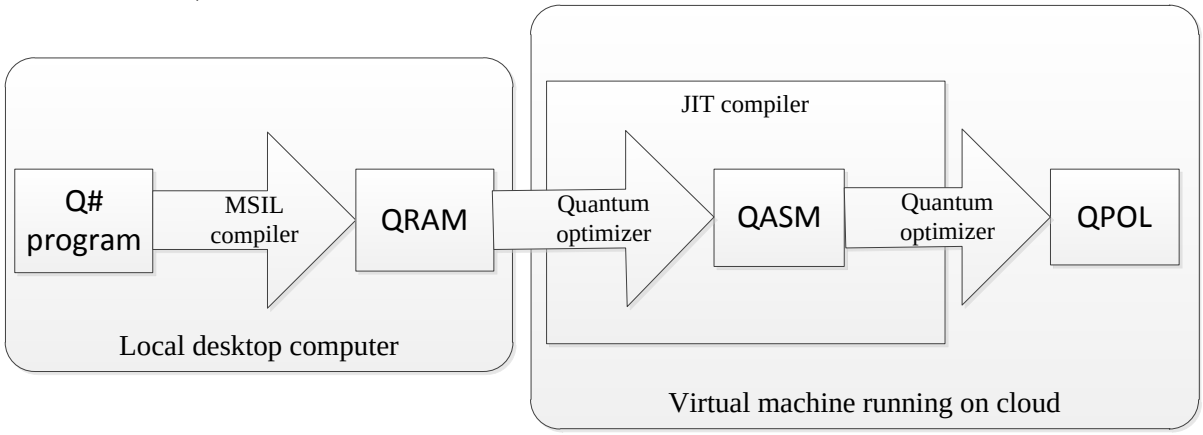
Another advantage of taking this approach would be provided by the way .Net languages are compiled. They are compiled in two steps: the first step takes the managed program and translates it into MSIL (Microsoft independent language) code. Then, the second step, which sometimes can even happen at runtime by the means of JIT (just-in-time) compiler, is the conversion to native, executable machine code. A C# based quantum programming language extension would work very well in this framework. The quantum programming aspects of a program will become quantum specific MSIL code – for example, a variant based on the QRAM model. Then, the JIT compiler would need to convert these MSIL instructions to the quantum hardware specific machine code, further optimizing it in the process.
The compilation process is schematically depicted below, where [69]:

- QRAM is the intermediate, low level quantum language, like the one described above
- QASM (Quantum Assembly Language) represents a low level quantum circuits representation, an optimized quantum circuit composed only of gates from the chosen universal set of elementary quantum gates; for example, CNOT, Hadamard, phase, Toffoli.

- QPOL (Quantum Physical Operations Language) represents a physical-language representation with technology specific parameters.



There is also another advantage of using a .Net based quantum programming language: integration with cloud computing provided by the Azure programming framework. When using quantum programs to perform simulations on classical computers or when using quantum programs to run them on quantum devices, in both these cases the required computing machines will need a very high spec set of requirements. Quantum computers are currently in prototype phase, they run only in specific, very closely controlled laboratory environments and for the foreseeable future, they will probably stay like this. On the other hand, classical computers intended to perform quantum simulations also require a huge amount of processing power as the quantum state spaces increases exponentially with the dimension of the input, so they are most likely super-computers.
Therefore, it makes very much sense that these machines to be exposed to the outside world of quantum programmers via cloud computing architectures (which can be based either on .Net or on Java).



