Quantum Information Technologies

Simulation techniques for Quantum Information Theory and Quantum Cryptography

Fernando Lucas Rodríguez

March 2016

Links

This document	http://dotqcf.sourceforge.net/data/thesis-report.pdf
Latest binary	http://sourceforge.net/projects/dotqcf/files/latest/download
SVN head	http://svn.code.sf.net/p/dotqcf/code/head/
Webpage	http://dotqcf.sourceforge.net/

Revisions record

Version	Date	Changes
1.0	2005/09/15	Initial release for the Master Thesis using MS Word
1.1	2006/11/15	Basic conversion into LaTeX
1.2	2007/02/22	Addition of some chapters generated for doctorate courses
1.3	2007/11/20	Partial translation into English
1.4	2008/01/22	Improvements in layout
1.5	2009/09/31	Translation advances, global rearrangement
1.6	2010/05/15	Full translation

This document was typesetted with XeLaTeX, using with the help of some extra packages such as the memoir class. All the TeX files and images were managed using SVN.
The fonts are OpenType embedded into the PDF, and the layout is A4 vertical double-side.
Most of the images are vectorial, allowing high resolution zoom for the graphic details, when using the electronic version of this document.

Abstract

This thesis aims to design a framework for simulating process of Quantum Information Theory, Quantum Communications and Quantum Cryptography.

Resumen

Version española

(This page is intentionally left blank)

Part I

About the thesis

(This page is intentionally left blank)

Preface and Objectives

This program is an evolution of my end of career thesis. All suggestions, comments and request are welcome. I will try to add them on next releases.

It tries to be a generic framework for simulating Quantum Information processes, but by now it only deals effectively with BB84, but it is designed to simulate logical gates and entanglement. Also some noise can be simulated and eavesdropping. As far I know it is the only one public available simulator of BB84.

The main difference with other simulations, is that all mathematical background is abstracted. You will only work with quantum states, not with the coefficients (matrix, vectors or whatever,...) and will apply to them operators, not mathematical operations. But states, are really simulated by complex coefficients, and you can explore them through debugger objects. This is the only one quantum information simulator that deals with quantum states as vectors (excluding Matlab packages or similar).

It is programmed with Microsoft Visual Studio 2005 and C#, but it is designed to be able to interoperate with other languages.

You are what you cache, so remember...

(This page is intentionally left blank)

List of contents

(This page is intentionally left blank)

Part II

Environment and Precedents

(This page is intentionally left blank)

Chapter 1

Quantum Information motivations

1.1 Current status of technology

From the production of EDSAC I in the mid-forties, the industry computers has continued to grow by giant steps. This industry and performance grows grows at an exponential rate. As a reference, the computational power of PCs that were sold six years ago is the same as in the current PDAs.

The invention of the transistor was the generation step of the first computers. On the one hand replaced vacuum lamps, which were used as a basic memory device, and secondly, allowed the construction of integrated circuits. The use of these marked the beginning of the race technology in which we are rapidly getting cheaper and with teams better benefits, as components of integrated circuits are becoming more powerful and more small.

In 1965, Gordon Moore [Moo65] made an observation, later known as Moore' Law, about the speed with which the number of transistors in the devices:

The complexity for minimum component costs has increased at a rate of roughly a factor of two per year...Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65000. I believe that such a large circuit can be built on a single wafer.

Moore proofed to be a visionary. Not as fast as he predicted, the number of transistors per chip has been duplicated, since the law has formulated, every 18 to 24 months.

The development of computers industry has been possible due to the miniaturization of the components for integrated circuits. This miniaturization reduces the material costs and the energetic consumption, while raising the operation speed. However, this approach cannot be sustained forever.

This uncertain future seems something distant. But if we apply inversely the Moore' Law, by calculating when the size of the transistors will similar to the size of one atom, when the semiconductor industry will collapse, we discover that this limit will be around 2020!

When it comes to atomic scale, the laws of physics change. Classical physics are no longer, which are based vacuum valves, and whose operation is simulated by semiconductors. There are new opportunities for

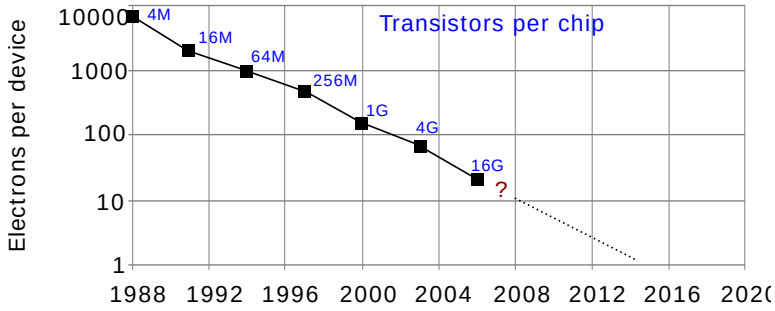

Figure 1.1: Transistors integration evolution

calculation, Fourier Transform are operations with a lower complexity than classical methods; but also usual ways of work are no longer valid, such is the copy of information, traditionally considered as something 'basic'.

Consequently, new ways of working different of the ones formalized by Shannon in 'A Mathematical Theory of Communication' [Sha48] are needed. There is need for a new information theory, the so called 'Quantum Information Theory'.

1.2 Historic evolution of Information Theory

The theory of information appears as a set of interpretations to the laws of thermodynamics and to the problems or paradoxes that appears. The most relevant of all of them is the 'Maxwell' daemon'

This daemon was first enunciated in 1871, in the book 'Theory of Heat' by J. C. Maxwell [Max71], under the section 'Limitation of the Second Law of Thermodynamics'. It was a mental experiment to illustrate the limitations of the second law, empathizing that it is an statistic principle, that applies generally in systems composed by many particles. This experiment is expressed as:

... if we conceive of a being whose faculties are so sharpened that he can follow every molecule in its course, such a being, whose attributes are as essentially finite as our own, would be able to do what is impossible to us. For we have seen that molecules in a vessel full of air at uniform temperature are moving with velocities by no means uniform, though the mean velocity of any great number of them, arbitrarily selected, is almost exactly uniform. Now let us suppose that such a vessel is divided into two portions, A and B, by a division in which there is a small hole, and that a being, who can see the individual molecules, opens and closes this hole, so as to allow only the swifter molecules to pass from A to B, and only the slower molecules to pass from B to A. He will thus, without expenditure of work, raise the temperature of B and lower that of A, in contradiction to the second law of thermodynamics.

When Maxwell enunciated the daemon was not much specific, it was not defined how it should operate neither its objectives. Consequently 2 different kind of daemons were understood and several methods to detect the molecules to allow to pass or reject. The original daemon was a *temperature* one, that work in systems thermally isolated and generates a difference of temperature without making any work. The other kind of daemon is the *pressure* one, that is more simple and only allows cross to one side or the other that goes in one or other direction respectively; basically it is a valve.

A temperature daemon must be able to detect the speed of individual molecules inside a gas, but up to the present any physical device is able to fulfill this aim with a minimal amount of work and with a minimal dissipation. As already said, for an daemon working successfully, the entropy of the system must not

increase (an irreversible process cannot take place). According to this, the daemon must be in thermal equilibrium with the gas in which is inserted.

If a daemon uses optical signals as a method for receiving the information from the molecules, starts receiving energy and rising its temperature, having to transfer heat to somewhere else (a reservoir, the exterior,...). In any case, assuming that daemon suppress the excess of energy goes against the hypothesis of a isolated system and in any case this violates the second law. Smoluchowski was the first to realize of this fact, but allowed the door opened to the possibility that an intelligent being could operate a perpetual movement machine suppressing these fluctuations.

In a more concise way, the laws of thermodynamic for this example, can be expressed as:

1. The second law needs that the entropy of the daemon increases in a quantity greater or equal to the reduction of entropy in the gas.
2. The first law implies that the energies of the daemon and the gas cannot change (isolated system).

Consequently the daemon much increase its entropy while keeping the same amount of energy. But if the daemon continues increasing the entropy without limits, at some moment it will be unable to continue executing its tasks (too noisy environment). A solution for this is to reset it, coming back to the initial state. This also makes the analysis easier, the main interest is put on the gas and not in the daemon (obviously, the reset of the daemon must be done without interfering with the gas). It is needed a heat reservoir so that the daemon can transfer the excess of entropy, and it is needed also a work source to supply energy to the daemon at a constant rate of entropy.

As a resume, the first two laws of thermodynamics assure that:

1. The entropy of the daemon increases with decreases the entropy of the gas.
2. It cannot be reset without exchanging energy with an energy source.
3. The reset process is an exchange of energy between an energy source and a reservoir. (energy source $\rightarrow$ daemon $\rightarrow$ reservoir)

In 1929 Leo Szilard published an article called 'On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings' [Szi29] where is enunciated the hypothesis of a 'perpetuum mobile' (perpetual movement machine) of the second type (using heat for lowering temperatures).

The model proposed by Szilard is a volume V where there is a single particle. After that, this volume V is divided into two identical subvolumes V_1 and V_2. At this moment the intelligent being intervenes determining where is the particle, if at the right (V_1) or at the left (V_2) and records the result. Let's suppose that the particle is at the right (V_1). How the location of the particle is known, this information is used for changing the division of the initial volume by means of a piston. The gas performs a work W onto the piston, restoring the initial volume V. The work needed for moving the piston is transferred as heat to the gas molecule by a reservoir, $Q = W$. After this process apparently the second law has been violated, because the reservoir transferred heat decreasing the entropy. Szilard assumes that one measurement is associated to certain mean production of entropy, and this entropy restores the entropy reduced by the daemon, as stated by the second law. Szilard also identifies and demonstrates that this fundamental amount is $kln2$. The most relevant critic to the Szilard hypothesis is that the entropy of a mesurable quantity does not depends of the knowledge or ignorance of the observer about the system. In any case the work of Szilard was pioneer in fields of Information Theory, cybernetics and computation.

Since 1940 it was already known that a high temperature lamp was needed so that the demon could distinguish the light signals produced by the Kirchhoff's radiation law. In 1949 Shannon introduced a called function *Entropy of Information*. In 1951 Brillouin Leon, assuming the latest, and using the quantum nature of the radiation, demonstrated that the entropy explicitly that the demon obtains was sufficient to preserve to the second law.

Explaining this better:

Lets suppose that P_0 is the total number of possible states of a system and that we ignore the current state of the system. Then the information that we have is $I_0 = 0$.

If now we carry out measurements and we reduce the number of possible states to P_1, the information that we have now is defined as $I_1 = K\ cdotln\,(P_0/P_1)$, with K_0 as a positive constant. If $P_1 > P_0$, $I_1 < 0$, we say that the systems looses information.

Arter this, in the book of 1956, 'Science and Information Theory' [Bri56], develops more this theory, and distinguishes between two types of information:

- 'free information', I_f, the one that does not have thermodynamical relevance, as can be the previous knowledge of a person.
- 'bound information', I_b, defined as function of the possible states of a system. The knowledge of a person can be converted into 'bound information' when it is transmitted using a physical mean.

Later, related the changes in information I_b with changes in the entropy in the following way:

$$I_{b1} - I_{b0} = K \cdot (lnP_0 - lnP_1) = S_0 - S_1 > 0 \tag{1.1}$$

As we can see, this says that if we gained information on a thermodynamic system, we make diminish the entropy. After that Brillouin defined the 'negentropy' as $N \equiv -\left(entropy\right)$, or equivalently, $\Delta N = -\Delta S$.

Relating these results to an isolated system, with an entropy $S_1 = S_0 - I_{b1}$, as stated previously, so we get

$$\Delta S_1 = \Delta S_0 - \Delta I_{b1} = -\Delta N_0 - \Delta I_{b1} = 0 \tag{1.2}$$

or more clearly

$$\Delta N_0 + \Delta I_{b1} = 0 \tag{1.3}$$

the amount negentropy plus information does not increases and in a reversible process it is fixed. This statement is a new interpretation of the second law of the thermodynamics. With these results Brillouin exorcised the daemon. As with the work of Szilard, it had varied reactions: this reinterpretation of the second law was accepted by some and rejected by others, the main critic was that the entropy is not subjective, does not depend on the observer.

In 1961 Landauer presented an article where it develops what today is known as **Landauer' Principle** [Lan61], that basically says that when a bit of information is erased, the entropy of the ambient is at least increased by $kln2$. If we remember the model of Szilard, what saved to the second law was that the diminution of the entropy of reservoir was compensated by the increase of the entropy due to the measurements done by the daemon. In an article published in 1973 [Ben73] Charles Bennett argued that the measurements do not increase the entropy since they can be done of such form that the heat dissipation is arbitrarily small.

Does this means this that the second law is broken in the model of Szilard?
No, because remembering that it was desirable to erase the memory of the daemon and this act is the one that saves the second law. Since according to the Principle of Landauer, when erasing the memory of the daemon the entropy of the reservoir is increased. Bennett in 1982 demonstrated that the 2 amounts were compensated, reason why the second law remains valid.

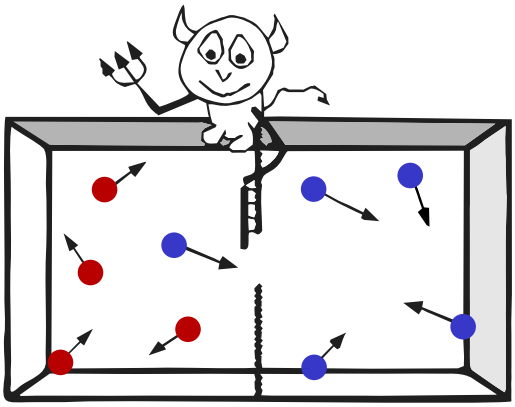

Figure 1.2: Maxwell' daemon

(This page is intentionally left blank)

Chapter 2

Some experiments on quantum physics

2.1 Polarized states experiment

The polarizers are optical devices that 'filter' the sources luminance. The luminance sources, generally, emit photons polarized in multiple directions and the polarizers only let pass photons in a certain direction (axis of polarization).

We begin with something simple, a luminance source and a polarizer as in Figure 2.1,

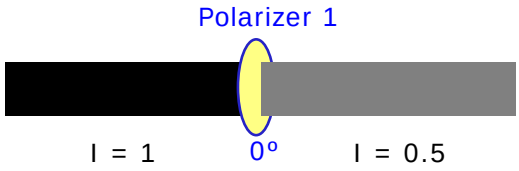

Figure 2.1: First test with the polarizers

after the first polarizer, the intensity luminance the reduced one to half.

We add a second polarizer next. We put if it in the same direction of polarization that first, the light continues going through. But turning it 90Âº from the first one, the light disappears, as is in Figure 2.2.

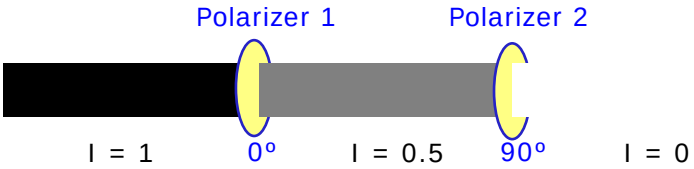

Figure 2.2: Second test with the polarizers

Leaving both polarizers in this position, we add among them a third one:

we turned while it we see as it returns to appear the light at the end of all the polarizers, at an angle of 45Âº.

It is strange that adding a third filter we obtain an increase of luminance intensity.

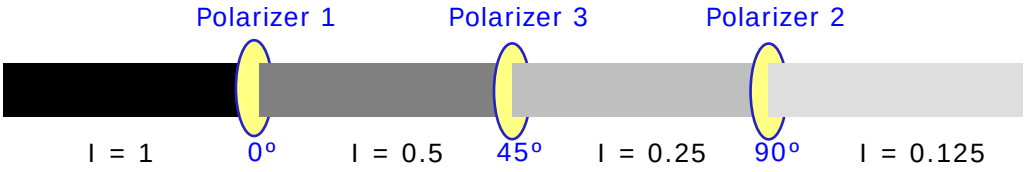

Figure 2.3: Third test with the polarizers

The explanation is the following:

- Each cascade polarizer (although all they are arranged in the same axis of polarization), reduce the luminance intensity to half.
- After going through the first one polarizer, the light ray is polarized in the same direction that the polarizer.
- The intermediate polarizer this to 45Âº with respect to first. Like not this nor in the same position that the first one, without in the opposed one (90Âº), but in an intermediate one, probability that they pass photons is of to the 50%. The last one polarizer is 45Âº with respect to the intermediary one (90Âº with respect to first) and like in the step previous, the photons happen with a probability of the 50%.
- If an intermediate polarizer did not exist the first and the last would directly have a relative turn of 90Âº to each other. This 90Âº is opposes direction of polarization, and the probability that they go through the polarizer is zero.

2.2 Young' double slit

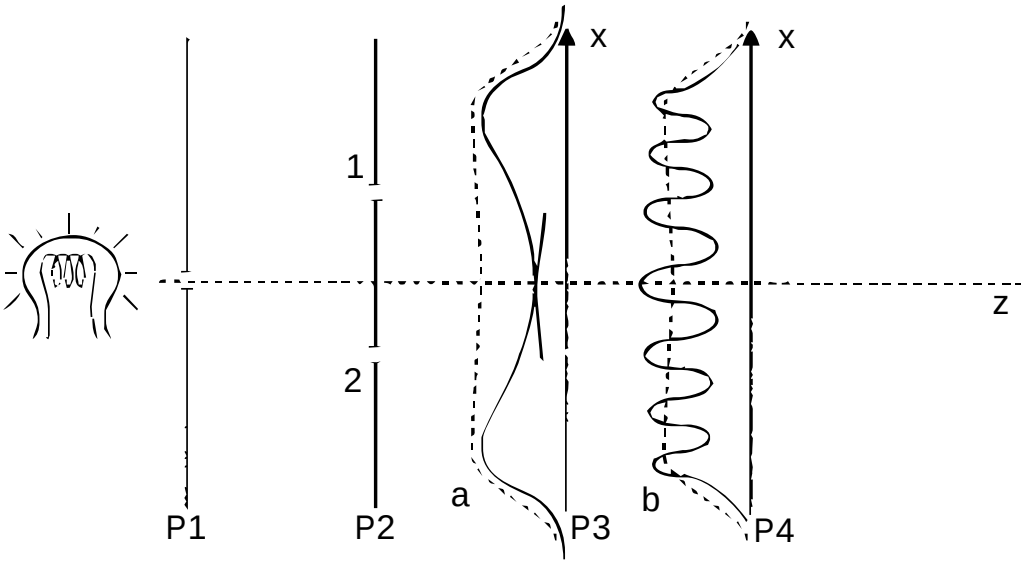

Figure 2.4: Young' double slit experiment setup

In the Young' experimentthe light leaving through a hole in the left wall must cross the central wall through two grooves. A detector in the wall of the right part measures the luminance intensity throughout the surface of this wall.

If a unique groove remains open, the luminance intensity reaches its maximum in the air line that unites the hole and luminance source. Moving away of this position the intensity it falls gradually and finally it

disappears. This is reflected in the drawing by means of curve A.

When both grooves remain open, the resulting luminance intensity is not the sum of the intensities of each groove separately. In fact it arises an oscillating landlord from light and the dark. Curve C illustrates east landlord. This effect is caused by the interference of the light that crosses both grooves. This effect is observable even if the luminance intensity of the source is reduced to individual photons. Making a distribution statistic of the resulting positions forms the curved B again. Is something strange and surprising, it could say that each photon interferes with itself.

The intention of this example is to show that the intuition for the classic physics is hardly applicable for the quantum systems.

(This page is intentionally left blank)

Chapter 3

Quantum Information Theory

3.1 Introduction to photonic quantum states

3.1.1 Properties of light

Light travels with the speed of light which is $300.000km$ per second in vacuum. This makes light a very good information carrier for transport. For instance optical fibers are often used in the communication industry. The radio waves received by an ordinary FM-radio are also a kind of light (with a slower oscillation frequency than visible light).

Light behaves like waves. It travels precisely as waves on the surface of the ocean or sound waves through the air - only a lot faster.

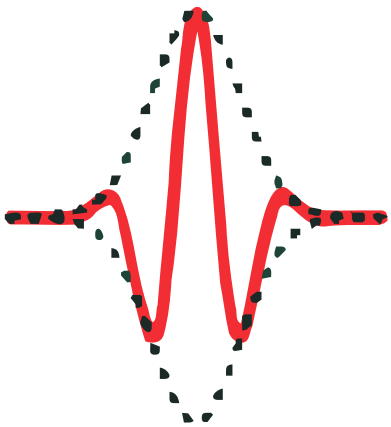

Figure 3.1: Example of a traveling light wave

There are different important properties of a light wave. The color of the light is decided by the wavelength as illustrated by the first picture. Blue light has a shorter wavelength than red light. The strength of the light is set by the magnitude of the wave excursion (second picture). Finally there is a property called phase. This property decides when the wave is at the top or bottom. In the last picture the phase is varied continuously.

3.1.2 Atoms as small spinning tops

Atoms have the property that they can be at rest. This is not the case for light. To store some information for an amount of time atoms are a good choice, contrary to light.

There are many different properties characterizing atoms. One of these is called spin. You may think of a little spinning top (hence the name). The spinning top behavior may be caused by an electron moving around the atomic nucleus. The direction of the axis around which the top is spinning can be anywhere in space.

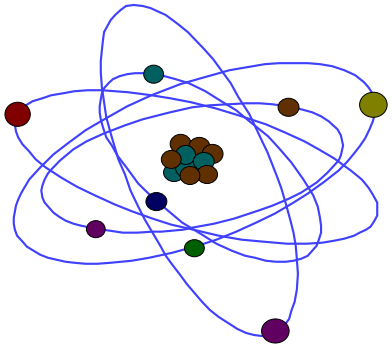

Figure 3.2: Sometimes atoms behave like spinning tops

3.1.3 Atoms as memory for a state of light

As we discussed above, light is good for transporting information at atoms are good for storage of information. We assume that some information has been encoded in the strength and the phase of this light. We would like to transfer this information to atoms and hence we must find a method to represent the strength and phase of the light in the spinning top.

Here we picture different states of light and atoms which correspond to each other. If the spinning top is pointing in the vertical direction the light is off (has strength zero). The stronger the light, the more the top is rotated away from vertical. The phase is encoded in the direction in which the top is rotated away from vertical.

Now let us turn to an actual procedure of storing the information carried by light in a spinning top. For instance one can simply measure the strength and phase of the light and note the result on a piece of paper. Then we know how much and in which direction to rotate the spinning top.

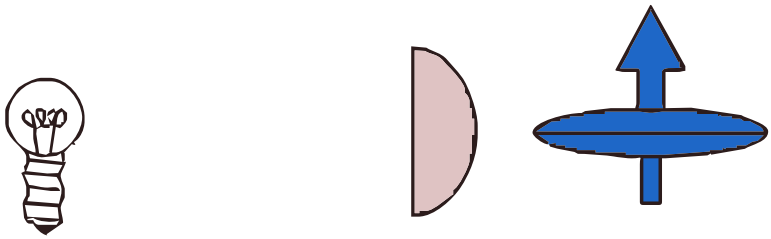

Figure 3.3: Photon interaction

The light hits a detector which measures the strength and phase of light. Then the spinning top can be rotated by the proper amount in the right direction and the information has been transferred. In the next section we denote this procedure ‘classical memory’.

There is a slight problem, though. We have forgotten to check with the laws of nature whether this is at all possible! The method explained above actually only works with a certain precision. This is a consequence of quantum mechanics. We will explain more below.

3.1.4 Quantum mechanics for light and atoms

Quantum mechanics is the physical theory which is necessary to describe the smallest parts of nature. The word 'quantum' arises from the fact that nature contains a kind of smallest building blocks (a smallest quantum) of literally everything.

A single photon represents very, very weak light, in fact, the smallest parts of light. For instance a red bulb with $60W$ light power emits roughly 210^20 photons every second. You have to look very carefully to see the effect of a single photon. However, this is possible in modern physics experiments.

The fact that nature is a collection of smallest building blocks has strong consequences, especially for systems of a size so small that only a few building blocks are present. In fact, a single photon only has space available to carry 'a certain amount of information'. Hence the total amount of information in a weak pulse of light has to be small. This may be compared to a book so small that there is only space for e.g. five letters. Evidently not much information can be written in such a book. Practically all this means that a weak pulse of light will seem to be quite noisy.

A weak pulse of light is noisy, it contains so-called quantum noise. This kind of noise is in principle similar to noise on an FM-radio or to 'snow' on a TV-screen. There is just one difference; you cannot get rid of the noise. It will always be there to some extent. It is not only light which becomes noisy as a consequence of quantum mechanics. A spinning top consisting of few atoms will also contain a considerable amount of noise. This may be visualized as if the direction of the spinning top is not exactly defined.

The fact that the direction of the spinning top cannot be precisely defined can be viewed from another perspective. You can say that the spinning top both points a little to the left and a little to the right at the same time. Within quantum mechanics physical systems can easily be in different states at the same time. This is commonplace for quantum physicists but nonsense for non-physicists. But whether you are a quantum physicist or not, this just has to be accepted without further explanation. This is the way nature works!

Quantum mechanics has consequences for the method of information transfer which we denoted 'classical memory' above. The quantum noise prevents us from measuring the incoming state of light precisely; it is simply not possible.

During a measurement we can only extract a certain amount of information about the state of light since the light particles that we will detect each has a maximum amount of information.

We are showing two cases. In one we obtain good information about the strength of the light state but we must accept that we do not know much about the phase. The other case is opposite, we detect the phase with good accuracy but now it is hard to estimate the strength of the light.

The above example represents a well-known phenomenon in quantum mechanics, the so-called complementarity principle. The strength and phase of light are complementary. They cannot both be precisely defined at the same time.