Using the Azure cloud computing architecture with .Net framework, the compilation and the running process is represented above.
Considering the same EPR pair generation example as above, such a Q# program would look very similar to a C# program, with LINQ extension:

```
qbit q[ 2 ] = {0, 0};                          // initialization
QApply( q => CNOT( H( q[ 0 ] ), q[ 1 ] ) )     // define and apply unitary operators
bit r[ 1 ] = QObserve( q[ 0 ] );               // apply measurement operator
```

10. Contributions and further research

10.1. Contributions

In chapter 2. the author conceived a few demonstrations for some known examples that prove the power of quantum computing and quantum information processing, explaining how they are deeply linked to physical processes.
The author analyzed Deutsch-Jozsa problem and its generalization to Boolean functions extended to n – dimensional finite spaces, and conceived a classical probabilistic algorithm for solving this problem. The performances of quantum algorithm and the classical probabilistic ones were analyzed and compared, proving the increased time efficiency of the quantum algorithm.
The author also provided a proof that the super-dense coding protocol is secure, meaning that a third party intercepting the qubit being sent can infer nothing about the classical information to be transmitted.
In chapter 3. a set of theorems together with their formal demonstrations, designed by the author, were analyzed in order to justify the graphical representations of qubits using three dimensional unit spheres and to provide for the quantum operations modeling through three dimensional rotations, only around the coordinate axes. The respective demonstrations are constructive, and therefore the provided formulae can be used to simulate some specific quantum computing operators by using a graphical processing integrated circuit or graphical modeling software. Graphical representations based on geometrical transformations are a very expressive method of simulating the operations performed on quantum information processing systems.
Therefore it was proved there is an interesting conceptual relationship between the spherical rotations on one hand, and the one qubit unitary operations on the other.
Considering a single qubit system, the proofs given in this paper justify the correspondence between a class of operations that modify the qubit state and some geometrical transformations on the Bloch sphere. The single qubit operations are expressed by the exponentiation of Pauli operators, whereas the corresponding geometrical transformations are rotations on the Bloch sphere around the coordinate axes.
Chapters 4. and 5. present in detail the theory of quantum circuits together with a few quantum circuits designed by the author:

- a circuit that offers a minimal implementation of the generic controlled operator on two qubits, this implementation using only one qubit gates and CNOT gates
- a circuit that implements the Fredkin gate using only one Toffoli gate and two CNOT gates
- a circuit that implements the Fredkin gate using only six gates, each on two qubits
- a circuit that implements the generalized Toffoli gate, without using working qubits. This circuit has a polynomial complexity of the second grade
- a circuit that implements a generic controlled operator on an arbitrary number of qubits, without using working qubits. This circuit too has a polynomial complexity of the second grade

Chapter 6. presents a rigorous analysis on the universality of quantum gates. The author analyzes the exact universality of infinite sets of gates on one qubit, and how complicated circuits can be decomposed into simpler ones. This chapter also includes the complexity evaluations, as conceived by the author.

Then, the author provides a proof for an approximate universality, with arbitrary small error, using a discrete set of quantum gates: Hadamard, phase, CNOT and Toffoli. Such a set of elementary quantum gates (named Shor's basis) is necessary for the approximation of the generic unitary operators, on any number of qubits. Since the proof is made by construction, a couple of circuits necessary for implementing some elementary unitary operators on one qubit are also provided.
In the last three chapters, a few known algorithms [16] are analyzed, under some new light; these algorithms are a proof for the increased efficiency of the quantum computers, with respect to the temporal complexity, as compared to the corresponding known algorithms from classical computing.
In the last chapter, the author provides an analysis of the current languages for quantum computing, both at low and high level. The author also proposes a new language, built around .Net framework architecture, which extends from the C# language and uses both imperative and functional programming paradigms. The architecture for the compiler of this new language is described in both local environments and on an architecture based on cloud computing.

10.2. Further research

Quantum computing and information processing is a relatively new domain [1], if we are to compare it to the previous computing methods, such as analogic one or the one based on the generalized Turing machine. Although there still are many unknowns in this new computer science area, and surprises may appear when they are least expected, there already are numerous research centers around the world that are trying to address this kind of problems. The current most difficult problems in this area are due to both the physical nature of the processes involved which are hard to control and investigate, and also to the technological limitations of building the necessary hardware. The relatively few, but fast increasing numbers of experimental implementations of physical computing machines based on the principles derived from quantum mechanics have to deal with hard problems raised by the necessity of manipulating matter at quantum scales. Although quite many experiments did successfully manage to achieve important results, by physically implementing some quantum algorithms, the respective machines require some very restrictive operational environments, available only in very sophisticated research labs. Furthermore, the size of the input data used is very small compared with the quantity of data processed by even ordinary personal computers.
There also are other kinds of limitations – due mainly to weakly software resources currently available to quantum computing [39]. Designing algorithms in quantum computing paradigm is still solely based on the inspiration of the people involved in the respective problem [55]. Yet, the tools aimed at supporting software development, compilers, interpreters or high level programming languages for such quantum machines, are already in research [51] [57]. To aggravate things, but also to make this subject even more interesting, because of the profound change in the theoretical computing paradigm, most of these software resources will have to be reviewed and most likely radically changed. If they will be based on one of the current software programming paradigms (procedural, functional, pattern matching, object oriented, etc.) is still open for debate. But, first steps in this direction have already been made, by using both procedural [45] and functional programming paradigms [49]. It would seem though that the functional programing paradigm, based on the lambda calculus, to offer a better perspective. This is because the quantum bit transformations that are applied to qubits as they pass through the quantum circuits, are expressed using operators, which conceptually speaking, are functions on state space.

Also, there already are software frameworks for simulating quantum computing machines on classic, Turing machines [39]; and this means that for investigating theoretical aspects of those algorithms designed for quantum computers, from a technical perspective, a personal computer is basically enough.
And last but not least, probably because we still miss a formal demonstration that this quantum approach will have profound and useful implications upon the way the information technology affects and influences other domains more close to day to day existence; all these make that the investments in such research to be quite small compared to somewhat similar domains, such us nano-technology, bio-technology and molecular computing.
It has been already demonstrated that such a quantum computer, built to a similar scale to a present, classical personal computer, will be capable to break any encryption code that relies on public key cryptography based on factorization of integers. But, obviously, there are not many people really wanting, or hoping for this to happen very soon. But there is also hope; in the area of data base unstructured searches [36], the quantum computing algorithms also offer a performance improvement [60]. Furthermore, in the area of scientific computing applications, where modeling and simulating physical processes that happen at a quantum scale, the quantum computing paradigm also offers an obvious advantage.
The computing model based on quantum circuits is equivalent to many other computing models, proposed previously, in the sense that these models require the same essential resources for the same kind of problems. For example, to illustrate this, one could question if by using a computing model based on quantum systems triplets (qutrits), instead of binary quantum systems (qubits), it is possible to gain some kind of computational advantage. Although from a practical point of view, such small advantages may exist, strictly theoretical speaking, the difference between these two models is practically negligible. This is because the computing model inspired by the quantum Turing machines, a generalization of the model based on the classical universal Turing machines, has been proved to be equivalent with the model derived from quantum circuits.
It is not yet obvious at all if the assumptions made within the theory of quantum circuits are totally justified by real-world physical laws. For example, the assumption related to the states space and to the way the initial states are designed, is just a matter of choice. In this model, the states space is considered to have a finite dimension. And one could rightfully ask if by switching to vector spaces having an infinite number of dimensions isn't possible to gain some kind of advantage.
Furthermore, the initial states of the qubits passing through the circuit are considered to be computational basis states. And it is a well-known fact that in nature, many physical quantum systems exist in highly entangled states. So, a second question that can be raised is if this kind of states could be used to benefit from some kind of computational advantage. Hence, because all these questions, the completeness of the computing model based on quantum circuits, and implicitly the completeness of the corresponding classical model is still debatable.
The area of quantum cryptography, whereas the quantum communication channels are used for the distribution of private keys used in the public key cryptography protocols, is probably the first one already open towards commercial applications.
Another area of promising research which is being pursued on multiple levels is related to the graphical computations and representations. Because of the strong, profound relationship between the qubit transformations and graphical rotations there are indications that graphical processing algorithms, and in general multimedia applications, may be good candidates for being ported to the quantum computing paradigm. But for this to happen in reality, the correspondence between single qubit operators and spherical rotations in tridimensional space will probably need to be extended to transformations on multiple qubits, perhaps by

considering rotations in multiple dimensions. The main issue with this extension is the fact that the simulation would have to properly handle entangled states.