For atoms, or our spinning top, there is also a complementarity principle. When the spinning top is pointing roughly in the vertical direction (within some allowed precision) you may e.g. ask one of two questions:

1. How much does the spinning top point in/out of the picture?
2. How much is the spinning top pointing left/right?

These two questions cannot be answered precisely at the same time. The two directions are complementary, just like the strength and phase was for light.

Now we may understand why the 'classical memory' is not entirely precise. In this case we would like to transfer both the strength and the phase of light to atoms. So instead of making a very precise measurement of either the strength or the phase, we have to compromise and make a half-precise measurement on both.

3.1.5 Quantum memory

But now let us see how to transfer a state of light to atoms with a method better than 'classical memory'. This is what we call 'quantum memory' (classical as opposed to quantum).

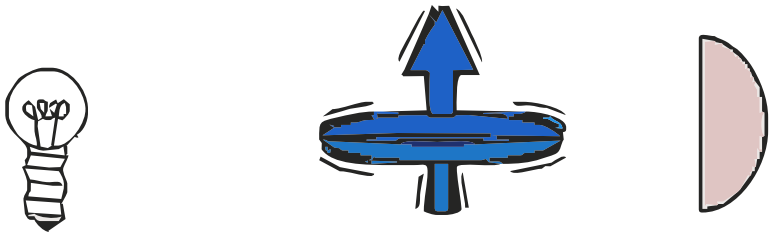

Figure 3.4: The light is sent **through** atoms

The experiment consists in sent photon through atoms. If the experiment is carried out correctly, a part of the light information (e.g. the phase) will be transferred to atoms.

The spinning top (atom) is rotated slightly out of the picture while the light is passing by. Next we may concentrate on measuring the remaining information about the light (e.g. the amplitude). We will not obtain information about the phase of the light but this is not important since this information is already stored in atoms. The measured information of the light strength can now be used to rotate the spinning top into the correct position.

The above procedure works better than 'classical memory'. The incoming light state is unknown for us, and quantum mechanics actually forbids us to obtain a complete picture of the incoming state. Anyway it is possible to design a procedure of information transfer from light to atoms which works better than if we had tried to measure on the light directly.

It is a two step measurement. The photon interacts with the atom while is sent through it. An interaction enough for change atoms' global phase. The photon properties are altered after this interaction. And in a second step using a polarization detector we concentrate on measuring only photon polarization, and externally change atom spin.

On each step only one property is measured and we get in conjunction a better description of the photon sent that with a unique measurement.

3.2 Quantum Mechanics postulates

The postulates of non relativistic quantum mechanics refers to the description of a physical system, the description of the physical magnitudes the measurement process of the system and the temporal evolution of the system.

Definition 1 (First Postulate). *In a given instant, t_0, the state of the quantum system is described by an element $|\phi(t_0)\rangle$ (named ket) from the set of possible states of the system.*

This set of possible states, has an structure of vectorial space. The wavefunction $\phi(r,t)$, is a representation of the quantum state of the system in a particular base, and contains the maximum of information of the system in that instant. The space of wavefunctions has the structure of a Hilbert's space.

As a consequence of the structure of vectorial space, this first postulate, implies the principle of superposition: a linear combination of state vectors is also a state vector.

Definition 2 (Second Postulate). *It states that every physical magnitude is described as operator, A, that acts into the state space. The set of eigenvectors of this operator (that is Hermitian) form a base in the space. So by establishing a basis, a vectorial representation of the operator can be obtained.*

Definition 3 (Third Postulate). *It states that the result of a measurement of a physical magnitude is one of the eigenvalues of the operator that describe that magnitude. How the eigenvalues of a Hermitian matrix are real, the measurement will be a real number.*

Definition 4 (Fourth Postulate). *It is the probabilist foundation of quantum mechanics. When a physical magnitude is measured, A, in an state, $|\phi\rangle$, normalized to the unit, the probability of obtaining the the eigenvalue a_n is the square of the module of the associated coefficient of the the related eigenvector, $|u_n\rangle$, of the state $|\phi\rangle$ in the basis $|u_n\rangle$.*

The application of this postulate to the measurement of a particle whose state is $|\phi\rangle$ allows to define $|\phi(r,t)\rangle^2$ as the probability amplitude of that particle as function of position r in the instant t. This concept is interpreted as the density of probability of finding the particle in that position.

Definition 5 (Fifth Postulate). *It is one of the most polemics. It states that when measuring a physical magnitude and finding out the value, (this means finding the eigenvalue a_n), the state of the system immediately after of the measurement is the associated eigenvector to that eigenvalue. A 'instantaneous collapse' of the state of the system takes place and the unique responsible is the measurement process.*

This must be understood as an 'update' of the information stored in $|\phi\rangle$, that is what really identifies the state of the system. When we obtain something from the system as consequence of an observation it is needed to ignore the parts of $|\phi\rangle$ that are inconsistent with the new information acquired from it.

Definition 6 (Sixth Postulate). *It deals with the temporal evolution of the quantum system. That temporal evolution of $|\phi\rangle$ is given by the Schrödinger equation:*

$$i\hbar\frac{d}{dt}|\phi\rangle = H(t)|\phi\rangle \tag{3.1}$$

being $\hbar = \frac{h}{2\pi}$ the reduced Planck constant, and $H(t)$ the Hamiltonian observable (corresponding to the physical magnitude 'total energy' of the system). It is necessary to mark, that if known the initial state of the system, the temporal evolution is fully determined, the quantum indeterminism resides in the measurement process.

Quantum mechanics, although the formal robustness is not free from critics. In fact, there are many arguments about if considering it a complete theory.

The quantum character is shown in the indeterminism. But this is not exclusive of quantum physics (chaotic systems are a good counterexample). And the other main characteristic is the non-separability of quantum states included in the superposition principle.

Those states, that are the lineal combination of quantum states, are known as 'entangled' states. They are in the basis of quantum physics, and they allow to develop important and spectacular applications as quantum cryptography and quantum computation.

3.3 Notation

One of the first problems when starting the study of Quantum Computation is the notation. We can find expressions quite unusual, as for example

$$\left\langle A\sum lim_x|x\rangle \middle| A\sum lim_x|x\rangle \sum \lim_y \left(c_y - A\right)|y\rangle \middle| \sum lim_y \left(c_y - A\right)|y\rangle \right\rangle \tag{3.2}$$

and we are surprised by the lack of symmetry in this notation. But once used to it, we will be able to follow problems that seems complicated, but they seem to become easier once used to the new notation.

We will use notation of *ket* or Dirack's notation. The name comes from using the notation *bra-ket*, where ket is an expression of the type $|a\rangle$, (it represents a column vector in a bi-dimensional Hilbert space), a bra will be of the type $\langle b|$, being a row vector. And the expression $\langle a\,|\,ab\,|\,b\rangle$ will be the *inner product*(scalar product) between both vectors; a and b. As an example, we can identify the vector $\begin{pmatrix}1\\0\end{pmatrix}$ with $|0\rangle$, and $\begin{pmatrix}0\\1\end{pmatrix}$ with $|1\rangle$.

When working with the quantum computing model, when using expressions of the type $|010\rangle$ we will think in the string 010; and in general, $|x\rangle$ with $x \in \{0,1\}*$ will represent the string x. If we see expressions as $|x\rangle\,|y\rangle$ they will be the concatenation, simply $|x\rangle\,|y\rangle = |xy\rangle$.

We have show the meaning of the notation *bra-ket*, however we rarely find expressions of the type $|x\rangle\langle x|$ or $\langle x\rangle$ in Quantum Information. It is not necessary to worry in the lack of symmetry in the notation, it is just a notation.

3.4 Superposition

This section defines the concept of superposition. This key concept, together with the uncertainly of the quantum theory, will introduce a massive parallelism in the data processing. The work space will be a Hilbert's space. (A Hilbert's space is an space with a norm and complete).

Definition 7. *A classic state is defined as a vector in a certain vectorial space. A classic state will represent a physical magnitude (a global description of the system): speed, position,...We will say that a physical system is a set of states that describe a certain situation.*

Definition 8. *Given a physical system with n different states $|1\rangle, |2\rangle, \ldots, |n\rangle$*
we will say that a superposition of them is a lineal combination of the type $|\phi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \ldots + \alpha_n|n\rangle$ where $\alpha_i \in C$ verify $\sum lim_{i=1}^{n}|\alpha_i\rangle^2 = 1$.

Each coefficient α_i is named as amplitude of $|i\rangle$ in $|\phi\rangle$.

The n lineally independent states, $|1\rangle, |2\rangle, \ldots, |n\rangle$, are orthonormalized.

Definition 9. *We will say that a quantum sate is a superposition of classic states; i.e. in the previous expression $|\phi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \ldots + \alpha_n|n\rangle$ we will say that $|\phi\rangle$ is a quantum state.*

A quantum state is therefore a vector with an euclidean norm of value 1 (because $\sum lim_{i=1}^{n}|\alpha_i|^2 = 1$) in a Hilbert's space of finite dimensions.

In the quantum theory there is a probabilistic element that cannot be defined. This randomness is the uncertainly of the measurement process and the way of modeling it using a probability. In this way we can follow all the possibilities of evolution with corresponding 'weights'. It can be understood that we have a

probability distribution where each variable represents a different state; but this is wrong. The superposition concept includes also the wave behavior. Two waves 'add' if their peaks coincide and 'subtract' if one peak and valley coincide. For this reason the amplitude are complex numbers that can compensate ones with another, and the probability of each one of them will be obtained through the square (in this way they are real values). For this reason it is necessary to verify $\sum lim_{i=1}^{n} |\alpha_i\rangle^2 = 1$.

We will see in Chapter 7 the difference between the quantum model and the probabilistic model. But the quantum one, can be understood as a generalization of the probabilistic one, that includes 'wave' behaviors, but the evolution is also fundamentally different.

Intuitively we see that a quantum state is a linear combination of classic states that evolves independently of the environment and in a certain moment, due to a process of interaction with the 'environment' (observation), the quantum state transforms results into a classic state well defined: it collapses. The square of amplitudes show what state has more probabilities of being obtained during the measurement process.

So, a quantum state can only do two things: 'evolve or being observed'.

3.5 Quantum Information unit: the qubit

In classic computing theory the *bit* is the basic unit of information, and has two possibles values: 0 or 1. In quantum computing we can consider the states $|0\rangle$ and $|1\rangle$, both vectors in a bidimensional Hilbert's space.

Definition 10. *A quantum bit, or qubit, is the superposition of*

$$\alpha |0\rangle + \beta |1\rangle \tag{3.3}$$

where α and β are complex numbers, amplitudes, verifying:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.4}$$

We can generalize this idea and say that a registry of n qubits has 2^n base states, each one of them of the type $|b_1\rangle, |b_2\rangle, \ldots, |b_n\rangle$ being $b_i \in \{0, 1\}$, and we can shorten this as $|b_1 b_2 \ldots b_n\rangle$. This string of size n can be understood as numbers between 0 y 2^{n-1} and therefore write $|1\rangle, |2\rangle, \ldots, \left|2^{n-1}\right\rangle$. So, a quantum registry of n qubits is a superposition as

$$\alpha_0 |0\rangle + \alpha_1 |1\rangle + \ldots + \alpha_{2^n-1} |2^n - 1\rangle \tag{3.5}$$

with the additional condition

$$\sum lim_{i=1}^{n} |\alpha_i|^2 = 1 \tag{3.6}$$

(In this expression $|0\rangle$ means $|00 \ldots 0\rangle$, k zeros and $|2^n - 1\rangle$ means $|111 \ldots 1\rangle$, k ones).

In a bit we store a value. In a qubit, however, we sotre two values. A qubit is the quantum superposition that when collapses will give one of those two values. But until this moment it stores the double of information than a classic bit.

We have 2^n complex numbers for a system of n-qubits and in a system of n bits we have n values. Somehow it is possible to say that the storage capacity has improved.

3.6 Quantum states operations

3.6.1 Unitary evolution

The quantum theory states that we can only perform linear operations on quantum states.

Therefore, the transformation of a quantum state of the type: $|\phi\rangle = \alpha_1 |1\rangle + \alpha_2 |2\rangle + ... + \alpha_n |n\rangle$
into another one of the type $|\phi\rangle = \beta_1 |1\rangle + \beta_2 |2\rangle + ... + \beta_n |n\rangle$
is done by means of a linear application, that is the matrix $U \in M(n, n)$:

$$U \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \tag{3.7}$$

Because $|\phi\rangle$ is a quantum state that verifies $\sum lim_{i=1}^{n} |\beta_i\rangle^2 = 1$, the application U must preserve the norm of the vectors, this is, it must be a unitary transformation. A matrix is unitary if it inversea, U^{-1}, is equal to its conjugated transposed, U^*, and this means that transforms vectors of module 1 into vectors of module 1.

Definition 11. *An unitary evolution between two quantum states is an unitary linear application, this means that $UU^* = I$ (being U^* the conjugated transposed of U), in the Hilbert's we are using.*

From the computational point of view, that the transformations are unitary means that the computations are reversible. This phenomenon, is a consequence, not a prerequisite. The unitary operations do not increase the entropy of the system, they do not release energy to the environment. This is known as *reversibility*, and allows to do the computation backwards. It is useful to reduce the noise levels of quantum superpositions (an external excitation will increase the entropy). Unitary evolutions of a quantum state transforms superpositions into superpositions.

We will see as an example a transformation very common: the *Hadamard-Walsh* transformation.

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{3.8}$$

This transformation will be applied into the qubit $|0\rangle$ to form the initial superposition of quantum algorithms.

It is easy to verify that an operation is unitary, it must fulfill the following:

$$HH^* = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \times \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.9}$$

3.6.2 Measurement or observation

We have stated that a quantum state can evolve always that there is no interaction with an external system, this interaction makes that the superposition is lost and we do not have the uncertainly state. This process, by means of an action external to the system, making the superposition to disappear and collapse into a classic state will be the 'measurement process' or 'observation'.

Definition 12. *The result of a measurement process, or observation, of a quantum state* $|\phi\rangle = \alpha_1 |1\rangle + \alpha_2 |2\rangle + ... + \alpha_n |n\rangle$ *is one of the classic states contained* $|i\rangle$, $i \in \{0, ..., n\}$; *(each one of the classic states gets the name of observable). Each* $|i\rangle$ *has a probability* $|\alpha_i|^2$ *of being the outcome of the process.*

We will say in this case that the quantum state $|\phi\rangle$ *collapses into the observable* $|i\rangle$.

So, we observe the quantum state and as result we obtain a well defined classic state. Each classic state or observable, has an specific probability of being observed (the square of its amplitude).

This measurement process is irreversible. The concept of superposition that raises from quantum theory to explain the random behavior; but after the measurement we 'recover' our classic, and exact, knowledge of the system; the quantum phenomena has finished.

The operations and superposition of quantum states remains until the measurement process. For this reason in quantum computing, the measurement can only be the very last step, no measurement can be done as an intermediary step.

This is one of the main problems for the practical development of quantum computers. It is possible to obtain superposition by adding some energy to some fundamental particles such as electrons, but any unforeseen event, as a not well isolated atom in the surroundings of the system, will affect the superposition, and as a consequence the calculations will be wrong. For that reason a quantum error correction theory, and a well defined quantum information theory are basic steps for solving this problem.

Remark 1. *Let be the bit* $|\varphi\rangle = a|0\rangle + b|1\rangle$, *and the measurement operators* $M_0 = |0\rangle\langle 0|$ *y* $M_1 = |1\rangle\langle 1|$.

The probability of measuring 0 *is:*

$$p(0) = \left\langle \varphi \middle| M_0^\dagger M_0 \middle| \varphi \right\rangle = \langle \varphi | M_0 | \varphi \rangle = |a|^2 \tag{3.10}$$

in the same way, the probability of measuring 1 *is:*

$$p(1) = |b|^2 \tag{3.11}$$

The resulting state after the measurement, in both cases, is:

$$\begin{aligned} \frac{M_0 |\varphi|}{|a|} &= \frac{a}{|a|} |0\rangle \\ \frac{M_1 |\varphi|}{|b|} &= \frac{b}{|b|} |1\rangle \end{aligned} \tag{3.12}$$

3.7 Entanglement

One of the most difficult concepts in quantum theory is the 'entanglement'. This concept will be a problem in quantum computation rather than an advantage, but it is one of the advantages of quantum cryptography; two entangled states do not behave independently.

Intuitively the idea is that two qubits can be linked one to the other and any disturbance in one of them can be observed instantaneously in the other, independently of the distance between them. The non-local character of this property has important consequences.

We have defined the concept of qubit, and understood what is a system of n qubits. So, a quantum system of 2-qubits will be in a Hilbert's of dimension 4, $H_4 = H_2 \otimes H_2$, and its orthonormal basis will be of the type

$$\{|0\rangle|0\rangle, |1\rangle|1\rangle, |1\rangle|0\rangle, |0\rangle|1\rangle\} \tag{3.13}$$

Therefore, a quantum state in this system will be the vector

$$\{|0\rangle|0\rangle, |1\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle\} \tag{3.14}$$

with $\sum lim_{i=0}^{4}|c_i|^2 = 1$. (Remember that $|ab\rangle = |a\rangle|b\rangle$).

Remark 2. *Given two matrices $A\,(m \times n)$ and $B\,(m' \times n')$ the matrix $A \otimes B$ of size $(m \cdot m') \times (n \cdot n')$ is defined as*

$$\begin{pmatrix} a_{11} \cdot B & \dots & a_{1n} \cdot B \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot B & \dots & a_{mn} \cdot B \end{pmatrix} \tag{3.15}$$

Definition 13. *We will say that a quantum state, $z \in H_4$, cannot be decomposed unless there exists x and y in H_2 such as $z = x \otimes y$. An state is 'entangled' when it cannot be decomposed.*

An example of entangled qubits is

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{3.16}$$

because it cannot be decomposed as the product of another two.

Let's suppose that

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{3.17}$$

$$= \left(a_0|0\rangle_a + a_1|1\rangle_a\right) + \left(b_0|0\rangle_1 + b_0|1\rangle_b\right) \tag{3.18}$$

$$= a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle \tag{3.19}$$

With $a_0, a_1, b_0, b_1 \in C$. Therefore $a_0, b_0 = \frac{1}{\sqrt{2}}; a_0, b_1 = 0; a_1, b_0 = 0; a_1, b_1 = \frac{1}{\sqrt{2}}$, and this is absurd.

Remark 3. *Given a quantum state in a system of two qubits*

$$c_0|00\rangle + c_1|01\rangle + c_2|10\rangle + c_3|11\rangle \tag{3.20}$$

The result of doing a measurement will be $|00\rangle$ with probability $\left|c_0\right|^2$, $|01\rangle$ with probability $|c_1|^2$, ...If observing only one of the qubits that form the system.

The first qubit will give 0 or 1 with probability $\left|c_0\right|^2 + |c_1|^2$ or $|c_2|^2 + \left|c_3\right|^2$, because the previous expression is understood as:

$$c_0|0\rangle|0\rangle + c_1|0\rangle|1\rangle + c_2|1\rangle|0\rangle + c_3|1\rangle|1\rangle \tag{3.21}$$

If we want to find out the probability of measuring 0 in the first qubit, we will only consider the underlined coefficients:

$$c_0 \left|\underline{0}\right\rangle |0\rangle + c_1 \left|\underline{0}\right\rangle |1\rangle + c_2 |1\rangle |0\rangle + c_3 |1\rangle |1\rangle \tag{3.22}$$

resulting a probability

$$\left|c_0\right|^2 + |c_1|^2 \tag{3.23}$$

However, if we want to calculate the probability of measuring 1 *ind the second qubit, it is* $|c_1|^2 + \left|c_3\right|^2$*, because we consider:*

$$c_0 |0\rangle |0\rangle + c_1 |0\rangle \left|\underline{1}\right\rangle + c_2 |1\rangle |0\rangle + c_3 |1\rangle \left|\underline{0}\right\rangle \tag{3.24}$$

The concept of entanglement comes from the Eintein-Podolsky-Rosen paradox, that tries to raise some incoherence in quantum theory. In resume, the Eintein-Podolsky-Rosen (EPR) paradox says that if following the consequences of the quantum theory communicating between two particles will be faster than the speed of the light, and this should be impossible...

Using the previous example

$$\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \tag{3.25}$$

we can have this superposition and locate both qubits in different places. One of them can be in Earth and the other in Mars. If we measure 1 in the qubit in Earth we will measure 1 with probability $\frac{1}{2}$ and the superposition will have collapsed into $|11\rangle$. In detail, looking into the underlined parts of the expression $\frac{1}{\sqrt{2}} |0\rangle |0\rangle + \frac{1}{\sqrt{2}} \left|\underline{1}\right\rangle |1\rangle$ the square of its coefficient is the probability of measurement. The superposition collapses into $|11\rangle$.

But if after this we measure the other qubit, in Mars, we will always measure 1 with probability 1; and not with probability $\frac{1}{2}$ if it were the case that the measurement in Earth had never been done. Somehow the measurement in Earth and in Mars are interrelated. The idea is that a local phenomenon, as the measurement (a lost of superposition), modifies the whole system instantaneously; and this is what 'entanglement' refers to.

The EPR paradox, was, and still is, a discussion point in scientific community. This kind of experiments have take place in laboratories, with particles separated up to $10\,km$ with the expected result.

Those behaviors allow many opportunities for the cryptography and communication, and are linked with other phenomenon as the 'teleportation'.

3.8 Quantum parallelism

Before ending, the concept of quantum parallelism must be clarified. The strength of the quantum computation, is that we can perform operations in all the initial data at the same time, in a single computational step. This is done using superpositions.

So, we build an specific superposition and by means of unitary evolutions we obtain another superposition; if we measure this last one, it will be the answer of the algorithm.

The main idea is that each step of the computation is acting by means of an unitary evolution into a superposition. This results into a massive parallelism over all the inputs of the algorithm. Usually an

probabilistic factor is associated, during the final measurement step. This must be controlled, and keep in reasonable limits, but usually verifying the answer is very easy, and the algorithm can be executed again.

The difficulty of the quantum algorithms stays in their design, it is necessary to get some profit form the concept of superposition prepare an initial state, and perform a final measurement.

Chapter 4

Quantum Communication Channel

4.1 An overview of Quantum Information transmission

The traditional model of a communication system involves a source, a channel and a receiver. The source of information uses symbols from a finite set (source alphabet) and sends them to regular intervals. Each of the symbols transmitted is not depended on the previous ones.

Discrete channels transmit symbols form an specific set (input alphabet), and they generate in their output another set of symbols (output alphabet). As the input alphabet and the output alphabet can be different, it is necessary an encoding that maximizes the efficiency of the transmission. Basically, the encoding consists in assigning to each one of the symbols in the input an specific word (built with elements of the output alphabet).

This must be done finding a mean length for the code to be minimum, but also an unique decoding in the receiver. Also, usually the channel is not ideal, the received information differs from the sent, and this difference translate into a probability of error in the behavior of the receiver; whose mission is just to recover the original information, with the maximum possible fidelity.

In Quantum Information Theory those concepts still apply, but with some modifications. First of all, a quantum sate is associated to the symbol generated by the source. Also, the alphabet of the channel is a bidimensional Hilbert space (qubits),

It is usually also needed that the source encoding assigns, to each symbol-state a representation in qubits. In a second place, any process related with the transmission of information that alters its state, is characterized by a super-operator. But in this case, the usual thing is to include the noisy behavior of the channel, because of the interaction of the system with the environment (more ore less active), in the input alphabet.

In the error-free channels a pure state is associated to each symbol, meanwhile in the noisy channels a mixed state is used. They are closed quantum systems, composed by subsystems associated to the information (open system) and to the noisy environment.

The quality of the communication is measured in the receiver with the Fidelity function.

4.2 Shannon's entropy

Lets suppose that we have a box with 10 balls, and that nine of them are white, and one black. If we extract one ball randomly, we consider the random variable associated to the experiment as X. We quantify the result giving to it the value 0 the ball is black, and 1 if it is white.

$$\begin{aligned} P_{\{X=1\}} &= \frac{9}{10} = 0.9 \\ P_{\{X=0\}} &= \frac{1}{10} = 0.1 \end{aligned} \tag{4.1}$$

We can calculate the *expected value* of the variable X:

$$E(X) = 1 \cdot 0.9 + 0 \cdot 0.1 = 0.9 \tag{4.2}$$

But this is not any surprise: the expected value, is the mean value expected after several repetitions of the same experiment. As a mean, 9 of 10 times we obtain 1 (while ball), therefore the expected value is 0.9.

There are only two possible values for X, but its associated information is quite different.

If we review the definition of amount of information associated to an event i:

$$I_i = -log_2(P(x_i)) \tag{4.3}$$

In our case, we obtain:

$$\begin{aligned} I_1 &= -log_2(0.9) = 0.15 \\ I_0 &= -log_2(0.1) = 3.32 \end{aligned} \tag{4.4}$$

We can see that the information associated to 'extracting a black ball' ($X = 0$) is much higher than the information associated to the event of 'extracting a white ball'. This can be understood as that extracting the black ball reduces all the uncertainly of next extractions: all the remaining ones are white. If we had extracted a white ball, the reduction of uncertainly is much lower: we cannot assure what is going to happen in the following extraction.

So, having a random variable implies not knowing exactly the outcome of the value, and this implies not knowing the amount of information acquired after the experiment, because each possible result has associated a different amount of information.

This simple idea allows us to define from a random variable, another associated random variable, that is the amount of information to be obtained in one experiment.

Definition 14. *Given a random variable, X, that takes values $\{x_1, x_2, \dots, x_n\}$ with probabilities $p_1, p_2, \dots, p_n$, that provide the amounts of information $I_1, I_2, \dots, I_n$, we define* ***random variable, (amount of information associated to*** X***)*** *to the random variable $I(X)$, that takes the values $I_1, I_2, \dots, I_n$ with probabilities $p_1, p_2, \dots, p_n$.*

In our example, $I(X)$ takes the value 0.15 with probability 0.9, and the value 3.32 with probability 0.1.