11. Selective Bibliography

[1] Aaronson S., Gottesman D.: „*Improved Simulation of Stabilizer Circuits*", Physical Rev. A, vol. 70, no. 5, 2004

[2] Adleman L., Demarrais J., Huang M. A.: „*Quantum Computability*", SIAM J. Comp., 26(5), 1997

[3] Aharonov D., Kitaev A., Nisan N.: „*Quantum Circuits with Mixed States*", STOC, arXive e-print quant-ph/9806029, 1998

[4] Aharonov Y., Rohrlich D.: „*Quantum Paradoxes: Quantum Theory for the Perplexed*", Wiley-VCH, Weinheim, 2005

[5] Ambainis A.: „*Quantum walk algorithm for element distinctness*", SIAM J. Comput. 37/210, 2007

[6] Ambainis A., Kempe J., Rivosh A.: „*Coins make quantum walks faster uantum walk algorithm for element distinctness*", in Proceedings of the 16th Annual ACM SIAM Symposium on Discrete Algorithms, 2005

[7] Aspuru-Guzik A., Dutoi A., Love P.J., Head-Gordon M.: „*Simulated quantum computation of molecular energies*", Science 309/5741, 2005

[8] Bacon D., Dam W.V.: „*Recent Progress in Quantum Algorithms*", Communications of the ACM, Vol. 53, No. 02, 2010

[9] Barenco A.: „*A Universal Two-Bit Gate for Quantum Computation*", Proc. R. Soc. Lond. A, 1995

[10] Barenco A., Bennet C. H., Cleve R., DiVincenzo D. P., Margolus N., Shor P., Sleator T., Smolin J., Weinfurter H.: „*Elementary Gates for Quantum Computation*", Physical Review Letters 52, 1995

[11] Bennett C. H., DiVincenzo D. P.: „*Quantum Information and Computation*", Nature, 404, 2000

[12] Bennett C. H., Shor P. W.: „*Quantum Information Theory*", IEEE Trans. Inf. Theory, 44(6), 1998

[13] Bernstein E., Vazirani U.: „*Quantum Complexity Theory*", SIAM Journal on Computing, 26(5), 1997

[14] Bettelli S., Calarco T., Serafini L.: „*Toward an Architecture for Quantum Programming*”, The European Physics J. D, vol. 25, no. 2, pp. 181-200, 2003

[15] Bhatia R.: „*Matrix Analysis*”, Springer-Verlag, 1997

[16] Bransden B. H., Joachain C. J.: „*Introducere în Mecanica Cuantică*”, Editura Tehnică, 1995

[17] Braunstein S. L., Kimble H. J.: „*Teleportation of Continuous Quantum Variables*”, Physical Review Letters, 80, 1998

[18] Brînzănescu V., Stănăşilă O.: „*Matematici Speciale*”, Editura All, 1994

[19] Buhrman H., Špalek R.: „*Quantum verification of matrix products*”, in Proceedings of the 17th Annual ACMSIAM Symposium on Discrete Algorithms, 2006

[20] Chester M.: „*Primer on Quantum Mechanics*”, Dover Publications, 2003

[21] Childs A.M., Cleve R., Deotto E., Farhi E., Gutmann S., Spielman D.A.: „*Exponential algorithmic speedup by quantum walk*”, in Proceedings of the 35th ACM Symposium on Theory of Computing, 59–68, 2003

[22] Chuang I. L., Modha D.: „*Reversible Arithmetic Coding for Quantum Data Compression*”, IEEE Trans. Inf. Theory, 46(3):1104, 2000

[23] Cleve R., Ekert A., Macchiavello C., Mosca M.: „*Quantum Algorithms Revisited*”, Proc. R. Soc. London A, 454, 1998

[24] Cormen T. H.: „*Introduction to Algorithms*”, Second Edition, MIT Press, 2001

[25] Dasgupta S., Papadimitriou C. H., Vazirani U.: „*Algorithms*”, McGraw-Hill, 2006

[26] Deutsch D.: „*Quantum computational networks*”, Proc. R. Soc. Lond. A 425, 1989

[27] Deutsch D.: „*Quantum theory, the Church-Turing Principle and the universal quantum computer*”, Proceedings of the Royal Society of London A 400:97, 1985

[28] Deutsch D., Barenco A., Ekert A.: „*Universality in Quantum Computation*”, Proc. R. Soc. Lond. A, 1995

[29] DiVincenzo D. P.: „*Quantum Computation*”, Science, 270, 1995

[30] DiVincenzo D. P.: „*Two-bit Gates Are Universal for Quantum Computation*", Physical Review Letters A, 51(2), 1995

[31] Dragne L., Moldoveanu F., Soceanu A.: „*An Object Oriented Framework For Network Management*", 13th International Conference On Control Systems And Computer Science, Bucharest – Romania, 2001

[32] Dragne L.: „*A Component Based Hardware Abstraction Layer For Multimedia Home Platforms*", 14th International Conference On Control Systems And Computer Science, Bucharest – Romania, 2003

[33] Dragne L.: „*Modelling the Controlled Swap Gate with Quantum Circuits*", Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium, 2009

[34] Dragne L.: „*Geometrical Representation of Quantum Bit Operations*", U.P.B. Scientific Bulletin, Bucharest – Romania, 2010

[35] Dragne L.: „*Elementary Gates for Fault-Tolerant Quantum Computing*", Annals of DAAAM for 2010 & Proceedings of the 21th International DAAAM Symposium, 2010

[36] Ekert A., Jozsa R.: „*Quantum Algorithms: Entanglement Enhanced Information Processing*", Proc. R. Soc. Lond. A, 356(1743), 1998

[37] Fortnow L.: „*One Complexity Theorist's View of Quantum Computing*", Theoretical Computer Science, 292(3), 2003

[38] Farhi E., Goldstone J., Gutmann S.: „*A quantum algorithm for the hamiltonian NAND tree quantum verification of matrix products*", Eprint arXiv:quant-ph/0702144, 2007

[39] Gay S. J.: „*Quantum Programming Languages: Survey and bibliography*", Bulletin of the EATCS, 86, 2005

[40] Geroch R.: „*Perspectives in Computation*", The University of Chicago Press, 2009

[41] Gilbert J., Gilbert L.: „*Linear Algebra and Matrix Theory*", Thomson, Brooks/Cole, 2004

[42] Gottesman D., Chuang I. L.: „*Quantum Teleportation Is a Universal Computational Primitive*", Nature, 402, 1999

[43] Grimaldi R. P.: „*Discrete and Combinatorial Mathematics: An Applied Introduction*", Addison-Wesley, 2003

[44] Grover L.: „*A Fast Quantum-Mechanical Algorithm for Database Search*", ACM Symposium on Theory of Computing, ACM, 1996

[45] Hallgren S.: „*Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem*", in Proceedings of the 34th Annual ACM Symposium on the Theory of Computation, New York, 2002

[46] Hirvensalo M.: „*Quantum Computing*", Springer, 2001

[47] Jozsa R.: „*Quantum Algorithms and the Fourier transform*", arXive e-print quant-ph, 1997

[48] Kitaev A. Yu., Shen A. H., Vyalyi M. N.: „*Classical and Quantum Computation (Graduate Studies in Mathematics)*", American Mathematical Society, 2002

[49] Lee C. F., Johnson N. F.: „*Let the quantum games begin*", Physics World, 2002

[50] Magniez F., Santha M., Szegedy M.: „*Quantum algorithms for the triangle problem*", in Proceedings of the 16th Annual ACM SIAM Symposium on Discrete Algorithms, 2005