An important property of this new variable is that, although it comes form the initial X, is not dependent on the numerical values X could take. It relies only in the probability distribution of those values.

In this point, and as $I(X)$ is just another random variable, anything stops us asking for it expected value, $E(I(X))$; and we note it with $H(X)$

$$H(X) = E(I(X)) = 0.15 \cdot 0.9 + 3.32 \cdot 0.1 = 0.135 + 0.332 = 0.467 \tag{4.5}$$

We see that the contribution of the less probable event is higher than the most probable one, although the amounts of information are multiplied by its corresponding probability.

We agree that an even with probability zero, does not affect this calculation. This boundary is needed because the product $p(x) \cdot log(P(x))$ is indeterminate when $p(x) = 0$.

What we have obtained with this definition?

We have a real number, $H(X)$, that is the expected value of the amount of information that we get from the result of the experiment defined by that random variable.

This value is called **Shannon's entropy** of the given variable; and is one of the main concepts in information theory.

The analytic expression of **Shannon's entropy** is the following:

$$\left.\begin{array}{c} H(X) = (E)\, I(X) = \sum lim_{i=1}^{n} p(x_i) \cdot i(x_i) \\ i(x_i) = -log_2\left(p(x_i)\right) \end{array}\right\} \rightarrow H(X) = -\sum lim_{i=1}^{n} p(x_i) \cdot log_2\left(p(x_i)\right) \tag{4.6}$$

4.3 Von Neumann entropy

4.4 Dense codification

$$\begin{array}{c} I \otimes I \left|\psi^-\right\rangle = \left|\psi^-\right\rangle \\ \sigma_z \otimes I \left|\psi^-\right\rangle = \left|\psi^+\right\rangle \\ \sigma_x \otimes I \left|\psi^-\right\rangle = -\left|\phi^-\right\rangle \\ \sigma_y \otimes I \left|\psi^-\right\rangle = i\left|\phi^+\right\rangle \end{array} \tag{4.7}$$

$$\begin{array}{c} \left|\phi^\pm\right\rangle = \dfrac{1}{\sqrt{2}}\left(|00\rangle \pm |11\rangle\right) \\ \left|\psi^\pm\right\rangle = \dfrac{1}{\sqrt{2}}\left(|01\rangle \pm |10\rangle\right) \end{array} \tag{4.8}$$

4.5 Classical channel without noise

4.5.1 Definitions

Capacity of a classical channel

In a discrete channel without memory, the capacity of the channel is defined as:

$$C = maxlim_{p(x)} I(X, Y) \tag{4.9}$$

The function $I(X,Y)$ gives us the information what we know form the input symbol X according to each output symbol Y, this means, the information we manage to send to the output from the input each time that we use the channel. The remaining part of information from X, $H(X/Y) = H(X) - I(X,Y)$, that we need to know we will have to 'estimate', and in this estimation will take place the error.

As $I(X,Y)$ depends of the source, that produces one or another distribution of X, we define the capacity abstracting the fact that the channel could be connected to different types of sources; and the relation between X and Y for the source is the one that uses best the channel (the one that allows to transmit more information for a single symbol of the channel).

Binary channel without noise

$$1 \xrightarrow{1} 1$$
$$0 \xrightarrow{1} 0$$

$$\begin{aligned} A_x &= \{0,1\} \\ A_y &= \{0,1\} \end{aligned} \tag{4.10}$$

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4.11}$$

In this

$$X = f(Y) \rightarrow I(X;Y) = H(X) \tag{4.12}$$

and then:

$$C = maxlim_{p(x)} I(X,Y) = maxlim_{p(x)} H(X) = 1 \frac{\text{bit}}{\text{symbol}} \tag{4.13}$$

This maximum is obtained for an uniform distribution of X. How the channel is error-free, it is logic that each symbol of the channel, can transmit the maximum amount of information, that for binary symbols in this type is $1\,bit$. In an ideal channel we can transmit information with zero error probability and without any redundancy.

Noisy coding machine

$$Q = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & \dots & 0 & \frac{1}{2} \end{pmatrix} \tag{4.14}$$

The capacity of this channel is:

$$C = maxlim_{p(x)} (H(Y) - H(Y/X)) = maxlim_{p(x)} (H) X - 1 \tag{4.15}$$

ya que para cada x belongs to A_X:

$$H(Y/X) = 1\text{bit} \rightarrow H(Y/X) = \sum lim_x p(x) H(Y/X = x) = \sum lim_x p(x) = 1 \tag{4.16}$$

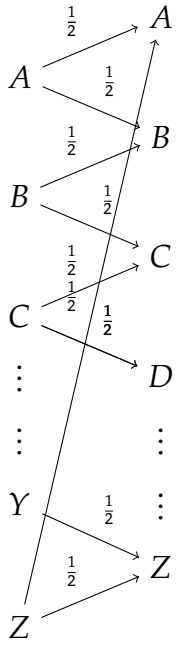

Figure 4.1: Noisy writing machine

As $P(Y = letra_i) = \frac{1}{2}P(X = letra_i) + \frac{1}{2}P(X = letra_{i\text{-}1})$ it always verifies

$$\begin{gathered} \forall i P(Y = letra_i) = \frac{1}{26} \\ \Rightarrow \forall i P(Y = letra_i) = \frac{1}{26} \Rightarrow maxlim_{p(x)} H(X) = log_2(26) \\ \Rightarrow C = log_2 26 - 1 = log_2(13) \frac{\text{bits}}{\text{symbol of the channel}} \end{gathered} \tag{4.17}$$

que se consigue para X evenly distributed.

Obervations

1. $H(Y/X) = 1$ represents in the expression of the capacity the bit that we loose because of the total uncertainly that we have over the input symbol is the received letter or the previous one.
2. We can transmit with fidelity the half of a 26-ary (a 13-ary symbol) each time we use the channel. We can only know in the output half of the information from X, and the older half will be dependent on error $\frac{1}{2}$. Int this case if the reception is B, there is certainly that the symbol sent was A or B, but there is only half of the chances to know if it was A or B. 13 subsets of inputs can be distinguished, that we can assign to 13 different symbols, but it is not possible to distinguish form the 26 letters directly.

Cadence of a coding machine

The cadence of a coding machine, R, is defined as,

$$R = \frac{log_2(M)}{n} \frac{\text{bits}}{\text{symbol of the channel}} \tag{4.18}$$

where M is the number of symbols in the input of the receiver, and n is the length of the words at the output of the receiver.

This parameter measures the efficiency of a coder, giving the number of bits per output symbol what we would obtain in our source if this source would produce M evenly distributed symbols (a maximum entropy source; source with maximum information). In general, any channel, for an input sequence we have an outputs of the type Y^n. All those sequences have the same probability, because the input are sequences of the type X^n. For decoding we have only to group the possible sequences Y^n of those sets, and make an estimation of the input symbol for that set. And according to the principle of evenly distribution, this estimation would have an irrelevant P_e. For building a decoder, the sets must be disjoints; because if not, we cannot decide with the same criteria.

The property of evenly distribution states that each sequence X_1^n sent through the channel, will have $2^{n\left(H\left(Y/X_n=X_1{}^n\right)\right)}$ possible outputs. As a mean, each set of outputs will have $2^{nH(Y/X)}$ elements, and there will be $2^{nH(Y)}$ possible output combinations in total. In this case for encoding W symbols we have to find W disjoint groups of sequences.

$$W \leq \frac{2^{nH(Y)}}{2^{nH(Y/X)}} = 2^{n(H(Y)-H(Y/X))} = 2^{nI(X,Y)} \tag{4.19}$$

We can only send $2^{nI(X;Y)}$ sequences of length n that can be distinguished at the output with $P_e^{(n)}$ as low as desired, if we have a sender (coder) that verifies

$$W = 2^{nI(X,Y)} \Rightarrow R = I(X,Y) \tag{4.20}$$

These relations shows that we cannot send with a cadence higher than the mutual information between X and Y if we want an error-free communication. We have therefore a compromise between the speed of the transfer (a high R, or information compression), and an error-free communication.

(M,n) codes

A code (M,n) for a channel $\{A_x, p\left(y|x\right), A_y\}$ is:

1. An index $W = \{1,2,\dots,M\}$
2. A coding function $X^n(L)$: That takes an element from the index and generates a word from the input

$$\{1,2,\dots,M\} \xrightarrow{X^n} A_X^n$$
$$1 \longrightarrow X^n(1)$$
$$2 \longrightarrow X^n(2)$$

   alphabet of the channel.
3. A decoding function Y^n ; A_Y^n that sets the values $\{1,2,\dots,M\}$

Error probability

$$\lambda_i = Pr\left(g(Y^n) \neq \frac{i}{X^n} = X^n(i)\right) = \sum lim_{y^n} p\left(\frac{y^n}{X^n(i)}\right) I\left(g\left(y^n\right) \neq i\right) \tag{4.21}$$

Where the function I gives value 1 when the condition is satisfied, and 0 when it is false.

Remark 4. *$P_e^{(n)}$ in general, is different to the probability error of each symbol of the source, because the probability of error of a symbol depends on the probability of that symbol. A source verifies $P_e^{(n)} = P_e$ only when satisfying the two following conditions:*

1. *The source of evenly distributed symbols $\{1, 2, \dots, M\}$*
2. *The error probability of all the symbols is the same.*

Definition 15. *For a given channel we state; The **cadence** is **reachable** if there exists a code $C\left(ceil\left(2^{nR}\right), n\right)$ such as:*

$$lim_{n\to\infty}\lambda^{(n)} = 0 \tag{4.22}$$

Remark 5. *If R is reachable we can build a receiver with a cadence R for sending the true values with $P_e^{(n)}$ as low as we want.*

4.5.2 Shannon coding theorem

1. Every $R < C$ is reachable (Direct Theorem)
2. Every $R > C$ is not reachable(Inverse Theorem)

Inverse Theorem

Let it be a code $C(M, n)$ con $M = ceil\left(2^{nR}\right)$

It can be proved that we cannot build a code with error probability as low as we wish if we do not respect the limitation imposed by $R < C$. If we suppose that we are using the channel to transmit symbols with the same probability, $p(i) = p\left(j\right)$ for every i, j belonging to W (code index).

With this hypotheses we can calculate:

If $n \to \infty \Rightarrow \frac{logM}{n} = \frac{log\left|2^{nR}\right|}{n} \approx R$ and in all cases (Cadencia de C) $\leq R$

$$\begin{gathered} nR = H(W) = H(W/Y^n) + I(W/Y^n) \leq H(W/Y^n) + I\left(X^n(W), Y^n\right) \\ \Rightarrow \overset{2}{lim}\, nR \leq 1 + P_e^{(n)} nR + I\left(X^n(W), Y\right) \leq \overset{3}{lim} 1 + P_e^{(n)} nR + nC \end{gathered} \tag{4.23}$$

1. Theorem of information processing
2. Following the Fano inequality we obtain

$$H\left(\frac{W}{Y^n}\right) \leq H\left(P_e^{(n)}\right) + P_e^{(n)} log(M-1) \leq 1 + P_e^{(n)} log\left(\left|2^{nR} - 1\right|\right) \leq 1 + P_e^{(n)} nR \tag{4.24}$$

3. Because they are channels without memory and without return.

$$\begin{gathered} I\left(X^n, Y^n\right) = H\left(Y^n\right) - H\left(Y^n/X^n\right) = H\left(Y^n\right) - \sum lim_{i=1}^{n} H\left(Y_i/Y_1, \dots, Y_{i-1}, X^n\right) \\ \Rightarrow I\left(X^n, Y^n\right) = H\left(Y^n\right) - \sum lim_{i=1}^{n} H\left(Y_i/Y_1\right) \leq \sum lim_{i=1}^{n} H\left(Y_i\right) - \sum lim_{i=1}^{n} H\left(Y_i/Y_i\right) \\ \Rightarrow I\left(X^n, Y^n\right) = \sum lim_{i=1}^{n} H\left(Y_i/Y_1\right) \leq max\left(I(X, Y)\right) = nC \end{gathered} \tag{4.25}$$

Clearing $P_e^{(n)}$ de $nR \leq 1 + P_e^{(n)} nR + nC$ we obtain:

$$P_e^{(n)} \geq \frac{nR - nc - 1}{nR} = 1 - \frac{c}{R} - \frac{1}{nR} \tag{4.26}$$

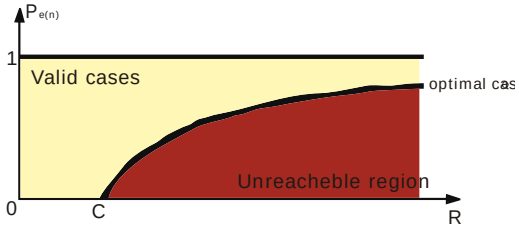

Figure 4.2: Regions diagram

Given a channel with capacity C, la $P_e^{(n)}$ this region is limited by the expression, that we can draw:

So if we try to build a code with cadence $R > C$ over a channel of capacity C we will be never able to obtain $P_e^{(n)}$ approaching to zero.

If we want that $P_e^{(n)}$ becomes zero when n grows up to the infinity it is necessary that $C > R$.

Direct Theorem

It is possible to buld a code for a channel with capacity C with $P_e^{(n)}$ as low as we wish, if $R < C$.

4.6 Quantum channel without and with noise

4.6.1 Fidelity

Let's suppose that we want to send trough a quantum channel a pure state described by the density operator $\widehat{\rho} = |\varphi\rangle\langle\varphi|$. The behavior of the channel is not ideal, and is characterized by the super-operator \$, and can produce an alteration of the symbol associated to $\widehat{\rho}$:

$$\widehat{\rho}' = \$\left(\widehat{\rho}\right) \tag{4.27}$$

Definition 16. *For this reason it is necessary to have a quantification of these unwanted alterations in the physical system. This means, to calculate the probability of receiving, the same symbol that was sent in the channel. This is done using the fidelity function, that is defined as:*

$$F\left(\widehat{\rho}, \widehat{\rho}'\right) = \langle\varphi|\,\widehat{\rho}'\,|\varphi\rangle \tag{4.28}$$

Obviously, as closer this $F\left(\widehat{\rho}, \widehat{\rho}'\right)$ it is to 1, better would have been the transmission. As has been been defined the fidelity function, its application is limited to pure states. For the case of mixed states, the formalism is more complicated:

$$F\left(\widehat{\rho}, \widehat{\rho}'\right) = tr^2\left|\left(\sqrt{\widehat{\rho}\widehat{\rho}'}\sqrt{\widehat{\rho}}\right)^{1/2}\right| \tag{4.29}$$

where we suppose that $\widehat{\rho}$ and $\widehat{\rho'}$ are mixed states.

The previous equation can be rewritten as

$$F\left(\widehat{\rho}, \widehat{\rho}'\right) = max\left|\langle\phi|\,\phi'\rangle\right|^2 \tag{4.30}$$

with $|\phi\rangle$ y $|\phi'\rangle$ being purifications of $\widehat{\rho}$ and $\widehat{\rho'}$.

Due to the importance of the fidelity function, some properties are listed without proof:

1. $0 \leq F\left(\widehat{\rho}, \widehat{\rho'}\right) \leq 1$, where the equality, $F\left(\widehat{\rho}, \widehat{\rho'}\right) = 1$, is verified if and only if $\widehat{\rho} = \widehat{\rho'}$.
2. $F\left(\widehat{\rho}, \widehat{\rho'}\right) = F\left(\widehat{\rho'}, \widehat{\rho}\right)$.
3. Given two number positive reals, p_1 y p_2, such as $p_1 + p_2 = 1$, then:

$$F\left(\widehat{\rho}, p_1\widehat{\rho}_1 + p_2\widehat{\rho}_2\right) \geq p_1 F\left(\widehat{\rho}, \widehat{\rho}_1\right) + p_2 F\left(\widehat{\rho}, \widehat{\rho}_2\right) \tag{4.31}$$

   Also:

$$F\left(\widehat{\rho}, \widehat{\rho}'\right) \geq \mathrm{tr}\left(\widehat{\rho}\widehat{\rho}'\right) \tag{4.32}$$

4. If $\widehat{\rho}$ is a pure state $\widehat{\rho} = \left|\varphi\right\rangle\left\langle\varphi\right|$, then:

$$F\left(\widehat{\rho}, \widehat{\rho}'\right) = \left\langle\varphi\right|\widehat{\rho}'\left|\varphi\right\rangle = \mathrm{tr}\left(\widehat{\rho}\widehat{\rho}'\right) \tag{4.33}$$

5. $F\left(\widehat{\rho}_1 \otimes \widehat{\rho}_2, \widehat{\rho}_3 \otimes \widehat{\rho}_4\right) = F\left(\widehat{\rho}_1, \widehat{\rho}_3\right) F\left(\widehat{\rho}_2, \widehat{\rho}_4\right)$
6. Any measurement that transforms the states $\widehat{\rho_1}$ and $\widehat{\rho_2}$ in, respectively, $\widehat{\rho'_1}$ and $\widehat{\rho'_2}$, verifies:

$$F\left(\widehat{\rho}'_1, \widehat{\rho}'_2\right) \geq F\left(\widehat{\rho}_1, \widehat{\rho}_2\right) \tag{4.34}$$

4.6.2 Schumacher coding theorem

Let it be a source that produces pure quantum states (belonging to a n-dimensional Hilbert's space) or a set not necessarily orthogonal, $\left\{\left|\varphi_1\right\rangle, \left|\varphi_2\right\rangle, \ldots, \left|\varphi_x\right\rangle\right\}$, with probabilities $p_1, p_2, \ldots, p_x$.

The symbol generated by a source is characterized by the following density operator:

$$\widehat{\rho} = \sum lim_x p_x \left|\varphi_x\right\rangle\left|\varphi_x\right\rangle \tag{4.35}$$

In the same way, a sequence composed of N symbols is described, using the tensorial product formalism, as

$$\widehat{\rho}^N = \widehat{\rho} \otimes \ldots \otimes \widehat{\rho} \tag{4.36}$$

Schumacher proved that give any two positive real numbers, δ y ε, and a sequence of N symbols (N enough big), it is always possible to encode, as a mean, each state generated by the source with $S\left(\widehat{p}\right) + \delta$ qubits, keeping the fidelity as high as wished ($F > 1 - \delta$).

Also, if using $S\left(\widehat{p}\right) - \delta$ qubits, the fidelity is never higher to ε. This means that $S\left(\widehat{p}\right)$ qubits represents the contents in quantum information, as a mean, of each symbol of the source, and constitutes the best possible encoding.

Observing the results, it is evident the similarity between the Shanon theorem and the Schumacher theorem. However, the quantum theory allows a higher compression of information. How comes this possibility?

The answer is the impossibility to distinguish perfectly between non-orthogonal quantum states. When the source alphabet are orthonormal kets, the quantum an classic theory coincide, meanwhile if they are different, the intrinsic not determinism of those states, makes some classic information in redundant, and this has as consequence a higher rate of compression.

4.6.3 Quantum channel without noise

Let's suppose that we want to send the classic information generated by a source, whose entropy is $H(X)$, through a error-free quantum channel. Obviously, for seceding in the transmission it is necessary to use a

quantum source. A possible solution is to prepare a bijective application among the symbols of the original input alphabet, with their associated probabilities and a set of quantum states $\left\{\left|\varphi_x\right\rangle\right\}_x$.

But, how good this solution? How much is reduced the uncertainly of X in the case of the best measurement in the received quantum state? For answering to those questions, we use the concept of accessible information, Acc (ε), that is defined as the maximum information (in bits) that can be recovered, by using the results of a measurement that maximizes that information stored in a quantum state,

$$\text{Acc}\,(\varepsilon) = maxx_{\left\{\widehat{F}_y\right\}} I\,(X,Y) \tag{4.37}$$

where $\left\{\widehat{F}_y\right\}$ is an operator Y is the random variable produced by the results of the measurement and its associated probabilities, and $I\,(X,Y)$ represents the classic mutual information between X and Y.

It can be proof that if $\left\{\left|\varphi_x\right\rangle\right\}_x$ is orthogonal. The best of the possible von Neumann measurements is $\widehat{P_y} = \left|\varphi_x\right\rangle\left\langle\varphi_x\right|$. In this way, the value a_y determines perfectly the preparation of the system, because

$$p\left(\frac{a_y}{x}\right) = \delta_{xy} \tag{4.38}$$

where is stated the perfect concordance with the established by Shannon in the classic theory for an error-free channel, $H\,(X/Y) = 0$ e $I\,(X,Y) = H\,(X)$.

However, it is much interesting analyze the case for a source alphabet consisting of pure states not mutually orthogonal. Although there is no expression for Acc (ε), an upper limit is known:

$$\text{Acc}\,(\varepsilon) \leq S\left(\widehat{\rho}\right) \tag{4.39}$$

Moreover, it can be proved, that using a proper coding and decoding, it is always possible to get asymptotically close to that limit. This means, that even if it is a error-free channel, due to the fact of being unable of perfectly distinguish the not orthogonal states, there is a degradation of classic information. The accessible information can never be more than the von Neumann entropy, that is lower than $H\,(X)$.

4.7 Quantum channel with noise

The analysis of accessible information in a noisy quantum channel es almost identical to the one without noise. The unique difference is that now it is allowed that the alphabet of the quantum source is an statistic mix. The decoherence effect transforms pure states in mixed states.

A noisy quantum channel can be modeled with a super-operator, \$, that acts directly in the symbols of the source alphabet,

$$\left|\varphi_x\right\rangle\left|\varphi_x\right\rangle \rightarrow \$\left(\left|\varphi_x\right\rangle\left\langle\varphi_x\right|\right) = \widehat{\rho}_x \tag{4.40}$$

It can be proved that in this case, and for channels without memory, that the upper limit of accessible information, that now is called Holevo limit, it is formed by the Holevo information of the set $\left\{\widehat{\rho}_x, \widehat{\rho}_y\right\}$

$$\text{Acc}\,(\varepsilon) \leq \chi\,(\varepsilon) \tag{4.41}$$

Also, and following the idea the previous section, it is always possible, using some proper encoding and decoding, approach asymptotically to the value $\chi\,(\varepsilon)$.

In this way, and following the equation

$$\chi\,(\varepsilon) \leq S\left(\widehat{\rho}\right) \leq 1 \tag{4.42}$$

it is deduced that for the case of quantum channels without noise, as for the noise ones, the maximum information that can be stored in a qubit is a bit.

4.8 Quantum states vs accessible information

4.8.1 Holevo limit

Let it be a quantum source, but instead of producing pure states, it produces an statistical mix $\widehat{\rho}_x \left(S\left(\widehat{\rho}_x\right) \neq 0\right)$ with probability 1. According to Schumacher's theorem, it would be necessary to transmit $S\left(\widehat{\rho}_x\right)$ qubits per symbol. But this is not reasonable, because it is a deterministic source, and it is not sending any information to the receiver. For understanding this difference of von Neumann entropy, the Holevo information concept is proposed. It is simply a generalization of the previous theorem, because is establish the minimum number of qubits needed to encode the symbols generated by the source (with a fidelity as low as desired), but for the case of pure states and mixed states.

Let's suppose now that the source sends symbols characterized by the density operator

$$\widehat{\rho} = \sum lim_x p_x \left|\varphi_x\right\rangle \left\langle\varphi_x\right| \tag{4.43}$$

The definition and notation for Holevo's information is therefore:

$$\chi\left(\varepsilon\right) \tag{4.44}$$

as a function that acts over the quantum state of the system according to the expression

$$\chi\left(\varepsilon\right) \leq S\left(\widehat{\rho}\right) - \sum lim_x p_x S\left(\widehat{\rho}_x\right) \tag{4.45}$$

The fact of subtracting the term $\sum lim_x p_x S\left(\widehat{\rho}_x\right)$ to $S\left(\widehat{\rho}_x\right)$ has a similar interpretation to the concept of mutual information in classic theory: the knowledge of the specific preparation of $\widehat{\rho}$ necessarily implies a reduction of the von Neumann's entropy. According to its meaning (the mean of quantum information stored in each symbol of the source), it is an amount that cannot be negative. It is straightforward to proof

$$S\left(\sum lim_x p_x \widehat{\rho}_x\right) \geq \sum lim_x p_x S\left(\widehat{\rho}_x\right) \tag{4.46}$$

Also, it can be proof that the action of a super-operator never can increase Holevo's information:

$$\begin{gathered} \$: \varepsilon = \left\{\widehat{\rho}_x, p_x\right\} \rightarrow \varepsilon' = \left\{\$\left(\widehat{\rho}_x\right), p_x\right\} \\ \chi\left(\varepsilon'\right) \leq \chi\left(\varepsilon\right) \end{gathered} \tag{4.47}$$

This means, a channel can keep or reduce the amount of information stored in a symbol, but anyhow to increase it.

4.8.2 Classical capacity of a quantum channel

Given a quantum channel without memory and characterized by the super-operator \$, its classic capacity is defined as the maximum amount of information (in bits) what can be transmitted with a probability of error as low as desired. As the mutual information is convex with respect to the probability distribution of the source, the capacity is simply its maximum,

$$C = max_{\left\{p_x, \widehat{F}_y\right\}} I\left(X, Y\right) \tag{4.48}$$

So considering Holevo's limit, and using a proper encoding, it is possible to approach asymptotically to its value, and the previous equation can be rewritten as

$$C\left(\$\right) = max_{\{\varepsilon\}} \chi\left(\$\left(\varepsilon\right)\right) \tag{4.49}$$

where have been included, by using $, the possible alterations induced by the channel.

Eventually, it is needed to remark that all results in this chapter suppose that the encoding of sequence of symbols produced by the source do not use entangled states. In fact, the possibility of obtaining higher data transfers, by using this kind of states, is a open question.

Chapter 5

Physical implementation of Quantum Information systems

5.1 Theoretical models of quantum information systems

5.1.1 Light system

The light pulse which travels through the atomic sample consists of two parts. Each of these have the same duration and spatial profile. The two polarization degrees of freedom are used for different purposes. The interesting one is a weak, y-polarized part containing only a small number of photons (or it may be the vacuum state). This is the quantum field to be stored in the atomic memory. The x-polarized part is strong and is providing the interaction strength required for the light and atomic states to communicate.

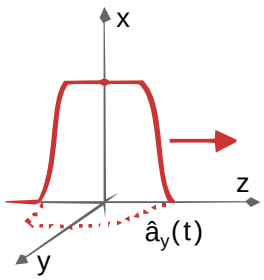

Figure 5.1: Strong light pulse

It is convenient to express the light quantum variables in the language of position- and momentum-like operators X and P satisfying the usual commutation relation $[X, P] = i$:

The quantum variables X,P for light are defined below. T is the pulse duration and the creation and annihilation operators in the integrals refer to the y-polarized mode of light.

$$\hat{X}_L = \frac{1}{\sqrt{2T}} \int_0^T \left(\hat{a}_y^t + \hat{a}_y\right) dt$$
$$\hat{P}_L = \frac{i}{\sqrt{2T}} \int_0^T \left(\hat{a}_y^t - \hat{a}_y\right) dt \tag{5.1}$$

5.1.2 Atomic system

In the atoms it is the spin degree of freedom that is relevant. We use cesium atoms in the $F = 4$ ground state and define collective spin operators J_x, J_y, and J_z as sums over all individual spin components.

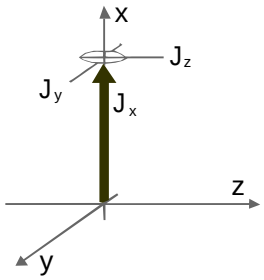

Figure 5.2: Spin components

The atomic sample consists of roughly 1011 atoms. By optical pumping it is possible to initially orient all atoms along the x-direction with an efficiency of 99%. Hence J_x has a very large mean value and its fluctuations are unimportant. The interesting quantum degrees of freedom are the transverse components J_y and J_z. Hence we define quantum variables X and P for atoms as below:

The quantum variables X,P for atoms. The atoms are spin polarized along the x-direction and J_x is effectively a classical c-number.

$$\hat{X}_A = \frac{\hat{J}_y}{\sqrt{J_x}}$$
$$\hat{P}_A = \frac{\hat{J}_z}{\sqrt{J_x}} \tag{5.2}$$

5.1.3 Interaction and memory protocol

In the interaction process the light and atomic quantum variables will affect each other as described below. The interaction strength k depends on the strength and duration of the light in the x-polarized mode, on the number of atoms in the sample, and on laser and geometrical settings.

The equations of interaction for the light and atomic quantum variables; ‘in’ and ‘out’ refer to the state before and after the light and atoms have interacted. The interaction strength is parameterized by the dimensionless parameter k which is of the order of unity in the experiment.

$$\left\langle \hat{X}_L^{out} \right\rangle = \left\langle \hat{X}_L^{in} \right\rangle + k \left\langle \hat{P}_A^{in} \right\rangle$$
$$\left\langle \hat{P}_L^{out} \right\rangle = \left\langle \hat{P}_L^{in} \right\rangle \tag{5.3}$$

$$\left\langle \hat{X}_A^{out} \right\rangle = \left\langle \hat{X}_A^{in} \right\rangle + k \left\langle \hat{P}_L^{in} \right\rangle$$
$$\left\langle \hat{A}_L^{out} \right\rangle = \left\langle \hat{A}_L^{in} \right\rangle \tag{5.4}$$

5.2 Experimental setup

In the macroscopic classical world, it is possible to copy information from one device into another. We do this everyday, when, for example, we copy files in a computer or we tape a conversation. In the microscopic world, however, it is not possible to copy the quantum information from one system into another one. It can only be transferred, without leaving any trace on the original one. The manipulation and transfer of quantum information is, in fact, a very active field of research in physics and informatics, since it is the basis of all the protocols and algorithms in the fields of quantum communication and computation, which may revolutionize the world of information. In the work published in Nature, November 25, 2004, Ignacio Cirac and other scientists from the Max Planck Institute for Quantum Optics in Garching and the Niels Bohr Institute in Copenhagen have proposed a scheme to transfer the quantum state of a pulse of light onto a set of atoms and have demonstrated it experimentally.

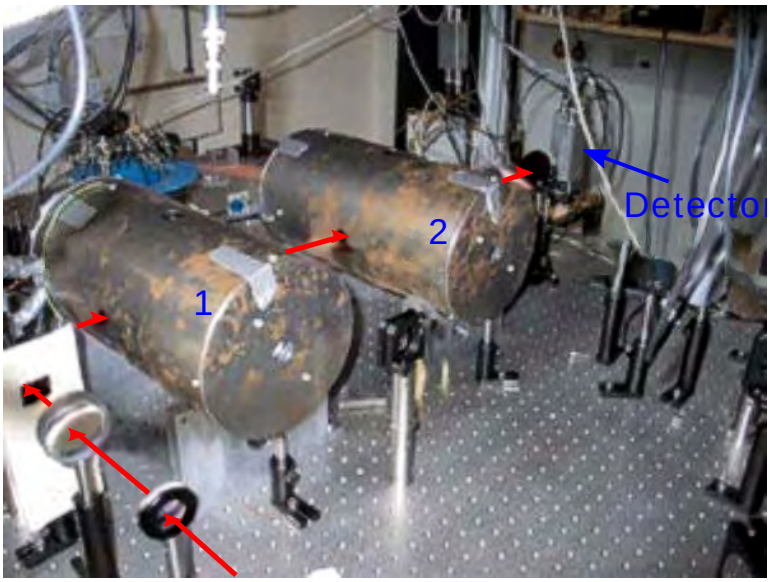

Figure 5.3: Atomic memory unit consisting of two caesium cells inside magnetic shields 1 and 2. The path of the recorded and readout light pulses is shown with arrows.

In the experiment, a pulse of light is prepared in a certain quantum state whose properties (polarization) are randomly chosen. Then, the light is sent through a set of atoms which are contained in a small transparent box (an atomic cell) at room temperature. In the cell, the light and atoms interact with each other, giving rise to an 'entangled' state in which the two systems remain correlated. After abandoning the atomic sample, the pulse of light is detected. Due to the fact that the light and atoms are entangled, the process of measurement on the light affects the quantum state of the atoms in such a way that they acquire the original properties of the light. In this way, the state of polarization of the photons is transferred into the polarization state of the atoms. This 'action at a distance', in which by performing a measurement on a system it affects the state of another system which is at a different location is one of the most intriguing manifestations of Quantum Mechanics, and is the basis of applications such as quantum cryptography or phenomena like teleportation.

In order to check that the transfer of polarization has indeed taken place, the researcher measured the polarization of the atoms at the beginning of the experiment and compared it with the original state of polarization of the light. In the experiment, these two polarizations coincided up to a 70% of the time. The main reason for the imperfections where the due to spontaneous emission, a process in which the atoms absorb the photons but then emit them in a different direction such that they do not go towards the photodetector.

A question that the authors of the paper had to carefully analyze was to what extent 70% percent of coincidence is enough to claim that the process was successful. Or, in other words, could they obtain the same result by measuring the state of polarization of the photons and then preparing the state of the atoms accordingly? The answer is no. Due to the basic properties of quantum mechanics, the state of polarization of a laser pulse cannot be fully detected. Due to the Heisenberg uncertainty principle, it is impossible to measure the full polarization exactly. In fact, as some of the authors together with K. Hammerer and M. Wolf (from the Max Planck Institute of Quantum Optics) have recently shown, the best one can do using this latter method would be 50%. This implies that the experiment indeed has successfully demonstrated the transfer beyond what one could do without creating the entangled state.

The current experiment paves the way for new experiments in which the information contained in light can be mapped onto atomic clusters and then back into the light again. In this way, one could not only store the state of light in an atomic clusters, but also retrieve it. This process will be necessary if we want to build quantum repeaters, that is, devices which will allow the extension of quantum communication far beyond the distances (of the order of 100 km) which are achieved nowadays.

To understand why the experiment is important, let us discuss the limitations Nature puts on us when we wish to store some information carried by light into atoms. First of all, as Niels Bohr pointed out many years ago, there is the so called complementarity principle in quantum mechanics (and hence in Nature). This principle states that certain properties of a physical system cannot be measured (or even defined) precisely at the same time. For a light wave the strength and the phase are complimentary. As a consequence, if we encode information into the strength and the phase of a light beam, this information can only be retrieved again with a certain precision. This lack of precision materializes as noise the so called quantum noise.

However, we might not want to retrieve the information from the light. Rather we could be interested in storing the exact information in a memory unit. If this storage procedure works with a precision better than the precision with which we could ever measure the light state directly, we have a so called 'quantum memory'.

The inset shows the pulse sequence for the quantum memory recording and readout:

1. Pulse (1) is the optical pumping (4 ms)
2. Pulse (2) is the input light pulse $\hat{a}(t)$ overlapped with the strong entangling pulse in orthogonal polarization with amplitude $\sqrt{n(t)}$.
3. Pulse (3) is the magnetic feedback pulse
4. Pulse (4) is the magnetic $\frac{p}{2}$ pulse used for the read out of one of the atomic operators.
5. Pulse (5) is the readout optical pulse.

5.3 Storing the quantum state

5.3.1 How the quantum memory works

In the picture below we see a part of the experiment. An atomic sample of cesium gas contained in a paraffin coated glass cell is placed inside a magnetic shielding (the cylinders) to prevent unwanted external fields from disturbing the experiment. The atoms can be accessed by lasers, and the outgoing laser light can be

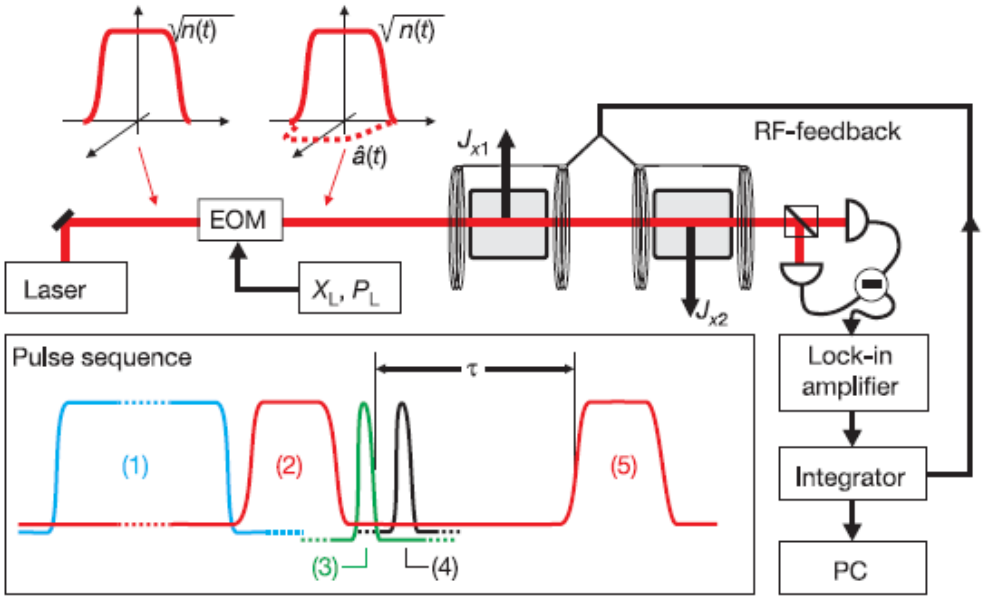

Figure 5.4: Experiment diagram

measured on photo-detectors. We have the possibility to affect the atoms via magnetic fields by sending currents through a set of coils close to the atoms.

The quantum memory protocol runs as follows:

1. An unknown input state of light is sent to the atoms. The strength and phase of this light is not exactly defined which is symbolized by the thickness of the line in the drawing of the light wave.
2. When light passes the atomic sample there is an exchange of information between light and atoms. A part of the light is stored in atoms (e.g. the phase information). At the same time the atoms act back on the outgoing light. In this process light and atoms become entangled.
3. The outgoing light is detected (e.g. the amplitude part). The obtained result both carries information about the incoming light amplitude and information about atoms.
4. With a feedback system the atoms are rotated by an amount conditioned on the measurement result. From the beginning the atoms contained quantum noise just as the input light, but the fact that light and atoms became entangled in step 2 enables us to cancel the information about atoms in the outgoing light with the initial quantum noise of atoms. The result is a storage of the incoming light state in the atomic system.

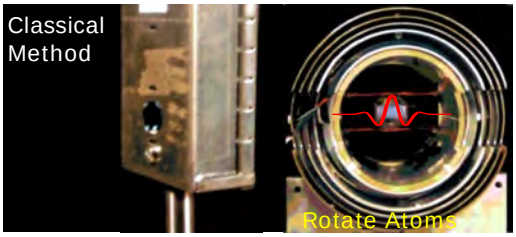

Figure 5.5: Input state

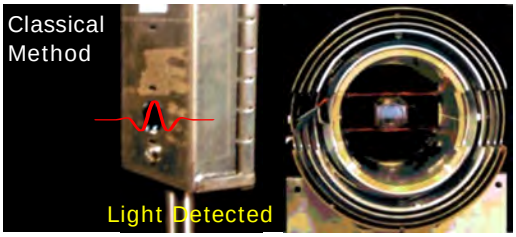

Figure 5.6: Imprint left

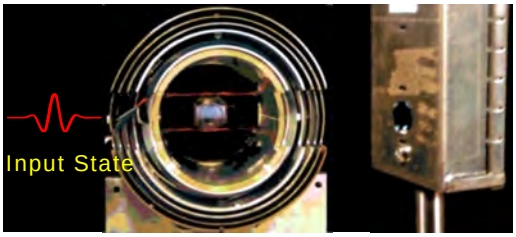

Figure 5.7: Feedback

5.3.2 Comparison with a classical protocol

A classical memory performs a measurement on the light directly which is shown in the picture below:

1. We are given an input state of light as before with some line thickness symbolizing the quantum noise.
2. The complementarity principle prevents us from detecting both the strength and phase of the light at the same time. We can only obtain a half-good result of the two (the measurement result is symbolized with a double thickness wave).
3. To store the measured information in atoms (or any other appropriate physical system) we must rotate atoms according to our measurement. But the atoms also contain initial noise and as a result we have tripled the noise level in the procedure.

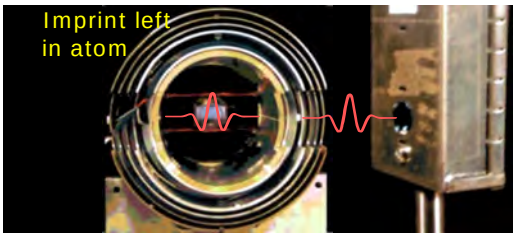

Figure 5.8: Rotate atoms

In conclusion, a classical memory cell will always add two units of quantum noise in addition to the initial one unit of quantum noise in the input state. A quantum memory can in principle work without adding extra noise. In our experiment we have created a memory cell which adds less noise than a classical memory cell. Hence we have demonstrated quantum memory for light.

Quantum memory has applications for quantum communication and quantum computation. These topics may some day revolutionize todays computer abilities and network structure. The ability to understand and cross the borderline between quantum and classical protocols may also lead to better classical communication performance.

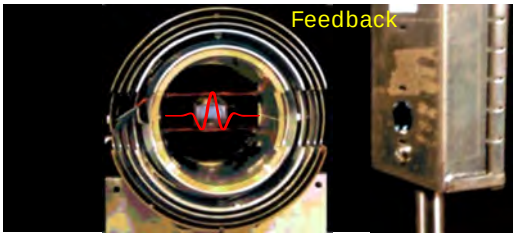

Figure 5.9: Light detected

The density of recorded states is 33 per cent higher than the best classical recording of light onto atoms, with a quantum memory lifetime of up to 4 milliseconds.

5.4 Reading the quantum state

A read-out x-polarized pulse is sent through the samples with a delay of $0.7ms$ to $10ms$ after the feedback is applied. Atomic memory generates a y-polarized pulse, which is analyzed as follows. As both $\hat{X}_A^{mem}$ and $\hat{P}_A^{mem}$ cannot be measured at the same time, we carry out two series of measurements for each input state.

Each series consists of 10^4 quantum storage sequences. To verify the

$$\hat{X}_A^{mem} \rightarrow -\hat{P}_A^{mem} \tag{5.5}$$

step of the storage, we measure the component

$$\hat{X}_A^{readout} = \hat{X}_A^{readin} - k\hat{P}_A^{mem} \tag{5.6}$$

of the readout pulse.

An example of such a measurement carried out after $0.7ms$ of storage is presented as a histogram of $\frac{1}{k}\hat{X}_A^{readout}$ (right histogram). For this series $\left\langle \hat{P}_L^{in} \right\rangle = -4$ and $\left\langle \hat{X}_L^{in} \right\rangle = 0$, corresponding to $\langle \hat{n} \rangle = 8$ photons in the pulse. From this measurement, we find the mean

$$\left\langle \hat{P}_L^{mem} \right\rangle = \frac{1}{k} \left\langle \hat{X}_A^{readout} \right\rangle \tag{5.7}$$

We note that only the knowledge of k and the shot noise level of light is necessary for the determination of the mean values and variances of the atomic canonical variables from the experimental data.

Next, we run another series of storage with the same input state for the verification of the step $\hat{P}_L^{in} \rightarrow \hat{X}_A^{mem}$. $\hat{X}_A^{mem}$ operator does not couple to the readout pulse in our geometry; therefore, we first apply a $\frac{\pi}{2}$ pulse to atoms, converting $\hat{X}_A^{mem} \rightarrow \hat{P}_L^{\pi}$ with the verifying pulse. We then find $\left\langle \hat{X}_A^{mem} \right\rangle$ and $\sigma_x^2 = \left\langle \left(\delta \hat{X}_A^{mem}\right)^2 \right\rangle$ of the memory state (left histogram).

The above sequence is repeated for different input states. From $\left\langle \hat{P}_A^{mem} \right\rangle / \left\langle \hat{X}_L^{in} \right\rangle$ and $\left\langle \hat{X}_A^{mem} \right\rangle / \left\langle \hat{P}_L^{in} \right\rangle$, the mapping gains for the two quadratures are determined. For the experimental data these gains are 0.80 and 0.84 respectively, which is close to the optimal gain for the chosen input set of states. This step would complete the proof of the classical memory performance, because we have shown that the y-polarized pulse recovered from the memory has the same mean amplitude and mean phase as the input pulse (up to a chosen constant factor).

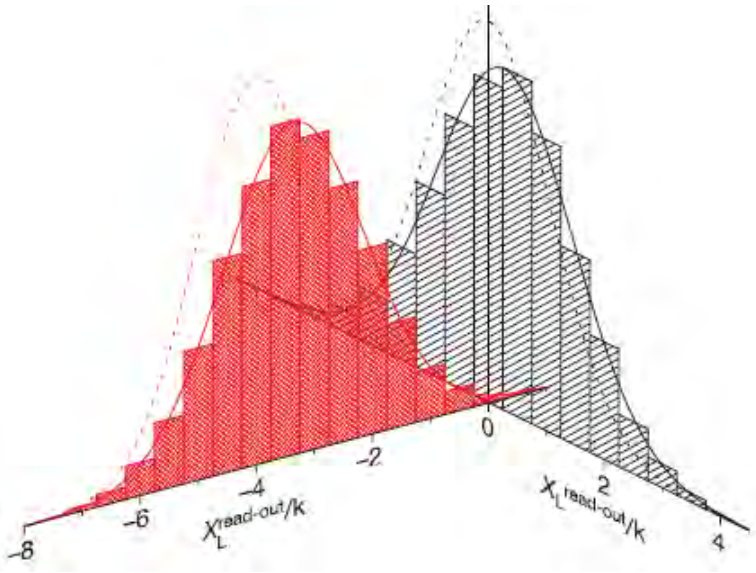

Figure 5.10: Read out experimental data

Chapter 6

Application: Quantum Cryptography

6.1 Vernam cryptographic system

Is is a very simple cryptographic system from the conceptual point of view. But if the required conditions are satisfied, it can produce a 'perfect secret', this means that there is no method to decipher the message.

It uses displacing of encoding. Each character to cipher, is displaced in the encoding table. For example, if using American Standard Code for Information Interchange (ASCII), the displacement could be both to the left or to the right, according to a preset sequence.

Ciphering:

The clear text message $x_1, \ldots, x_n$ produces the ciphered message $y_1, \ldots, y_n$ by doing: $y_j = x_j + k[j]$

Deciphering:

The ciphered message $y_1, \ldots, y_n$ is deciphered into the message $x_1, \ldots, x_n$ by doing: $x_j = y_j + k[j]$

The present sequence K must be the same size or longer than the message to cipher. This sequence is named as 'common secret', and is a private information and secret between the sender and the receiver that cannot be shared by public means.

Moreover, for being really safe, this sequence K must be a random string and cannot be use more than once. Statistic analysis could be used to find it out.

Example:

	Text	ASCII encoding				
Common secret key:	HELLO	48h	45h	4Ch	4Ch	4Fh
Data to sent:	there	74h	68h	65h	72h	65h
Ciphered data:	<- 1 >4	3Ch	2Dh	31h	3Eh	34h

The practical problem of this system is to obtain two copies of the sequence K (common secret key), without being spied. This problem is quite similar to the initial problem of exchanging the information. The only difference is that K can be any message of certain length, meanwhile the initial message has to be exactly that one.

6.2 Protocol BB84

6.2.1 Introduction

Figure 6.1: Charles Bennett

Figure 6.2: Gilles Brassard

This protocol was proposed by Charles Bennet and Pilles Brassard in 1984, and this is the origin of its name. The scenario is like this: Alice, wants to share a key with Bob, in a secure way. For this purpose they are going to use two communication channels, both of them insecure, one for quantum information, and another for classic information. The quantum channel could be an optical fiber, or any other channel able to transmit polarized states. And the classic channel could be Internet, a phone or any other mean.

Alice will send photos to Bob one by one, with a vertical, horizontal or diagonal polarization in a random way. Bob will measure them to find out what polarization they have. They can use two different set of basis.

Using one basis they can distinguish between photos with a vertical or horizontal direction. And by using the other base it is possible to know which one of the diagonals has been used. However, if a wrong basis is selected, the photon will change its state during the measurement process into one of the fundamental states of the new basis, but Bob will not notice about this problem.

This reorientation of states using a wrong basis for the measurement will produce an error rate of 25%.

The explanation of this phenomenon has to do with the uncertainly principle, the probability distribution of the measurement process and the basis used for encoding the bits. When using a base for encoding, if using the same one for decoding there is a success rate of 100%. But if measuring into another base, this depends in how this second base is decomposed into the linear independent states of the first one.

The next sections will explain the concepts and data encoding used in the protocol. Also a *master* and *slave* station are defined, and its relation with an *spy* station.

6.2.2 Basis definition

The most common set of basis is the one expressed by the two states

$$\textbf{Basis A} \quad |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

And when there is need for a set of orthonormal basis to the previous one, the common states are

Basis B $|+\rangle = |\uparrow\rangle + |\downarrow\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $|-\rangle = |\uparrow\rangle - |\downarrow\rangle = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

6.2.3 Define coding criteria

As a general rule, we can use the first state of the basis to encode `true`, and the second one to encode `false`, resulting:

$$false_A = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad true_A = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
$$false_B = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad true_B = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

In the simulator this mapping this internally done for each data type. The user does not need to modify this parameter, if the simulator assures the internal consistency.

However exploring different ways of encoding the information and the implications of noise and other parameters is a quite interesting research line.

6.2.4 Check measurement probabilities of every codification state

	measure *false* using basis *A*	measure *true* using basis *A*	measure *false* using basis *B*	obtener *true* using basis *B*
Encode *true* using basis *A*	100%	0%	50%	50%
Encode *false* using basis *A*	0%	100%	50%	50%
Encode *true* using basis *B*	50%	50%	100%	0%
Encode *false* using basis *B*	50%	50%	0%	100%

The base *A* are two orthogonal values, and the base *B* are the diagonal of the states defined in *A*. In other words, the elements of the base *B* are in half way between the states if base *A*. For the states each basis, there is a probability of 50% of 'collapsing' in each one of the other basis states, when using the other basis for measurement.

The simulator performs this calculation in a generic way, it finds out the probabilistic distribution of the state being measured according to the basis used. Later it selects one of possible the final states of the measurement, following the probabilistic distribution already calculated.

6.2.5 Generation phase: Step 1, qubits transmission

The *master* station chooses a basis randomly, and a value `true` or `false` also randomly. Those two classical values are stored.

It sends the quantum state of encoding the `true` or `false` value in the selected basis.

The *slave* station measures the quantum states using randomly selected basis. It stores the used basis and measured value.

This is the only phase of the protocol that uses a quantum communication channel. For all the other steps and phases is enough with a classical channel.

The *master* station must send $(4 + \delta) N$ qubits. Being N the size of the key to generate, and δ a parameter related to the specific quantum channel. This parameter is related with the noise level or fidelity of the channel.

6.2.6 Generation phase: Step 2, basis transmission

The *master* station sends the basis used to encoding the data, and the *slave* store them.

It does not sends under any reason the data that has been used to generate the qubits. Those are secret values that are waiting to be be transmitted.

The simulator transmits a bit for each qubit. The total number was calculated in the previous step.

If the bit is set to `false` means that the base A was used, and if is *true* the base B was used.

6.2.7 Generation phase: Step 3, transmission of coincidences

The *slave* station sends the positions that performed a correct measurement.

The *slave* station does not sends what measured in the first step of this phase. This is the secret to transmit, and must be kept secret.

The simulator, sends as many bits as many qubits were calculated in the first step. If the bit is `false`, means that there was no coincidence, and if the bit is *true* means that the `master` and *slave* station measured in the same base.