[51] Mosca M.: „*Quantum Computer Algorithms*", Ph.D. Thesis, University of Oxford, 1999

[52] Nielsen M. A., Chuang I. L.: „*Quantum Computation and Quantum Information*", Cambridge University Press, 2004

[53] Shannon C. E., Weaver W.: „*The Mathematical Theory of Communication*", University of Illinois Press, 1998

[54] Pittenger A. O.: „*An introduction to Quantum Computing Algorithms*", Progress in Computer Science and Applied Logic, Vol. 19, Birkhauser, Boston, 2001

[55] Ömer B.: „*A Procedural Formalism for Quantum Computing*", doctoral dissertation, Dept. Theoretical Physics, Technical Univ. of Vienna, 1998

[56] Preskill J.: „*Advanced Mathematical Methods of Physics – Quantum Computation and Information*", California Institute of Technology, 1998

[57] Raussendorf R., Briegel H. J.: „*A One Way Quantum Computer*", Physical Review Letters, 86, 2001

[58] Rieffel E.: „*An Introduction to Quantum Computing for Non-Psysicists*", ACM Computing Surveys, Vol. 32., 2000

[59] Ruediger R.: „*Quantum Programming Languages: An Introductory Overview*", The Computer Journal, 50(2), 2007

[60] Rosen K. H.: „*Discrete Mathematics and Its Applications*", McGraw-Hill, 2003

[61] Selinger P.: „*Towards a Quantum Programming Language*", Mathematical Structures in Computer Science, 2004

[62] Shende V.V., Markov I.L., Bullock S.S.: „*Syntesis of Quantum Logic Circuits*", IEEE Trans. Computer-Aided Design of Integrated circuits, 2006

[63] Shende V.V., Markov I.L., Bullock S.S.: „*Finding Small Two-Qubit Circuits*", Proc. SPIE, vol. 5436, 2004

[64] Shor P. W.: „*Algorithms for Quantum Computation: Discrete Logarithms and Factoring*", IEEE Press, 1994

[65] Shor P. W.: „*Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*", SIAM, 26(5), 1997

[66] Shor P. W.: „*Introduction to Quantum Algorithms*", Proceedings of the Symposium in Applied Mathematics, 58, 2002

[67] Shor P. W.: „*Why Haven't More Quantum Algorithms Been Found?*", Journal of the ACM, 50(1), 2003

[68] Sipser M.: „*Introduction to the Theory of Computation*", Thomson Course Technology, 2005

[69] Svore K. M., Aho A. V., Cross A. W., Chuang I., Markov I. L.: „*A Layered Software Architecture for Quantum Computing Design Tools*", Computer, January 2006

[70] Svore K.M., Terhal B.M., DiVincenzo D.P.: „*Local Fault-Tolerant Quantum Computation*", Physical Rev. A, vol. 72, no. 5, http://arxiv.org/abs/quant-ph/0410047, 2005

[71] Unruh W. G.: „*Maintaining Coherence in Quantum Computers*", Physical Review A 51, 992, 2001

[72] Viamontes G.F., Markov I.L., Hayes J.P.: „*Graph-Based Simulation of Quantum Computation in the State-Vector and Density-Matrix Representation*", Quantum Information and Computation, vol. 5, no. 2, 2005

[73] Yanofsky N. S., Mannucci M. A.: „*Quantum Computing for Computer Scientists*", Cambridge University Press, 2008

[74] Zalka C.: „*Grover's Quantum Searching Algorithm Is Optimal*", Physical Review Letters A 60(4), 1999

[75] Zhou X., Leung D. W., Chuang I. L.: „*Quantum Logic Gate Constructions with one-bit Teleportation*", arXive e-print quant-ph/0002039, 2000

[76] http://www.qubit.org
[77] http://www.theory.caltech.edu/people/preskill/ph229/#lecture
[78] http://www.eskimo.com/~knill/qip/prhtml/node2.html