6.2.8 Integrity verification of transmitted key size

This phase checks the size of the intermediary key just generated. It should have a minimum size of $2N$, being N the size of the data to cypher. This is the reason of the mysterious 4 that appeared in the first phase of the algorithm. How the success rate of measuring in an unknown base is $\frac{1}{2}$, and we ignore the invalid ones, at the end of the first phase only half of the sent values remains. So, if we want to receive $2N$ values, we should send at least $4N$.

However it also exists the possibility, but small, of not granting the minimum size, (this is configurable with the δ parameter), in this case the BB84 key distribution is aborted. Several attempts of establishing a valid key will be done before concluding that the channel is broken or there is a spy (will be explained later).

The simulator, model this raising an exception of informing of the wrong minimum key size. This exception should be catch.

We have an intermediary key of size $Z \geq 2N$. We transmit the exceeding $Z - N$ (at lest they will be N) for ensuring the integrity of the transmission of the first step of the previous phase (transmission of quantum states).

6.2.9 Checking phase: Step 1, transmission of positions to check

The *master* stations transmit the positions to check, and the *slave* station stores them.

The simulator performs this operation transmitting as many bits as there are in the intermediary key. If the bit is stored in the position P of the intermediary key that is going to be checked by public means, the bit P of the transmission status will be `true`. If the bit will not be checked through a public channel it is set to `flase`.

6.2.10 Checking phase: Step 2, master station transmit the values

In this step the *master* station and the *slave* station already know what bits are going to be compared. The *master* one sends the values that wanted to transmit, and the *slave* store them.

The simulator only transmit the values associated to $Z - N$ bits selected in the previous step.

6.2.11 Checking phase: Step 3, slave station transmit the values

In this step the *slave* station will send what has received and measured in the correct basis during the generation step. In this way the *master* station will be able to calculate the noise level of the channel.

6.2.12 Decision phase

Now the *master* and *slave* station have enough information to determine the noise level that happened during the data transmission. Both have what was intended to be sent, and what was received in reality. If the noise level is above certain level, the transmission is canceled.

An *spy* is detected by means of a very high error rate. The transmission errors happens when the *master* station sends, as example, $|\uparrow\rangle$. But meanwhile in the channel (because of nose of an spy) the state $|\downarrow\rangle$ changes into the state $|\uparrow\rangle$. We assume that we measured in a correct basis, because the elements that were measured in the wrong bases were already discarded in the previous phases of the protocol. Alice has sent `false` in the basis A but Bob has received `true` in the basis A.

An *spy* measuring $|\uparrow\rangle$ in the base A would not change the state being sent; but 50% of the times, will change it if measuring using the base B. From this 50% of wrong basis guessing of the *spy*, a 50% would not be detected (due to wrong guessing in the *slave* station). In total, the *spy* produces an error rate around the 25% of transmitted qubits.

In absence of noise, the only chance for the *spy* of not being detected is to guess all the basis of the quantum transmission, that this is equivalent to guessing the data to be ciphered without accessing the channel.

The data transmission is safe because no previous secret is needed. Only having an agreement on the basis and the bit encoding into quantum states is needed, but is agreement is not a secret, can be considered as public information.

The only security issues would come because of the implementations in the *master* and *slave* stations, mainly in the random numbers generation.

They must be true entropy sources, physical devices connected to the station. They cannot be algorithms for generation pseudorandom numbers. Special techniques for predicting them could be developed and applied.

6.3 Protocol B92

The protocol B92 is a modification by Bennett of BB84 using only two not orthogonal polarization states. Consequently, it is a simplification of BB84.

6.4 Protocol E91

This protocol was developed by Artur Ekert in 1991, and uses entangled photons. Those photos can be prepared by Alice, Bob or a third party, and they are distributed in such a way that Alice and Bob have a

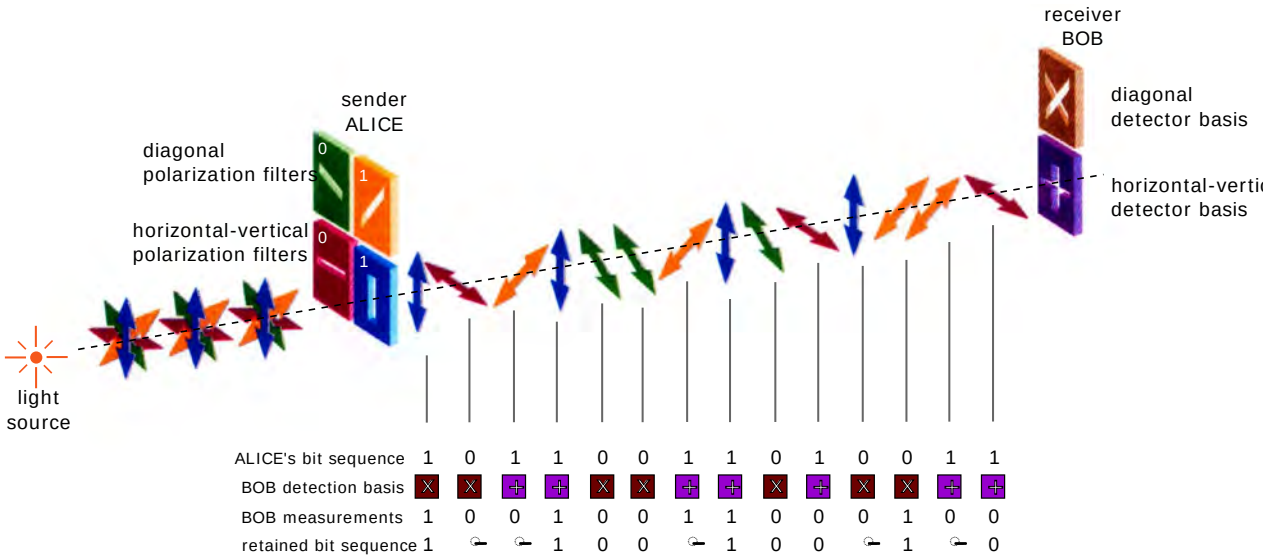

Figure 6.3: Graphical diagram of BB84

Figure 6.4: Artur Ekert

photon of each pair. The photos are in the state:

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{6.1}$$

The operation is based on the entanglement properties. Alice an Bob have each one a photon antipararel to the other one. If they measure their photons in a vertical or horizontal orientation they will measure opposite results, the same will happen with diagonal basis. It is not possible to predict if Alice will measure, for example, in a vertical or horizontal orientation. Also, any attempt of Eve for eavesdropping the data transmission, will affect the correlation, so Alice and Bob could detect it.

6.5 Commercial developments

The first implementation of QKD was done in 1989 by IBM. The transmitted photos through the air to a distance of 32 cm and implemented the BB84 protocol. From this milestone many improvements in distance has taken place...

In 1994 a group of British Telecom sucess in using an optical fiber of 30 km for QKD. En 1995, researches from the university of Geneve implemented B92 with an optical fiber of 23 km length, between Geneve and

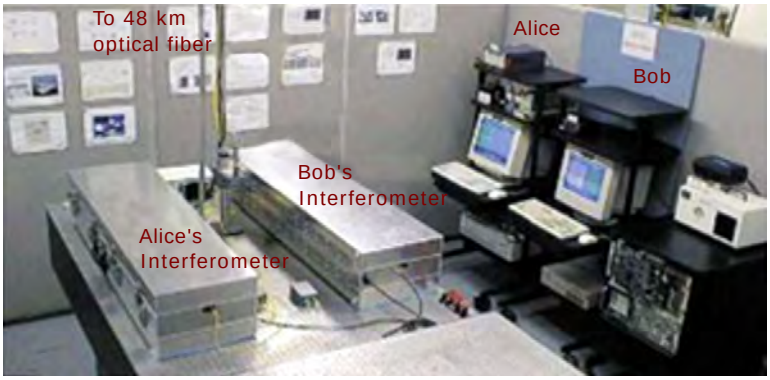

Figure 6.5: First experiment of QKD

Figure 6.6: Geneve

Nyon through the Leman lake. Later, in 1999, the Los Alamos laboratory extended this limit up to 48 km. Beginning of 2002, a Swiss company establish a distance of 60 km. In November of 2002 Mitsubishi extends this up to 87 km. And in June of 2003, Toshiba Research Europe sets the limit up to 100 km.

Since some time ago there are several commercial implementations for QKD and some other services.

An example of is is *IdQuantique*, in Switzerland. They produce a device that interfaces with a PC through an USB port. Those devices are interconnected using an optical fiber, up to a distance of 100 km, and they negotiate the cryptographic key generation autonomously. As a detail, it is possible to choose for the key generation Advanced Encryption Standard (AES) or 'single usage keys'. It provides AES support for legacy systems what would like to use the true entropy sources of the system.

IdQuantique also sells related products, as PCI cards for true random numbers based on photos polarization, and individual photos detectors.

Another company *MagicQ*, produces two branches of products, one devoted to research and education, and another for professional use. The peculiarity of the professional products is a connectivity to several

Figure 6.7: IdQuantique system photo

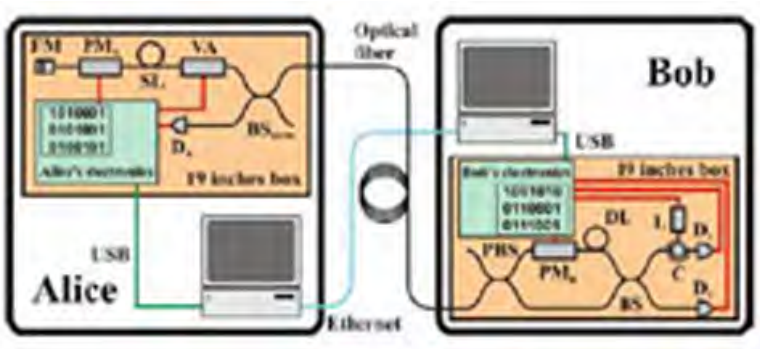

Figure 6.8: IdQuantique conceptual design

terminals, that can be considered as a LAN for quantum computing. It operates up to 120 km, and interfere with other security approaches such as Public Key Infrastructure (PKI).

Figure 6.9: MagicQ QKD research oriented devices

Figure 6.10: MagicQ QKD commercial oriented devices

A common fact for all the companies that commercialize this technology, is that they assure that quantum cryptography is absolutely secure. As already stated, the protocols can provide a 'perfect secret', and the companies claim that their implementation is bug free. They are already used by big companies and banks, but currently it is out of the question for domestic use due to the optical fiber physical installations.

Chapter 7

Complexity of quantum computational model

7.1 QTM

A QTM is made of:

- A finite set of states S, where a differentiated state $s_0 \in S$, is designed 'initial state'.
  We can map each element of S as a vector of the Hilbert's space, H_S, of dimension $|S|$, where we take an orthonormal basis. In this way, we map the states of S into quantum states (vectors from H_S). It is enough by considering the new biyective application $m_S : S \to H_S$.
  We use the notation $|s\rangle := m_S(s)$
- An alphabet of symbols Λ (input alphabet)
- An alphabet of symbols Γ (output alphabet)
- A differentiated symbol, Δ, (blank symbol)
- A tape or a memory device that is just a set of labeled cells:
  $\ldots; cell[-1]; cell[0]; cell[1]; \ldots$
  Each entry may contain a symbol from $T := \Lambda \cup \Gamma \cup \{\Delta\}$
  At the start all the cells contain the white symbol. Once that we introduce the input string, they will be stored in the correspondent cells, but all the others will continue having the white symbol, Δ.
  The contents of each cell is identified by the quantum states of the Hilbert's space H_T of dimension $|\Lambda| + |\Gamma| + 1$.
  The Hilbert's space described by the tape is consequently $H_M = \otimes_{\infty}^{\infty} H_T$. The biyection $m_T : T \to H_T$ maps each element form the tape cells into the elements of the Hilbert's space H_T.
  We use the notation $|t\rangle := m_T(t)$
- A reading head that can read the contents of each cell, write the symbol T, and move to left and right. All those actions can happen at the same time. The position of the reading head can be identified with the orthonormal states of a Hilbert's space H_{TH} of finite dimension. The biyective application $m_{TH} : Z \to H_{TH}$ maps the position of the reading head (i.e., an integer that identifies the cell) with elements form the Hilbert's space.
  We use the notation $|j\rangle := m_{TH}(j)$
- A finite set of transitions for the states and symbols of $\Lambda \cup \Gamma \cup \{\Delta\}$. A transition is a 6-tuple ordered $(a; b; c; d; e; f_{a;b;c;d;e})$ con $a \in S, b \in \Lambda \cup \Gamma \cup \{\Delta\}, c \in S, d \in \Gamma \cup \{\Delta\}, e \in \{+1, -1\}$ and $f_{a,b,c,d} \in C$.
  Being a the current state, b the symbol readout by the reading head, c the next state, d the symbol

that the reading head will write in the next cell and $e = +1$ (or $e = -1$) move the reading head towards right (or left). If there are no transitions for $(a; b; c; d; f_{a;b;c;d;e})$ we define $f_{a;b;c;d;e} = 0$.
It is mandatory that:
$\sum lim_{c,d,e} f_{s,t,c,d,e} \overline{f_{s',t',c,d,e}} = \delta_{s,s'}\delta_{t,t'}$
$\sum lim_{c,d,e} f_{s,t,c,d,e} \overline{f_{s,t,c',d',e'}} = \delta_{c,c'}\delta_{d,d'}\delta_{e,e'}$
being δ the Kronecker's delta: $\delta_{i,j} = \left\{ \begin{array}{l} 1, \text{if} i = j \\ 0, \text{if} i \neq j \end{array} \right\}$

7.2 Some quantum complexity classes

We can define the complexity class **Bounded-Error Probabilistic Polynomial-Time (BPP)** as follows:

A language L is included in BPP if there exists a probabilistic matrix T as the one described before and a polynomial time t such as:

- for $x \in L$ there is a $T^t(c_I, c_A) \geq \frac{2}{3}$
- for $x \in L$ there is a $T^t(c_I, c_A) \leq \frac{1}{3}$

We could replace $\frac{2}{3}$ and $\frac{1}{3}$ by $1 - 2^{-p(x)}$ and $2^{-p(x)}$ respectively; and in this way we can control the error with the polynomial $p(x)$.

Generalizing this idea in the quantum model...

We can define the class **Bounded-Error Quantum Polynomial-Time (BQP)** as follows:

A language L is in the class BQP if there is a quantum matrix T and a polynomial time T such as:

- for $x \in L$ there is a $\left(T^t(c_I, c_A)\right)^2 \geq \frac{2}{3}$
- for $x \in L$ there is a $\left(T^t(c_I, c_A)\right)^2 \leq \frac{1}{3}$

Where a quantum matrix is an unitary matrix of finite dimension, that is exponential to the size of the input.

It can be proofed that this idea is the class BQP.

7.3 Relations among quantum complexities

For an exhaustive categorization of complexity classes, quantum an classic, the best source of information is http://www.complexityzoo.com.

7.4 Conclusions

Theoretical researches show that the processing power of the quantum model can be greater than the one described by the Turing'machine. But it is important to make clear that there is no formal proof of this assumption. It is not known if the quantum model really has a greater computation power than than the classic one.

The quantum computing model has been presented as an extension of the already known probabilistic model. The complexity class BQP is formulated as an extension of BPP. This model formalization includes all the computing power of the Quantum Computing, but still looks like a simple extension of the probabilistic model, and quite distant from the Quantum Mechanic postulates.

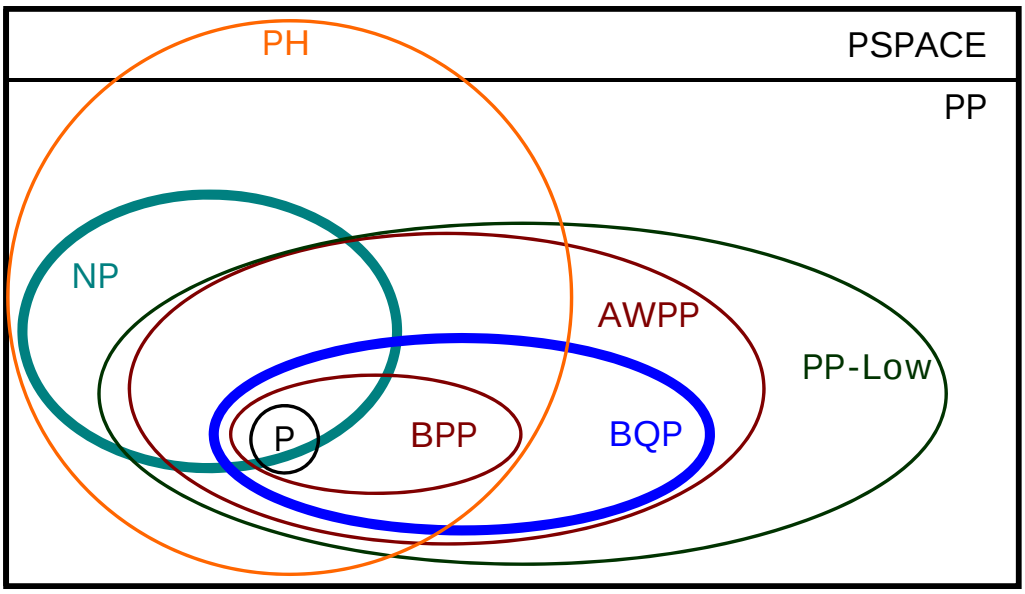

Figure 7.1: Quantum complexities classes

It looks strange that several concepts are not properly addressed in the formalization, such as the need of complex numbers, what happens with reversible computations?, and about the measurement process?,...All those points are discussed in [Buh00]. In this paper it is described that the concept of reversibility is a consequence of the quantum formalism, and the measurement process is included as the 'acceptation' in the class BQP.

Consequently, by considering the quantum model as an extension of the probabilistic model, the computing power will be a factor of the parallelism degree obtained by the superposition of quantum states. This also happens in the probabilistic model, but in the quantum one, we can 'cancel' the non-relevant computations. This means, that the difference among both models lays in the concept of 'interference': two states can compensate mutually if their amplitudes have the same value and opposite sign.

The concept of entangled states, far from could be expected in a first approach, is an slow down for the computing power. In the same way, that the matrices must be unitary is also a drawback.

So, for summarizing, the main points are:

- Once built, how powerful will be quantum computers?, What is the computational complexity of those devices?, Does BQP contains to NP?,...
- Therefore, the need of understanding the complexity class BQP is of critical to evaluate the processing power of the quantum computing model.
- There is no certainly that the quantum model has a greater processing power the classic one. But everything to indicate that it is greater.

(This page is intentionally left blank)

Part III

Requirements and Solutions: QIT

(This page is intentionally left blank)

Chapter 8

Objectives of this software

8.1 General overview

The idea was to develop a system able to simulate (not even efficiently) the processes of quantum computation an cryptography. To archive this aim it is necessary to simulate 'physical processes'. I didn't just to show the results of the theoretical model, but to perform all the intermediary steps that the model tries to describe.

The simulation of quantum states has to deal with several types of formalization:

- Using differential equations governing the temporal evolution of the systems. This involves using a large number of integrals and differential equations such as:

$\frac{d}{dt}\left(m\frac{dx}{dt}\right) = F$ Newton's Law

$-\vec{\nabla}\cdot\left(k\vec{\nabla}\vec{u}\right) = \vec{Q}$ Poisson's Equation

$\vec{\nabla}\cdot\vec{\nabla}\vec{E} = \varepsilon_0\mu_0\frac{\partial^2\vec{E}}{\partial t^2}$ Maxwell's Equation

- Representation based on states (vectors), and define the temporal evolution of the system with operators (matrices) that include the differential equations. This is the notation used in the introduction chapters.
- Using logic gates. Each 'wire' that connects the gates, represents an state, and the logic gate represents the operators performing the evolution of the states.

All these approaches have the same computational power, but the difference is how easy is to use them in the different contexts. From a theoretical point of view they are complete an correct, and any representation can be transformed into the others.

The formalization based on differential equations is very difficult to handle overtime. When new states are considered, all the equations must be rewritten to consider the new variables. This is computer intensive, and does not produce any advantage over the other approaches. Also if there is no symbolic calculus, precision losses will happen over time.

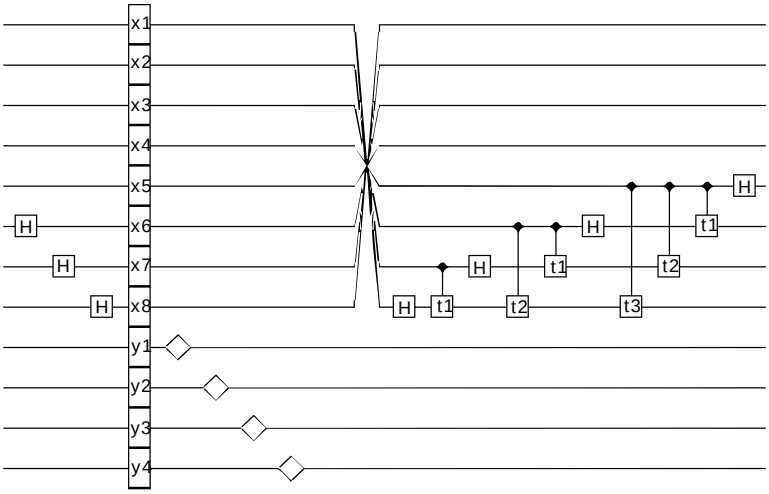

Figure 8.1: Quantum logical gates circuit

The formalization based on matrices, has the problem of the states growth. From a practical point of view, when two systems are put together, they behave now as a single system, and there are more states involved, from a global point of view. This implies to modify the internal structure of the states to show that at some moment they become part of a bigger system. If they were isolated, memory could be reused, but as a single system, this is no longer possible.

The simulation of logical gates, does not have this type of problems and according to other simulators seems to work quite well with already developed algorithms. It is even able to handle also entangled states. However this approach, needs a mathematical back-end for the input and output state, and the choices on this representation will impose constrains in the simulator design.

I decided to use a simulation based on vectors and matrices. Any quantum state can be formalized as a vector, and any evolution of a system as a matrix multiplication. For presentation purposes, a component dealing with logic gates and translating into states and matrices can be developed. This is also the most common formalism in the theoretical researches. Also matrix based calculus is very common in scientific computing; and processors and algorithms are more optimized for handling data blocks rather than floating point operations.

8.2 Functional requirements to implement

8.2.1 Randomness

This objective is included from the very early phases of the project. All programming languages provide some 'random number generator', but but observing patterns on them is not a surprise.

8.2.2 Matricial operations with complex numbers

Usual programming languages, include support for matrices, but only for the basic types already implemented, the problem are to provide matrices for complex numbers. The data structures and operations must be implemented.

8.2.3 Quantum states simulation: pure, mixed and entangled

They are are implemented using vectors or matrices, but pure, mixed and entangled states must be treated as independent concepts. This allows a better abstraction when differentiating the mathematical formalization from the physics phenomenology.

8.2.4 Quantum states growing

By doing the tensor product among two states, those to initial states modify their internal structure, and they are linked to a third state not accessible directly.

8.2.5 Physical states basis

Simulating the states is not enough, for operating with them it is necessary to define the basis used. The basis are also quantum states, but with an identifier associated, for being able to refer to them.

8.2.6 Operators

They are implemented as matrices, but as already explained with the states, this is a different concept. For that reason they are encapsulated as a different interface `IOperator`.

8.2.7 Noise

This is a complicated issue, there are many formalizations for the noise. But all of them ignore some issues to be considered good for a realistic simulation. They are based in the probability of transition from one state to another. The noise can be also implemented in the transmission over a channel; it would the probability of sending one state, but receiving another.

8.2.8 Conversion of physical states into information

We need to define, even a minimalistic, a data type theory for encoding and decoding logic types, integers, arrays and strings into and from quantum states.

8.2.9 Logical gates

For the simulation of BB84 this component is not needed, but its concepts are roughly defined for future extensions of the framework.

8.2.10 Communication channels: senders and receivers

For the distribution of qubits (quantum states), it is needed a simulation of an stack of Open Systems Interconnection (OSI) levels. Independently of what kind of logic to implement a physical simulation of the channels is mandatory.

8.2.11 Protocol logic

The idea is to establish some kind of logic or state machine, that can be reused to simulate many protocols or automates.

8.2.12 Classic cryptography

The Vernam ciphering is implemented to do the data ciphering after the quantum key distribution using BB84.

8.2.13 BB84 key distribution

To implement a BB84 simulation flexible enough to show all the steps during the key distribution.

8.2.14 User interface

To provide an User Interface to expose all the functionally of the framework. User friendliness is only relevant on the BB84 simulation.

8.3 Nonfunctional requirements

- It should be able to work in a single PC, but also work in a distributed environment, taking advantage of 'grid' technologies.
- An efficient usage of the memory, this is more important than computing speed. The incrementing size of matrices during the simulations is a critical point.
- Isolate the theoretical concepts, from the mathematical foundations and the implementation techniques. This allows to find over time better implementation to the physical concepts, allows a better distributed system, and even more important, has fundamental consequences for usage of this simulator for theoretical researches when the operators and states are not linked to the formalization.

Chapter 9

Software architecture

In order to achieve such objectives I decided to implement framework and an example application using it.

The framework should manage all the simulations. This environment should be multiplatform and distributed. A first approach of a distributed system would use repositories for the states, the bases and all the objects. For being distributed multiplatform all the design should be done using interfaces and components. In addition, there can be multiple instances of each component, in this way every machine can have a local copy of the repository or components, this would improve the performance, in case than the master of the information is a remote node. Each one of these components can be implemented with a different language and different structures of data.

For these purposes it was necessary an intensive usage of design patterns. All the logic for processing data, must be independent of how to store it.

Direct references to classes could not be used, only interfaces. For that reason all the objects and data that are generated come from generators initialized statically or classes factories.

As an example, lets take the case of logic doors. Each of these doors needs an associated operator and the input states (all of them represented by matrices). When it is necessary to perform an operation, a matrix for the operator is instantiated. This matrix can be a fresh object or a object that is already been used previously (the matrix of the operator is always the same).

It well could be a shared resource on the network, something independently globally in each machine, something that cannot be reused among executions. Later, it gains the control of the unique elements that are the states . Therefore the processing limit is the limit of states non dependent among them. This limit is the limit of the data flow.

Another example are the random number generators. They are used very frequently and it is not necessary that they are unique elements. All the objects/components that do not have an internal state, like factories, operators, generators,...can be replicated in each machine without majors problems. Although he will be advisable that is registered in global repository with the purpose of traceability/debugging.

The graphical application only has the necessary logic for the screen display of data and user interaction. The framework, owns all the logic of the simulation and must provide the sufficient methods so that the application can perform its tasks. But also the framework must hide internal methods, so that the

application does not generate incoherent situations within the framework.

The internal structure of the framework is quite more complex. It is structure as an stack of N components, having dependencies among them. That stack of components has an structure of layers as represented in Figure 9.1. Each stack could located on different machines or replicated into each one.

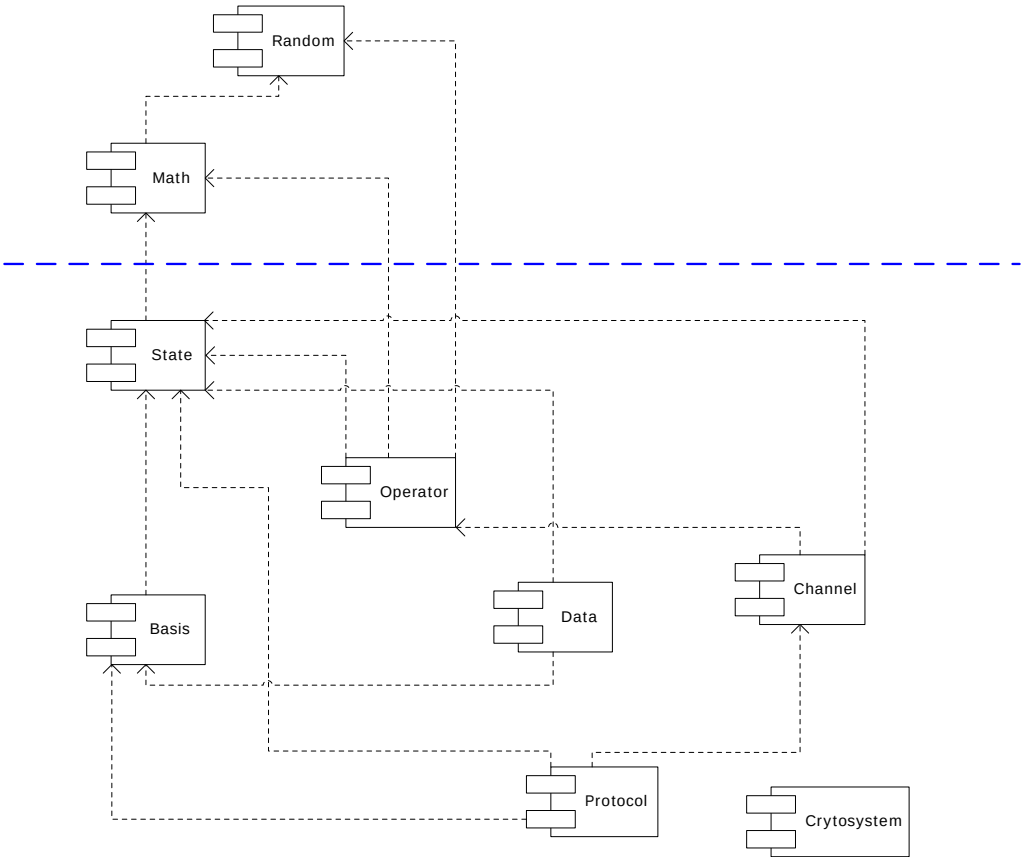

Figure 9.1: Framework architecture

As it can be observed, for dealing with states (measurements) it is not necessary to access the formalization done with matrices and vectors. And there is no direct reference to the randomness module.

Similarly, when working with the BB84 protocol, it is not necessary to specify the states directly, only the communication channels, with their respective basis.

Chapter 10

Choice of software tools

10.1 Alternatives

For the development of the project I compared numerous alternatives. From using a unique language of high efficiency like C++ or freepascal and using portable libraries, to languages that did not depend absolutely on the execution machine as it can be Java.

As also it was important the possibility to use the framework in a system way through a network, an option for this were 'remote procedures' or Common Object Request Broker Architecture (CORBA).

But without losing of view the main target of the operations also it was also necessary intensive calculation capacities (floating point) and an efficient management of the memory. Even more important that the mathematical operations is the memory management, since numerous objects will be created. The quantum data storage simulations, due to the entanglement, has an exponential complexity in memory.

	C++	Java	C#
Portability	medium	high	high
2D/3D features	media	low	low
Other languages integration	high	low	high
Complex numbers	medium	low	low
Memory management	high	medium	medium
Execution speed	high	low	medium
Remote procedures	medium	high	high
Programming environment	medium	medium	high

Table 10.1: Comparative overview

10.2 Further implications

10.2.1 Portability

I also evaluated mixing C++ with Java, through Java Native Interface, but this discarded due to maintenance problems. In addition to this, this was not resolving the portability portability problems, since the integration code is on the C++ side, and is platform dependent.

The option to use freepascal was very interesting. It allowed to use a syntax more comfortable than C++, it supports almost all the platforms available, but does not have any support for 2D/3D.

C# although originally from windows can be executed in UNIX machines with the Mono project. It is not this related to any specific software architecture or hardware.

10.2.2 2D/3D graphics

C++ depends completely for this purpose in external libraries, this complicates the portability, because specific portable libraries such as vxWindows for 2D are needed.

Java has complete support for 2D, but their three-dimensional capacities are limited to x86. In spite of everything they exist some components that allow to generate graphs 3D without using native code, but are quiet slow.

C# has an excellent support for 2D, but it is very limited for 3D.

10.2.3 Integration with other languages

C++ is the star with difference, and in fact all the other languages insist on having possibilities of communication with C++. It is used as a 'common bridge'.

As already commented the ones of Java are not well suited.

C# has all the integration capabilities of the .NET platform. This is the great improvement with respect to Java. It is quite easy to code something in C# to to use classes programmed in native C++, or even native FORTRAN.

10.2.4 Mathematical features (complex arithmetic)

In this point the star is FORTRAN with difference, it is even the unique language with native support for complex numbers. In addition it allows to use data types of enormous precision (up to 8 bytes).

Examples: blast, lapack, atlas

At a certain distance there is C++, much more structured and modern than FORTRAN code, but less specific. Many mathematical libraries internally use FORTRAN code.

Examples: cmvlib, blitz++, gsl, tnt

Java owns a virtual machine very limited in this aspect and almost all the weight falls on software code. Although there are many utilities and libraries, it is not well suited for intensive mathematical processes.

Examples: jsci, jama, mtj

There are some pure high-level software developments in the .NET platform. They are not much optimized, but at this platform does not define an low-level virtual machine, further optimizations are possible.

Examples: psdotnetmatrix

10.2.5 Memory management

In this aspect C++ is most efficient, since absolute control of the memory is obtained. All the techniques for pointer handling, counting of references,...can be used.

Java and C# have automatic memory collectors. This facilitates the programming enormously; it is not necessary to reserve it nor to release the memory. But this simplicity of use is not a critical aspect, with a good design and methodology in C++, complex memory structures can be created and maintained, while being easy to use in the rest of the program.

10.2.6 Execution speed

C++ is quite efficient, moreover if the compiler is also able to optimize the resulting code.

Java and C# do not generate native code, the compile for an intermediate layer that needs to be interpreted/compiled again.

The approach of Java is to define an architecture virtual hardware. A machine with his instructions and stack levels...

C# defines much more semantically rich an intermediate language, but with equal theoretical capacity to solve problems. But the reality is that being this intermediate language of much more expressive, it allows to define instructions of more higher level. This allows to do some special optimizations when converting this intermediary code into native code.

10.2.7 Remote procedures

C++ directly does not have any type of capacity in this aspect. Everything needs to be done by means of external libraries. So, the only one choice in this language is to use CORBA.

Java supports Remote Method Invocation (RMI) and CORBA, consequently the choice would be CORBA.

C# has .NET remoting, a very transparent mechanism. Nevertheless to maintain the portability and interoperability with other systems/architectures it would be necessary to use CORBA. For this purpose a CORBA broker written in C++ or Java has to be integrated in the .NET platform.

10.2.8 Development environment

For the programming in C++ or C# I would use Microsoft Visual Studio, meanwhile for Java I would use Eclipse.

Eclipse and Visual Studio (VS) both have excellent refactoring capacities, compatible with several languages.

10.3 Conclusions

I decided to use the .NET platform as a first approach for the development.

This decision allowed me to use C# language, in my opinion is a quite convenient one, and a powerful IDE as it is Visual Studio 2010.

In this way certain portability and graphical capabilities are assured certain portability and graphical capacities. But mainly it also offers an uniform approach a uniform to the external libraries, as explained in the previous discussions.

At some moment, the components of the framework that required special features, like the matrices, or communications with remote objects, can be extended using the capacities of interaction with other languages. In this way highly optimized native code can be reused in the .NET platform.

Chapter 11

Framework components design

The framework is structured in a set of 'components'. Each one of them try to solve different functional requisites. By communicating among them, under the coordination of a protocol (also a component), through the User Interface, the cryptographic simulation takes place.

11.1 Random

11.1.1 Classes and interfaces listing

`IRandomConfiguration.cs`, `ExceptionRandomGenerator.cs` `RandomRepository.cs` `IRandomGenerator.cs` `AbstractRandomGenerator.cs`, `ByteRandomGenerator.cs` `EntropyLCG.cs`, `EntropyMCG.cs`, `EntropyLEcuyer.cs`, `EntropyLanguage.cs`, `EntropyHotBits.cs`, `EntropyDotOrg.cs` `FactoryRandom.cs`

11.1.2 Hierarchy diagram

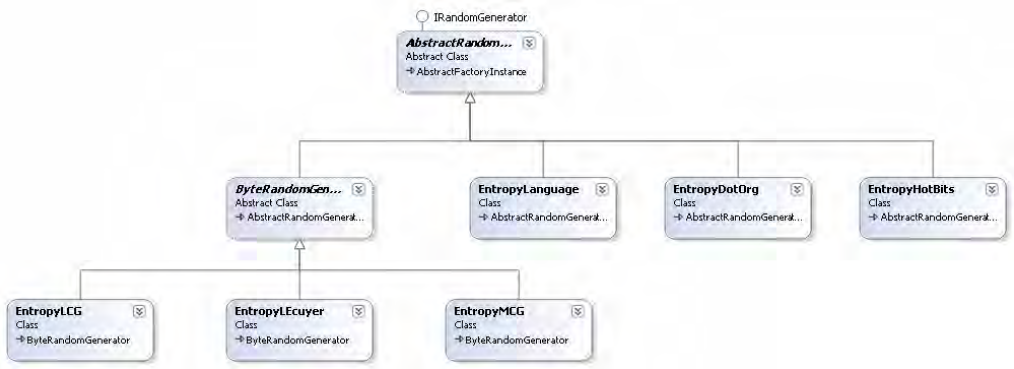

Figure 11.1: Hierarchy diagram

11.1.3 Description

The random numbers components is able to use internally several sources of entropy for generate unprocessed data. Those sources are of two types, pseudorandom generators and true entropy sources.

The **pseudorandom generators** are based on seeds and the evolution of some CPU registries. The result is quite unpredictable, but they suffer many problems, such as long series of identical values. Or not be equally balanced the 0's and 1's on long term.

Each programming language provides a random number generator. But usually the algorithm is not specified. The drawbacks are not known and for a serious usage of random numbers does not have a satisfactory performance.

The **true entropy sources** are physical devices that respond to some environment parameter and encode it in long series of 0 and 1.

Some examples are the electromagnetic noise. The Intel Pentium 3 processors include inside the main chip a generator of this type. Also many manufactures

Other sources are the atmospheric noise. An example of this is http://www.random.org. They process them and they are distributed with a web service.

Another option is to use a radiation detector, as is done in http://www.fourmilab.ch.

11.2 Math

11.2.1 Classes and interfaces listing

`IComplexNumber.cs, AbstractComplexNumber.cs, BasicComplexNumber.cs IComplexNumberOperations.cs, BasicComplexNumberOperations.cs FactoryComplexNumber.cs ExceptionMatrixDimension.cs IComplexMatrix.cs, AbstractComplexMatrix.cs, HashComplexMatrix.cs, SparseComplexMatrix.cs, BasicComplexMatrix.cs IComplexMatrixOperations.cs, BasicComplexMatrixOperations.cs FactoryComplexMatrix.cs FactoryNativeNumber.cs`

11.2.2 Hierarchy diagram

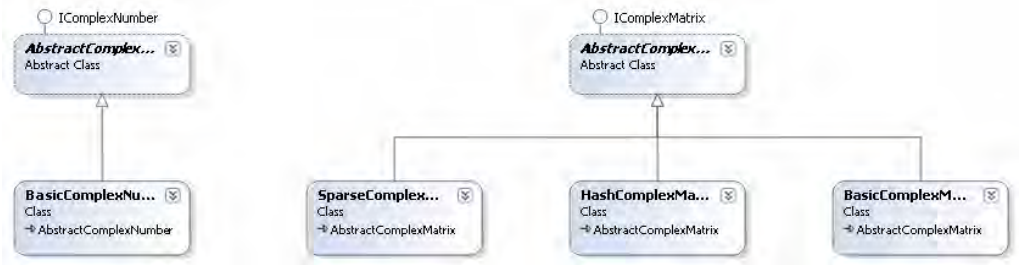

Figure 11.2: Hierarchy diagram

11.2.3 Description

As all the quantum mechanics is based on probability amplitudes, and those amplitudes are complex numbers, it is necessary a mathematical component that supports complex numbers in all kind of

operations, scalar and vectorial.

To archive this aim, it is necessary to implement the complex number definition, matrices and also operators. But as the aim of the framework is to be multiplatform, we need our own implementation 'from the scratch' for the complex number. Most of the computing languages does not offer the complex numbers as native types, one remarkable exception is FORTRAN.

For this reason, one design alternative would have been to use FORTRAN. By using the .NET platform the mathematical part could be implemented in one language and the rest of the framework using another better suited for other purposes. Furthermore, this part could use advanced mathematical packages such as LAPACK or Blitz++. Those packages can also be compiled for specific platforms with native code instead of .NET code.

The decision has been to implement a minimalistic approach using C#, to skip the problems of language interoperability in the prototyping of the framework. But the ideal solution would be to reimplement the interfaces defined in the C# prototype using the advanced libraries.

The design is based on factories of complex numbers and matrices. They return an interface that is used by the operations.

11.2.4 Numbers

The interface `IComplexNumber` defines operations for reading the values and modifying the real and complex part.

11.2.5 Matrices

For the matrices, the are 3 different implementations:

- The most simple one is a traditional matrix with complex numbers as the base type. This implementation has a good access speed for the individual values, but the memory consumption is quite demanding.
- Another option is to use a *hastable*. The key is the position, and the value the complex number. It can be quite useful in spare matrices. If the value is 0 the value is not hashed. Only values different from zero are included in the list. This saves memory, but the access speed can be more irregular.
- An intermediary approach is to use a matrix of pointers. If the value is zero, a 'null' pointer is stored, in other case an `IComplexNumber` is generated an stored.

11.2.6 Operations

The operations with the values, are implemented in top of the types interfaces, but using an independent class. An example of this is `IComplexNumberOperations`.

The operations does not change the internal state of the object they are applied to; they generate a new object.

```
result = FactoryComplexNumber.Operations.Multiplication(A, B);
```

Being `result` a new object different of A and B.

Implementation dependent optimizations are possible by doing a casting of the interfaces and accessing the private properties of the data type.

11.3 State

11.3.1 Classes and interfaces listing

`IStateDebugger.cs, IInformationState.cs, AbstractInformationState.cs ClStateDebugger.cs, ExceptionClassicalState.cs, FactoryClassicalState.cs, IClassicalStateDebugger.cs, ClBasicState.cs, IClassicalState.cs, IClassicalStateCollection.cs, ClStateCollection.cs, ExceptionClassicalDebugger.cs, AbstractClassicalState.cs AbstractQuantumState.cs, ExceptionQuantumState.cs, FactoryQuantumState.cs, IQuantumStateDebugger.cs, IQuantumStateComposed.cs, IQuantumStateIsolated.cs, IQuantumStateSimple.cs, QuStateDebugger.cs, QuStateIsolated.cs, QuStateRepository.cs, QuStateSimple.cs, QuStateComposed.cs, IQuantumState.cs, QuStateCollection.cs, IQuantumStateCollection.cs, ExceptionQuantumDebugger.cs`

11.3.2 Hierarchy diagram

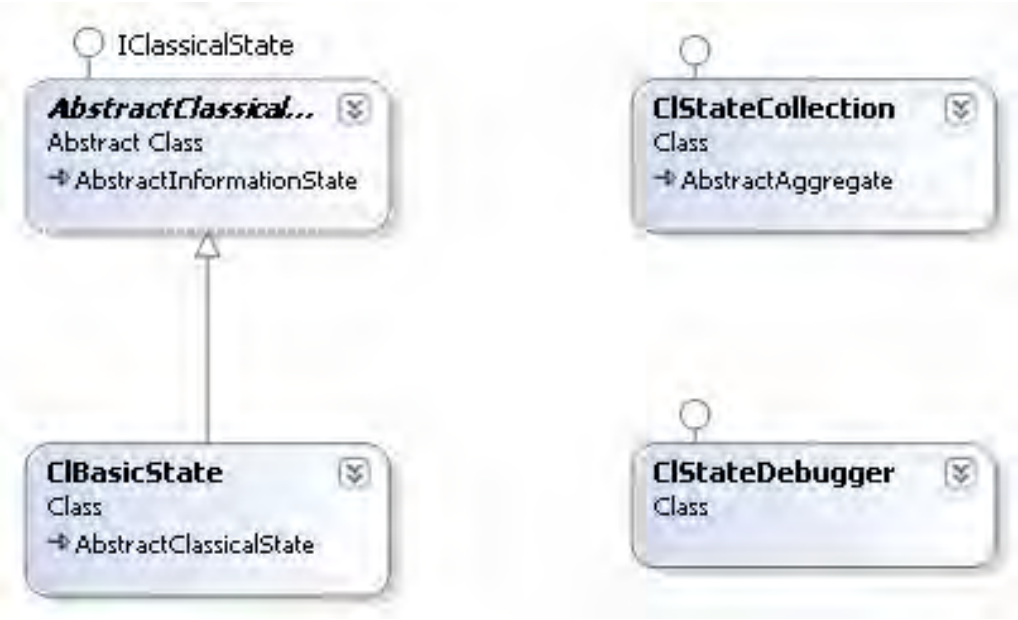

Figure 11.3: Hierarchy diagram

11.3.3 States

There are three implementation for the states:

- `Isolated`: simulates an isolated state, without being composed by substates.
- `Simple`: in principle is an `Isolated` state, but when interacts with another state, by means of a tensor product, its internal structure changes.
- `Composed`: represents an aggregate of several `Simple` states. It is not a tensor state; in a tensor state the bases are shared among the substates. This state is intended to put together in the same state two different states with different basis and without relation between them.

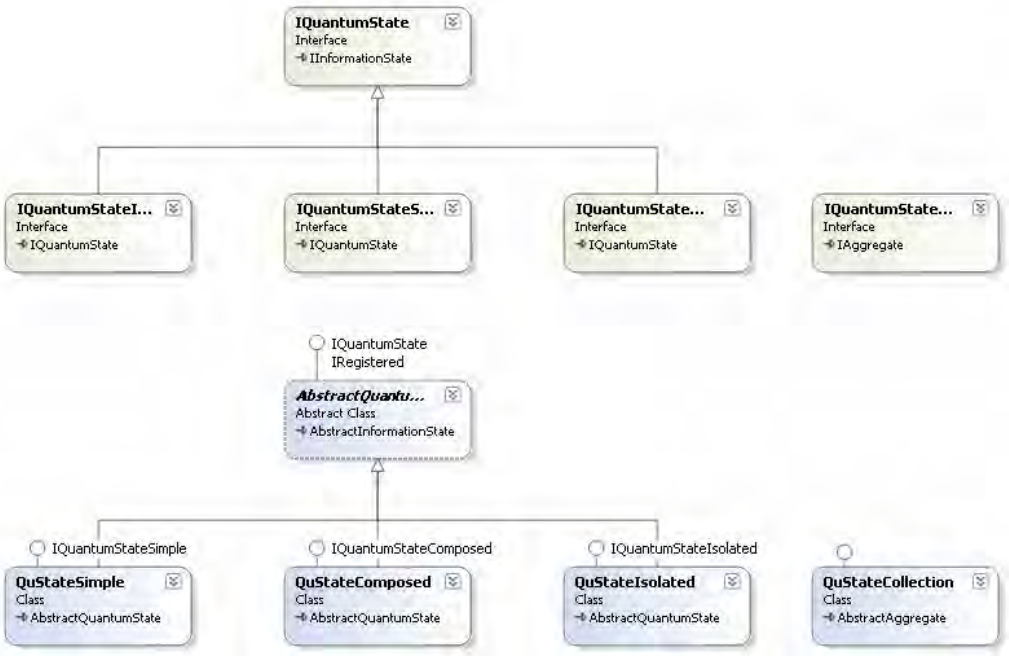

Figure 11.4: Hierarchy diagram

- `Tensor`: it is the result of a tensor product between two states. It should not be necessary to deal with it directly, because the operations into the `Simple` state should change the internal `Tensor` state, if needed.

All this variety of states is abstracted through the interface `IQuantumState`. Programming should be done against this interface and not against specific types of states.

11.3.4 Debugger

There are no direct means to access the coefficients of the mathematical formalization based on vectors. They only way of showing them is to generate a `IDebugger` using the function `generateQuantumDebugger()`.

This debugger allows to inspect and alter the coefficients, but this is not good approach. The protocol or algorithm logic should be based on states and operators, not on coefficients. Internally the framework uses the debugger for the evolution of the states.

11.3.5 Normalization

Internally the coffients are not normalized, and it is not done automatically. Only when they are going to be shown by a *Debugger* are normalized. This also happens in some specific operators.

This is not normalized on purpose to avoid unnecessary calculations, losses of precision, and to allow the usage of symbolic calculus in a future.

11.4 Basis

11.4.1 Classes and interfaces listing

`IBasisDebugger.cs, IBasis.cs, AbstractBasis.cs, ClBasisElement.cs, FactoryClassicalBasis.cs, IClassicalBasisElement.cs, IClassicalBasis.cs, ClBasis.cs, FactoryClassicalBasisElement.cs, IClassicalBasisDebugger.cs, ClBasisDebugger.cs, IClassicalBasisElementCollection.cs, ClBasisElementCollection.cs, ExceptionClassicalBasis.cs, IQuantumBasis.cs, IQuantumBasisElement.cs, IQuantumBasisElementItem.cs, QuBasis.cs, QuBasisElement.cs, QuBasisElementItem.cs, QuBasisElementItemRepository.cs, QuBasisElementRepository.cs, QuBasisRepository.cs, FactoryQuantumBasis.cs, QuBasisDebugger.cs, IQuantumBasisDebugger.cs, IQuantumBasisElementCollection.cs, QuBasisElementCollection.cs, FactoryQuantumBasisElement.cs, ExceptionQuantumBasis.cs`

11.4.2 Hierarchy diagram

Figure 11.5: Hierarchy diagram

11.4.3 Description

The bases are a set of states, and to each one of those states is associated an identifier. In general for every state there is need of some basis, and this could imply a large amount of memory. For this reason the basis are all stored in a common repository, those basis are usually the same and only one single instance is needed. Also, this repository could evolve to a *flyweight* pattern.

The framework in it current state only makes uses of two set of states as basis. They are called `FactoryQuantumBasis.UsualBasis_A` y `FactoryQuantumBasis.UsualBasis_B`, and they are predefined as static object.

Figure 11.6: Hierarchy diagram

For the inspection of basis there are 'debuggers' that allow to show the coefficients and the identifiers of the states.

11.5 Operator

11.5.1 Classes and interfaces listing

```
 IUnitaryOperator.cs, INonUnitaryOperator.cs, IOperator.cs
IClassicalOperator.cs, AbstractClassicalOperator.cs, IClassicalRandomizeOperator.cs,
IClassicalMeasurementOperator.cs, IClassicalNoiseOperator.cs,
IClassicalBasisInitializerOperator.cs, ClBasicBasisInitializerOperator.cs,
ClBasicMeasurementOperator.cs, ClBasicNoiseOperator.cs,
ClBasicRandomizeOperator.cs, FactoryClassicalOerator.cs
FactoryQuantumOperator.cs, IQuantumBasisInitializerOperator.cs,
IQuantumNoiseOperator.cs, IQuantumMeasurementOperator.cs,
IQuantumOperator.cs, IQuantumRandomizeOperator.cs, QuBasicBasisInitializerOperator.c
QuBasicMeasurementOperator.cs, QuBasicNoiseOperator.cs,
QuBasicRandomizeOperator.cs, AbstractQuantumOperator.cs
```

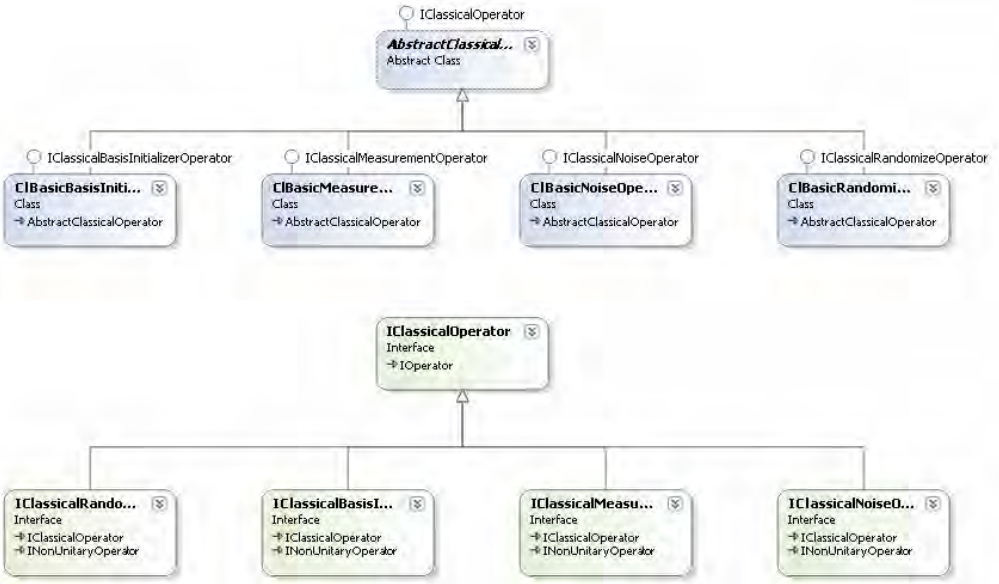

Figure 11.7: Hierarchy diagram

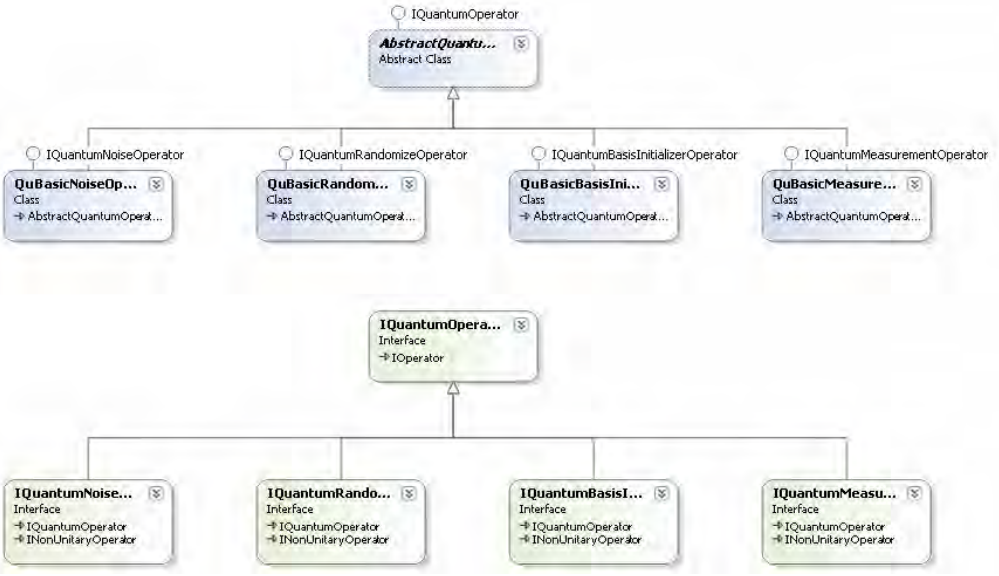

Figure 11.8: Hierarchy diagram

11.5.2 Hierarchy diagram

11.5.3 Description

This component simulates the temporal evolution of quantum states. It does not have huge memory requirements, but it is computing intensive.

To perform those calculations, it uses the 'debuggers' of the input states of the operator. It extracts the normalized coefficients with the method `getVector()`, multiplies by the corresponding matrix that defines the operator, and with the method `setVector()` modifies the output state.

Also in this component are modeled as operators the noise and the delay of the channels.

11.6 Channel

11.6.1 Classes and interfaces listing

IPointToPointChannel.cs, IStation.cs, IScalarOrientedChannel.cs, ITransmissor.cs, IVectorOrientedChannel.cs, IMultiAccessChannel.cs, AbstractStation.cs, ExceptionChannel.cs, AbstractTransmissor.cs, AbstractReceiver.cs, AbstractChannel.cs, IReceiver.cs, TransmissionMethod.cs, IChannel.cs, AbstractQuantumChannel.cs, IQuantumChannel.cs, IQuantumReceiver.cs, IQuantumTransmissor.cs, ExceptionQuantumChannel.cs, QuBasicP2PChannel.cs, IQuantumStation.cs, QuBasicTransmissor.cs, QuBasicReceiver.cs, QuBasicDifussionChannel.cs, QuBasicStation.cs, FactoryQuantumChannel.cs AbstractClassicalChannel.cs, ExceptionClassicalChannel.cs, IClassicalChannel.cs, IClassicalTransmissor.cs, IClassicalReceiver.cs, ClBasicDifussionChannel.cs, ClBasicP2PChannel.cs, ClBasicReceiver.cs, ClBasicTransmissor.cs, IClassicalStation.cs ClBasicStation.cs, FactoryClassicalChannel.cs IDualStation.cs, DuBasicStation.cs, FactoryDualChannel.cs ISniffer.cs, IStationSniffer.cs, AbstractSniffer.cs, BasicStationSniffer.cs, FactorySniffer.cs

11.6.2 Hierarchy diagram

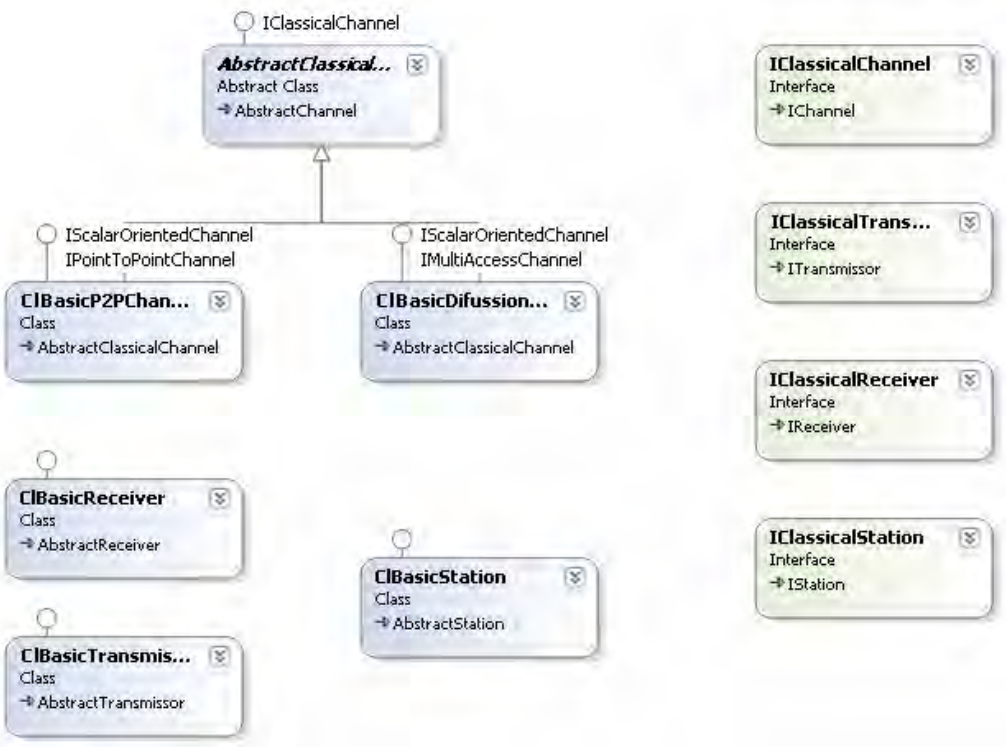

Figure 11.9: Hierarchy diagram

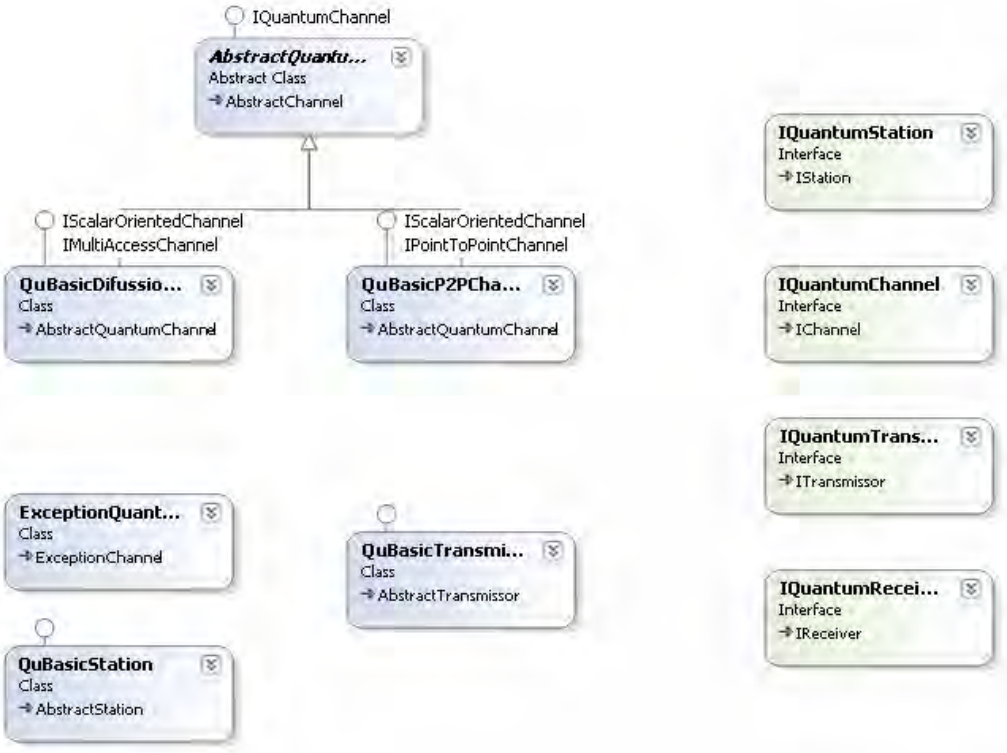

Figure 11.10: Hierarchy diagram

Figure 11.11: Hierarchy diagram

11.6.3 Description

This is a simulation in the level 1 of OSI. There is no data link, no flow control or collision detection. What is transmitted are states and not datatypes.

The datatype theory and the encoding is responsibility of the module `Data`.

Only a point to point link has been implemented. Broadcast channels are not implemented due to the lack of formal theoretical models.

For sending and receiving through the channel there are the objects `Receiver` and `Transmissor`. Those object groups as stations within the channel. An interesting point in the stations is when the station is sending states. It could also read the state sent and modify state, or do not perform any measurement.

The channels also have the option to simulate noise and delays in the transmission.

11.7 Data

11.7.1 Classes and interfaces listing

`IAbstractData.cs, IAbstractString.cs, IAbstractBit.cs, ExceptionDataError.cs, IAbstractBitArray.cs, IAbstractDataDebugger.cs, AbstractData.cs, AbstractDataDebugger.cs AbstractClassicalData.cs, IClassicalBit.cs, IClassicalData.cs, IClassicalString.cs, ClBasicString.cs, FactoryClassicalData.cs, ClBasicBit.cs, IClassicalDataDebugger.cs, ClDataDebugger.cs, ClBasicBitArray.cs, IClassicalBitArray.cs FactoryQuantumData.cs, AbstractQuantumData.cs, IQuantumData.cs, IQuantumBit.cs, IQuantumString.cs, QuBasicString.cs, QuBasicBit.cs, IQuantumBitArray.cs, QuBasicBitArray.cs, IQuantumDataDebugger.cs, QuDataDebugger.cs NativeString.cs, AbastractNativeData.cs, NativeBit.cs, FactoryNativeData.cs, INativeData.cs, INativeDataDebugger.cs, NaDataDebugger.cs`

11.7.2 Hierarchy diagram

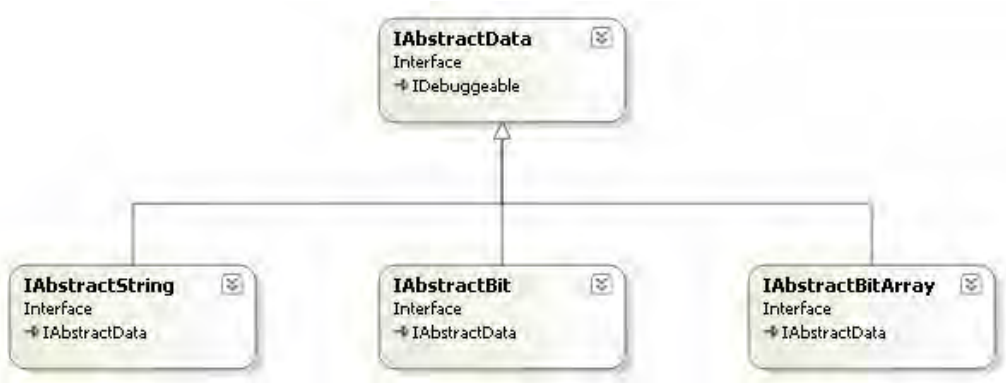

Figure 11.12: Hierarchy diagram

11.7.3 Description

It has been implemented 3 datatype categories: `Classic`, `Quantum` and `Native`. The idea of this approach is to be able to define 'Native' types, and operate with without knowing if the data is stored in quantum or classic states.

The main difference is the propagation/copy of data. This is modelled using a general method, `PropagateTo`, that does not specify how they must be transmitted. It both datatypes (source and destination) are the quantum, this method implements a teleportation (modifying the initial state). If both

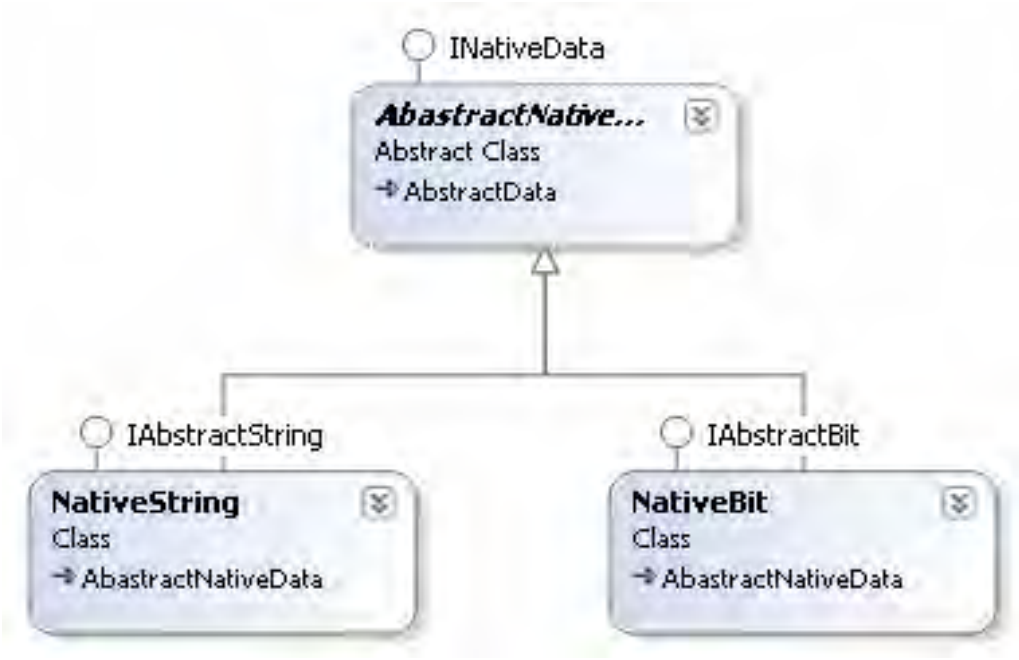

Figure 11.13: Hierarchy diagram

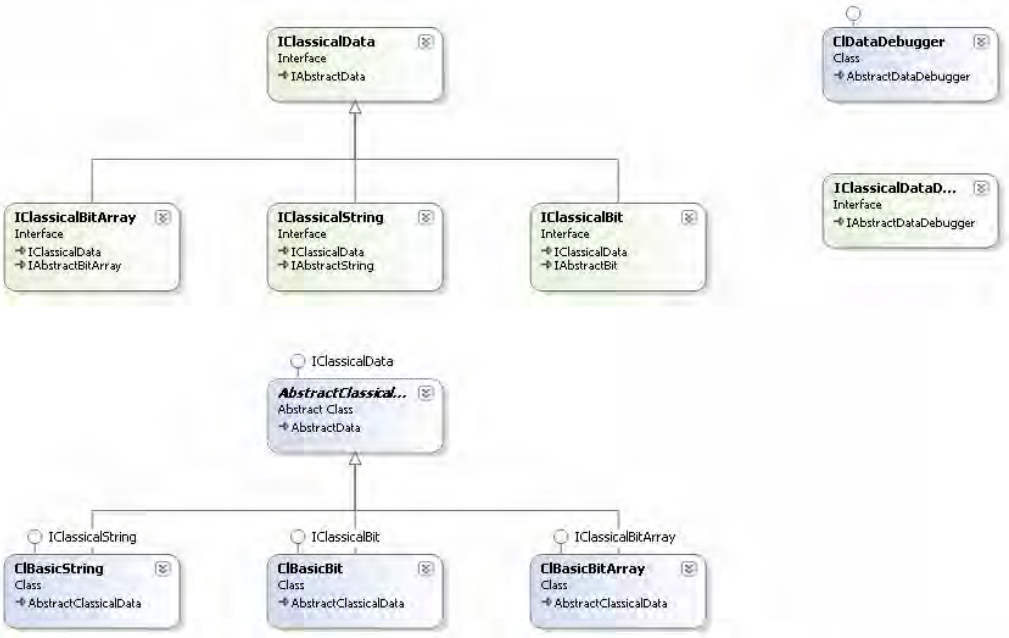

Figure 11.14: Hierarchy diagram

are classic, the state is copied without more complications. However if they are different datatypes, it is necessary to measure the state, following the encoding rules.

It has been implemented only boolean datatypes (classic and quantum) and arrays of boolean (classic and quantum), because they are the only ones required in BB84.

The data is defined as physical states using some encoding specification. This encoding has to be *standarized* for the interoperability of different systems. This encoding is also based in the so called 'type theory', for

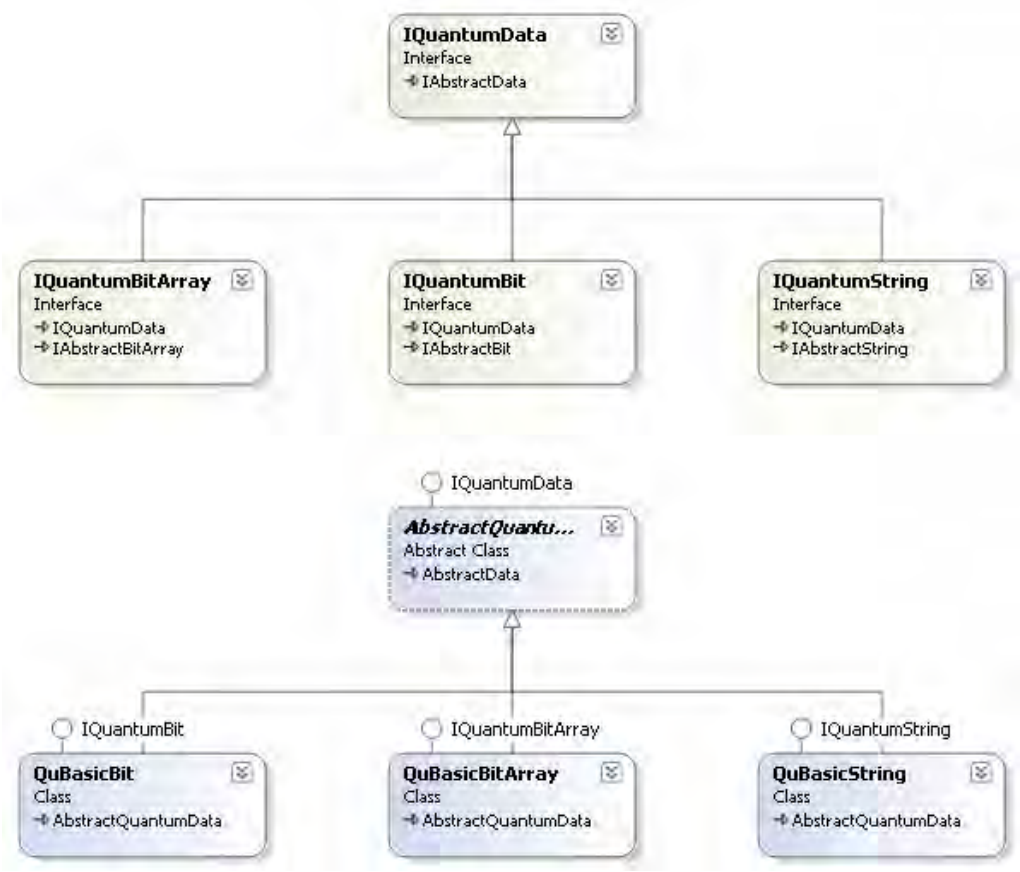

Figure 11.15: Hierarchy diagram

this reason there is need of a 'quantum data type theory'.

Internally, the data is represented using a physical state, it is necessary to define methods for working with the internal state. The idea is not to alter the state, but to know what No se trata de alterarlo, sino establecer cuál es el estado que se va a usar para almacenar la información. Para ello existe la propiedad de lectura/escritura `InternalState`. Esta propiedad no se crea con el tipo de datos, sino que es necesaria asignarla previamente para poder almacenar o leer información.

11.8 Protocol BB84

11.8.1 Classes and interfaces listing

`IProtocol.cs, AbstractAgent.cs, IAgent.cs IProtocolBB84.cs, ProtocolBB84_CSS.cs, ProtocolBB84_QKD.cs, ProtocolBB84_Chau.cs, IAutomataBB84.cs, FactoryProtocolBB84.cs, ProtocolBB84_Secure.cs, AbstractProtocolBB84.cs, IAgentBB84.cs, AgentBB84.cs, ExceptionProtocolBB84.cs`

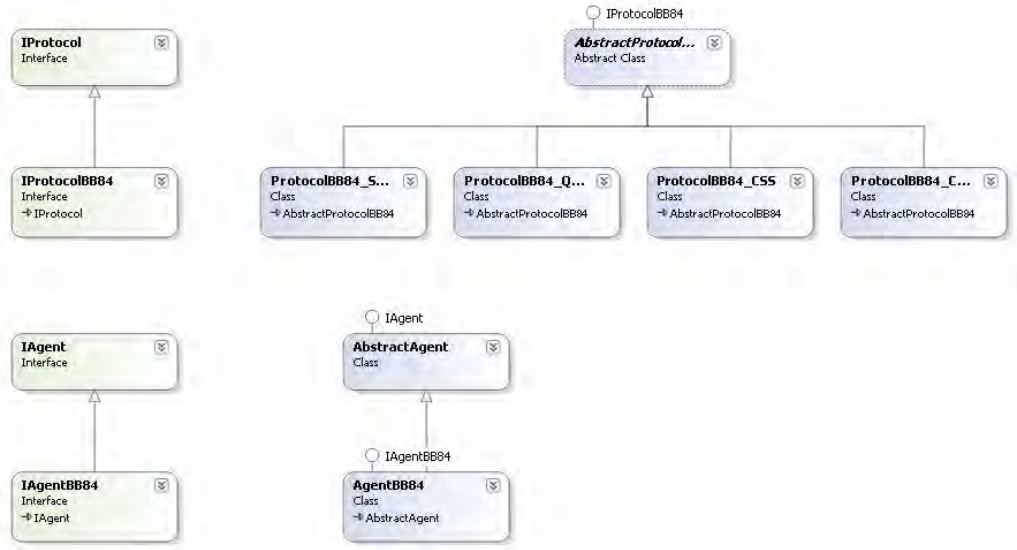

Figure 11.16: Hierarchy diagram

11.8.2 Hierarchy diagram

11.8.3 Description

It is an implementation of the BB84 algorithm for the quantum keys distribution. This is a communication protocol of level 2 in the OSI classification. It only establish the logic to send data but does not care about collisions, error correction or detection of stations in the channel.

Furthermore, the logic of the protocol is implemented as Finite State Machines. This Finite State Machine (FSM) drives the agents. *Alice*, *Bob* and *Eve*, define events for the reception of states, they store the received values, processing it and obtaining the new data. When the protocol ends the data transfer, if case of success, *Alice* and *Bob* will have stored in the property `SecretSecureKey` a boolean array of length N, being N the size of the data to be ciphered.

The protocol does not act in the communication channels, but by commanding the the agents. Consequently, the agents interacts with the physical channels.

In contrast with the communication channels, the protocol deals with data types, and the channel with physical states (quantum or classic). The encoding has to take place in the agent.

11.9 Cryptosystem

11.9.1 Classes and interfaces listing

`ICryptosystem.cs, AbstractCryptosystem.cs, ExceptionCryptosystem.cs IVernam.cs, BasicVernam.cs IVigenere.cs, BasicVigenere.cs`

Figure 11.17: Hierarchy diagram

11.9.2 Hierarchy diagram

11.9.3 Description

This component implements the classic cryptographic systems, such as Vernam and Vigènere. It is loosely integrated in the rest of the framework.

11.10 Testing system

For checking the integrity of the system several types of unitary tests are implemented. The aim is to produce a coverage for most of the framework functionalities.

However, even being quite exhaustive they are not a 100% verification of the code.

Those unitary tests are directly accessible from the graphical interface, but there is not a direct dependence on it. They conform a totally independent subsystem in parallel to the framework itself.

The unitary tests are defined inside each component of the framework, while the load of the component they are registered into a repository. The graphical interface just explores this repository and generate the proper menu entries for instantiating the tests.

11.11 Configuration options

Most of the configuration options are related to the level of verbosity of the simulations. Each framework component raises its own messages with its own configuration.

Also the noise levels, the random numbers generation, the quantum basis and the graphical properties can be set.

In resume, all the global parameters of the framework and the user interface are reflected somewhere in the configuration menu.

(This page is intentionally left blank)

Chapter 12

How to develop with the framework

12.1 Programing techniques

The idea of this section is to resume and gather ideas about the usage of the framework already given in the framework architecture section.

12.2 Random

To generate two random numbers and used them to initialize a complex number.

```
1 public void Randomize()
2 {
3   setPartReal( FactoryRandom.GetSingleIntenger() );
4   setPartComplex( FactoryRandom.GetSingleIntenger() );
5 }
```

Listing 12.1: Generating a random complex number

12.3 States

To generate a random state (default initialization), and show the coefficients using the debugging subsystem.

```
1 IQuantumState stateA = FactoryQuantumState.generateStateSimple(count);
2 TesterLog("Debugging Random QuantumStateSimple A:");
3 TesterLog(stateA.generateQuantumDebugger().FullDebug);
```

Listing 12.2: Generating an unknown quantum state

12.4 Math

Multiplication of a vector by a complex number.

```
public IComplexMatrix getVectorNormalized()
{
  IComplexMatrix vector = getVectorInternal();
  IComplexNumber coeff = FactoryComplexNumber.generateComplexNumber(
      generateNormalizationCoefficient(),0);
  return FactoryComplexMatrix.Operations.Multiplication(coeff, vector);
}
```

Listing 12.3: Multiplying a vector by a complex number

12.5 Data

To generate a qubit, initializate it to *false* while using some specific basis.

```
bool data = fase;
IQuantumBit quantumBit = FactoryQuantumData.generateBit();
quantumBit.ReferenceBasis = FactoryQuantumBasis.UsualBasis;
quantumBit.setBit(data);
return quantumBit;
```

Listing 12.4: Generating and storing false in a qubit

12.6 Operator

To generate noise in a specific state.

```
IQuantumNoiseOperator noiser = FactoryQuantumOperator.generateNoiseOperator();
noiser.UseGlobalConfiguration = fase;
noiser.Configuration.NoiseFactor = 0.3;
noiser.Evaluate((IQuantumState)state);
```

Listing 12.5: Adding noise into a state

12.7 Channel

To define a channel, an state and the transmission of that state over the channel.

```
IDualStation station = FactoryDualChannel.generateStation();
IClassicalState ClStateTransmissed = FactoryClassicalState.generateState(
    FactoryClassicalBasis.UsualBasis.CountElements);
station.ClassicalTransmitSynchronous(ClStateTransmissed);
```

Listing 12.6: Transmmision of a quantum state

12.8 Protocol

Define a quantum channel, a classic channel and to assign them to a protocol agent.

```
IClassicalChannel classicalChannel = FactoryClassicalChannel.generateP2PChannel()
    ;
IQuantumChannel quantumChannel = FactoryQuantumChannel.generateP2PChannel();
IAgentBB84 alice = FactoryProtocolBB84.generateAgentAlice();
alice.ClassicalChannel = classicalChannel;
alice.QuantumChannel = quantumChannel;
```

Listing 12.7: Setting up a quantum protocol

12.9 BB84 ciphering code generation

```
1  string dataSecret = "Hello!";
2  bool[] UTF8BoolArrayData = ArrayTools.EncodingUTF8_StringToArrayBool(dataSecret);
3  TesterLog("GENERATING COMMON SECRET KEY FOR TEXT '" + dataSecret + "' (Length: "
       + dataSecret.Length + ") ");
4  TesterLog("   convernting into binary... '" +
5  ArrayTools.BoolArrayToString(UTF8BoolArrayData) + "' (Length: " +
       UTF8BoolArrayData.Length + ") ");
6  try
7  {
8    protocol.Process(UTF8BoolArrayData.Length);
9    TesterLog("KEY DISTRIBUTION SUCCESSFUL");
10 }
11 catch (ExceptionProtocolBB84KeySize)
12 { TesterLog("INVALID INTERMEDIARY KEY SIZE.... BACK LUCK,,,, TRY AGAIN....");  }
13 catch (ExceptionProtocolBB84KeyPartialComparison)
14 { TesterLog("ERROR DURING KEY PARTIAL COMPARISON.... POSSIBLE EVASDROPER OR HIGH
       NOISE...");  }
```

Listing 12.8: BB84 minimal

(This page is intentionally left blank)

Chapter 13

How to use the simulator

13.1 Main window

13.1.1 Introduction

In this screen is shown the global interface of the program. Several unitary tests are registered, the so-called `Testers`, and there are also configuration options.

The windows that appear below the menu correspond to the debugging system. Whenever a type of `Tester` is used, the corresponding messages are provided by an associated window.

All the generated messages are presented by means of the GLOBAL window, including the interns of framework, are not presented in the individual `Tester` windows.

13.1.2 Configuration options

With the option random several random number generators can be selected. The recommend choice is `RandomDotOrg` or `RandomHotBits`, using authentic entropy sources available through Internet.

Nevertheless a connection to Internet is required, and for that reason the default is `RandomStored`, that are stored 'random numbers' from the previous sources. This provides better results that `RandomLanguage`, that is the default random number generator of C#.

The menu `Operators`, is had an option to manage the noise level. By defect it is deactivated, and for didactic aims this is most recommendable. It only is useful for anecdotal verifications.

There are also options to show the probability distributions in the measurement process, and if to apply delays in the communication channels.

The reference quantum bases, are totally modifiable during run time. They are predefined to the usual values for BB84, but it is possible to change them without problem. In addition it is not either necessary to introduce the standard coefficients, they will be normalized internally.

In order to control the protocol BB84, exists another configuration menu. It allows to indicate the size of the parameter δ, as well as if to carry out the verification phase on the generated key in order to detect

Figure 13.1: Main window

possible spies (activated by defect).

It is also possible to set a debugging of all the events raised during the protocol simulation. Those messages can be showed one by one when the are generated (quite slow) or in groups in a *bulk* mode, showing all of them at the end of the process.

Also it is possible to activate debugging messages for all events that take place in protocol. Similarly the shown messages can be show at the same instant that they are generated (quite slow) or to do it in way *bulk*, showing all this debug information at the end of the process.

13.2 Testers

13.2.1 Introduction

The `Tester`, are grouped in diverse categories according to their thematic. Basically they serve for unitary tests during the framework development, but also they illustrate part of the simulation process.

I will emphasize most relevant ones.

13.2.2 QuantumWorld.States

Figure 13.2: Configuration window

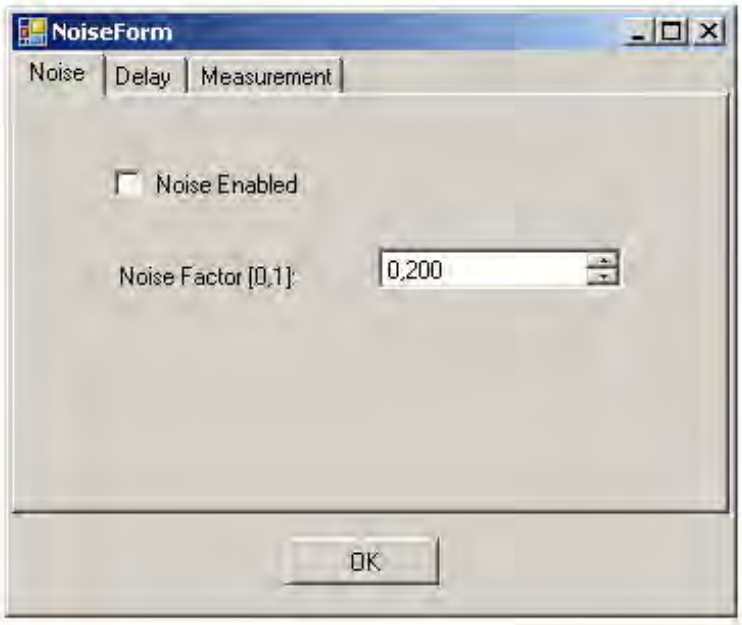

Figure 13.3: Noise window

```
1  --- START: QuantumWorld.States ---
2
3  Debugging Random QuantumStateIsolated:
4  QuantumState Implementing Class:
5      QuantumInformationFramework.State.Quantum.QuStateIsolated
6  QuantumState Inner Vector/Matrix:
7      Internal Values: [ [ (+7708000, -1983000i) ], [ (+6383200, +5681400i) ] ]
8      Normalized Values: [ [ (+0,08981, -0,02310i) ], [ (+0,74375, +0,66198i) ] ]
9  Debugging Random QuantumStateSimple A:
```

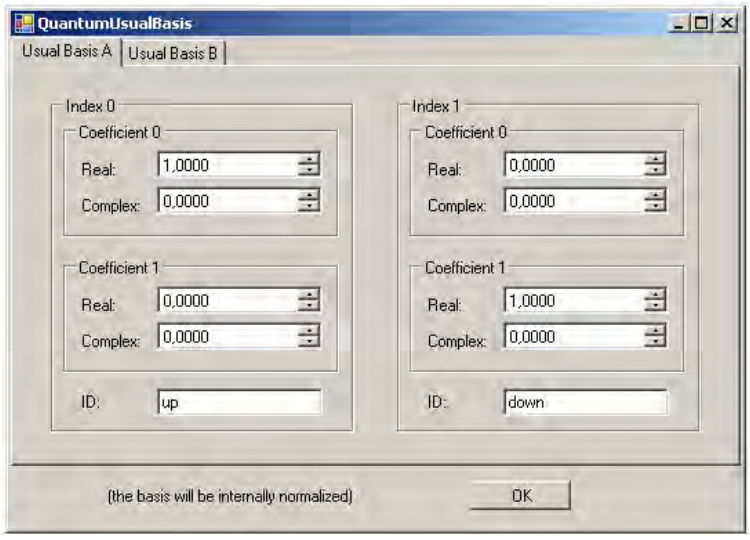

Figure 13.4: Basis window

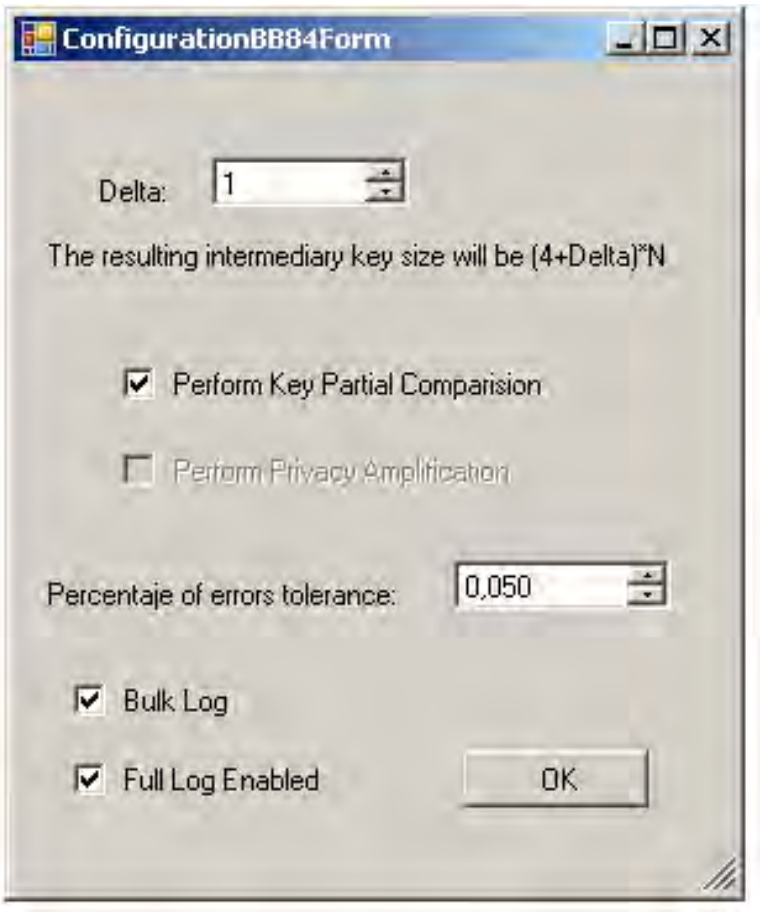

Figure 13.5: BB84 window

```
10  QuantumState Implementing Class:
11      QuantumInformationFramework.State.Quantum.QuStateSimple
12  QuantumState Inner Vector/Matrix:
13      Internal Values: [ [ (-3458000, -9670900i)], [ (-8056800, -3858400i) ] ]
```

```
14      Normalized Values: [ [ (-0,02625, -0,73432i) ], [ (-0,61176, -0,29297i) ] ]
15  Debugging Random QuantumStateSimple B:
16  QuantumState Implementing Class:
17      QuantumInformationFramework.State.Quantum.QuStateSimple
18  QuantumState Inner Vector/Matrix:
19      Internal Values: [ [ (-8759500, +8580500i)], [ (+9677000, +4416200i) ] ]
20      Normalized Values: [ [ (-0,67026, +0,65656i) ], [ (+0,07404, +0,33791i) ] ]
21  Debugging QuantumStateComposed from A and B:
22  QuantumState Implementing Class:
23      QuantumInformationFramework.State.Quantum.QuStateComposed
24  QuantumState Inner Vector/Matrix:
25      Internal Values: [ [ (-3458000, -9670900i)], [ (-8056800, -3858400i) ], [
            (-8759500, +8580500i) ], [ (+9677000,+4416200i) ] ]
26      Normalized Values: [ [ (-0,01863,
27  -0,52123i) ], [ (-0,43424, -0,20795i) ], [ (-0,47211, +0,46246i) ], [(+0,05215,
        +0,23802i) ] ]
28  Debugging FULL QuantumStateComposed from Isolated and
29  Composed:
30  QuantumState Implementing Class:
31      QuantumInformationFramework.State.Quantum.QuStateComposed
32  QuantumState Inner Vector/Matrix:
33      Internal Values: [ [ (+7708000, -1983000i)], [ (+6383200, +5681400i) ], [
            (-3458000, -9670900i) ], [ (-8056800, -3858400i) ], [ (-8759500, +8580500
            i) ], [ (+9677000, +4416200i) ] ]
34      Normalized Values: [ [ (+0,03770,-0,00970i) ], [ (+0,31225, +0,27792i) ], [
            (-0,01691, -0,47307i) ], [(-0,39412, -0,18874i) ], [ (-0,42849, +0,41973i
            ) ], [ (+0,04733,+0,21603i) ] ]
35  Level 0 has as normalized coefficient (+0,03770, -0,00970i) and powerednorm
        0,00151582370980731
36  Level 1 has as normalized coefficient (+0,31225, +0,27792i) and powerednorm
        0,174741273054822
37  Level 2 has as normalized coefficient (-0,01691, -0,47307i) and powerednorm
        0,224089372106651
38  Level 3 has as normalized coefficient (-0,39412, -0,18874i) and powerednorm
        0,190955213223754
39  Level 4 has as normalized coefficient (-0,42849, +0,41973i) and powerednorm
        0,359788296384456
40  Level 5 has as normalized coefficient (+0,04733, +0,21603i) and powerednorm
        0,0489100215205099
41  All the powered norms SUMS 1
42
43  --- END: QuantumWorld.States in 100 milliseconds ---
```

Listing 13.1: QuantumWorld.States output

In this point can be observed that the internal coefficients are different from the standardized ones, and in addition that the normalization constant is correct (the sum of all the coefficients norms to the square must be 1).

13.2.3 QuantumWorld.Measurement

```
1  QuantumWorld.Measurement : --- START: QuantumWorld.Measurement ---
2
3  QuantumWorld.Measurement : State before measurement
4  QuantumWorld.Measurement :
5  QuantumState Implementing Class:
6      QuantumInformationFramework.State.Quantum.QuStateSimple
7  QuantumState Inner Vector/Matrix:
8      Internal Values: [ [ (-9015200, +8000500i)], [ (-2815300, -7221900i) ] ]
9      Normalized Values: [ [ (-0,62909, +0,55828i) ], [ (-0,19645, -0,50395i) ] ]
10 DEFAULT : Measurament Probabilities Distribution: [0,707437159791861,
       0,292562840208139 ]
11 QuantumWorld.Measurement : State after measurement into UsualBasis => 0
12 QuantumWorld.Measurement : QuantumState Implementing Class:
```

```
13      QuantumInformationFramework.State.Quantum.QuStateSimple
14  QuantumState Inner Vector/Matrix:
15      Internal Values: [ [ (+1000000, +0000000i)], [ (+0000000, +0000000i) ] ]
16      Normalized Values: [ [ (+1000000,+0000000i) ], [ (+0000000, +0000000i) ] ]
17  DEFAULT : Measurament Probabilities Distribution: [ 1,0 ]
18  QuantumWorld.Measurement : State after second measurement into UsualBasis => 0
19  QuantumWorld.Measurement : QuantumState Implementing Class:
20      QuantumInformationFramework.State.Quantum.QuStateSimple
21  QuantumState Inner Vector/Matrix:
22      Internal Values: [ [ (+1000000, +0000000i)], [ (+0000000, +0000000i) ] ]
23      Normalized Values: [ [ (+1000000, +0000000i) ], [ (+0000000, +0000000i) ] ]
24
25  --- END:QuantumWorld.Measurement in 30 milliseconds ---
```

Listing 13.2: QuantumWorld.Measurement output

In listing can be observed how the measurement alters the internal states, and that successive measures on a same state, produce the same result (the state already has collapsed).

13.2.4 QuantumWorld.Data

```
1   QuantumWorld.Data : --- START: QuantumWorld.Data ---
2
3   QuantumWorld.Data : QuantumBit Integrity Test:
4   QuantumWorld.Data : Store FALSE...
5   QuantumWorld.Data :
6       Internal state => [ [ (+1000000, +0000000i) ], [ (+0000000, +0000000i) ] ]
7   QuantumWorld.Data : Read Data:
8   DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
9   QuantumWorld.Data : False
10  QuantumWorld.Data :
11      Internal state => [ [ (+1000000, +0000000i) ], [ (+0000000, +0000000i) ] ]
12  QuantumWorld.Data : Store TRUE...
13  QuantumWorld.Data :
14      Internal state => [ [ (+0000000, +0000000i) ], [ (+1000000, +0000000i) ] ]
15  QuantumWorld.Data : Read Data:
16  DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
17  QuantumWorld.Data : True
18  QuantumWorld.Data :
19      Internal state=> [ [ (+0000000, +0000000i) ], [ (+1000000, +0000000i) ] ]
20
21  QuantumWorld.Data : QuantumBitArray Integrity Test:
22  QuantumWorld.Data : Store 00101010...
23  QuantumWorld.Data : Internal state => [ [ (+0,35355, +0000000i) ], [ (+0000000,
        +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [
        (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0,35355, +0000000i)
        ], [ (+0000000, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,35355,
        +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [
        (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0,35355, +0000000i)
        ], [ (+0000000, +0000000i) ] ]
24  QuantumWorld.Data : Read Data:
25  DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
26  DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
27  DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
28  DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
29  DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
30  DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
31  DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
32  DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
33  QuantumWorld.Data : 00101010
34  QuantumWorld.Data :
35      Internal state => [ [ (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [
            (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0000000, +0000000
            i) ], [ (+0,35355, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0000000,
            +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [
```

```
          (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0000000, +0000000
          i) ], [ (+0,35355, +0000000i) ], [
(+0,35355, +0000000i) ], [ (+0000000, +0000000i) ] ]

QuantumWorld.Data : RANDOM QuantumBitArray Integrity Test:
QuantumWorld.Data : Store 11000100...
QuantumWorld.Data : Internal state
=\begin{math}>\end{math} [ [ (-0,03457, -0,16867i) ], [ (+0,11327,
+0,06901i) ], [ (-0,09601, +0,08966i) ], [ (+0,05863, +0,19494i) ], [
(+0,11541, -0,26980i) ], [ (+0,21999, -0,19579i) ], [ (+0,32191,
-0,00024i) ], [ (-0,11804, -0,15455i) ], [ (+0,18406, -0,06802i) ], [
(-0,26135, -0,26484i) ], [ (+0,23703, +0,31837i) ], [ (+0,22433,
-0,09670i) ], [ (+0,28393, +0,05304i) ], [ (-0,07453, -0,17326i) ], [
(+0,15992, +0,08230i) ], [ (+0,08757, +0,16296i) ] ]
QuantumWorld.Data : Read Data:
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
QuantumWorld.Data : 11000100
QuantumWorld.Data :
    Internal state => [ [ (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [
        (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0,35355, +0000000
        i) ], [ (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0000000,
        +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0000000, +0000000i) ], [
        (+0000000, +0000000i) ], [ (+0,35355, +0000000i) ], [ (+0,35355, +0000000
        i) ], [ (+0000000, +0000000i) ], [
(+0,35355, +0000000i) ], [ (+0000000, +0000000i) ] ]
QuantumWorld.Data : QuantumBitArray RANDOM CODING:
QuantumWorld.Data : Random basis: 11000110
QuantumWorld.Data : Erroneus basis: 01110011
QuantumWorld.Data : Store 00101010...
DEFAULT : Selected UsualBasis_B for bit encoding
DEFAULT : Selected UsualBasis_B for bit encoding
DEFAULT : Selected UsualBasis_A for bit encoding
DEFAULT : Selected UsualBasis_A for bit encoding
DEFAULT : Selected UsualBasis_A for bit encoding
DEFAULT : Selected UsualBasis_B for bit encoding
DEFAULT : Selected UsualBasis_B for bit encoding
DEFAULT : Selected UsualBasis_A for bit encoding
QuantumWorld.Data :
Internal state[ [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [
    (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i)
    ], [
(+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [
    (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i)
    ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (-0,28867,
    +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ] ]
QuantumWorld.Data : CORRECT Read Data:
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
```

```
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
QuantumWorld.Data : 00101010
QuantumWorld.Data : Internal state => [ [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (−0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ] ]
QuantumWorld.Data : ERRONEOUS Read Data:
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
QuantumWorld.Data : 10101110
QuantumWorld.Data :
    Internal state => [ [ (+0000000, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (−0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (−0,27735, +0000000i) ], [ (+0,27735, +0000000i) ], [ (+0,27735, +0000000i) ] ]

QuantumWorld.Data : AGAIN CORRECT Read Data:
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [0,5, 0,5 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 1, 0 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
DEFAULT : Selected UsualBasis_B for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0, 1 ]
DEFAULT : Selected UsualBasis_A for bit decoding
DEFAULT : Measurament Probabilities Distribution: [ 0,5, 0,5 ]
QuantumWorld.Data : 10001111
QuantumWorld.Data :
    Internal state => [ [ (+0,28867, +0000000i) ], [ (−0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (−0,28867, +0000000i) ], [ (+0,28867, +0000000i) ], [ (−0,28867, +0000000i) ], [ (+0000000, +0000000i) ], [ (+0,28867, +0000000i) ] ]

--- END: QuantumWorld.Data in 460 milliseconds ---
```

Listing 13.3: QuantumWorld.Data output

This other listing, tries to illustrate the codification of logical data to quantum states.

This tester just checks it reads just like it writes. It is done for individual bits as for arrays of bits, that in quantum data types are implemented as a `ComposedState`.

Also it illustrates how the measurement alters the internal state. The writing and the reading is done in a coherent way, using the same basis. In a later step the basis are changed and a wrong readout is executed. When returning to the original basis for reading, the value that was stored initially are no longer accessible. The readout with the wrong basis damaged the internal quantum state.

13.2.5 KeyDistributionBB84.Simple

```
1  --- START: KeyDistributionBB84.Simple ---
2
3  GENERATING COMMON SECRET KEY FOR TEXT 'Hello!' (Length: 6)
4      convernting into binary...
5  '010010000110010101101100011011000110111100100001' (Length: 48)
6  KEY DISTRIBUTION SUCCESSFUL
7  ALICE COMMON SECRET SECURE KEY: 110000110001101100010101001001110100001100111101
       (Length: 48)
8  BOB   COMMON SECRET SECURE KEY: 110000110001101100010101001001110100001100111101
       (Length: 48)
9
10 --- END: KeyDistributionBB84.Simple in 1341 milliseconds ---
```

Listing 13.4: KeyDistributionBB84.Simple output

It simulates protocol BB84, generating a cryptographic key common and secret. In fact, the process is neither so direct nor as simple as is shown in the reduced listing.

For this part, it is necessary to have read Section 6.2, *Protocol BB84*, where the protocol is explained in detail.

According to the listing GLOBAL, in a sequential order, several parts are relevant...

```
1  GENE: #--#--#--#--#--#--#--#--#--#--#--#
2  GENE:
3  GENE: BASE KEY LENGTH FOR QUANTUM TRANSMISSION: 240
4  GENE:
5  GENE: #--#--#--#--#--#--#--#--#--#--#--#
```

Listing 13.5: KeyDistributionBB84.Simple default output

It is necessary to calculate the size of the intermediate key, this one will be reduced because of wrong guessing selecting the measurement basis, and with a surplus to detect spies (a formula dependent on δ).

```
1  GENE: STEP 1: 0 | Master randomly generated Data Bit: False
2  Selected UsualBasis_B for bit encoding
3  GENE: STEP 1: 0 | Master Send Quantum State: [ [ (+0,70710, +0000000i) ], [
       (+0,70710, +0000000i) ] ]
4  GENE: STEP 1: 0 | Slave State Received: [ [ (+0,70710, +0000000i) ], [ (+0,70710,
        +0000000i) ] ]
5  Selected UsualBasis_A for bit decoding
6  Measurament Probabilities Distribution: [ 0,5, 0,5 ]
7  GENE: STEP 1: 0 | Slave conversion into Bit:  True
8  GENE: STEP 1: 1 | Master randomly generated Data Bit: True
9  Selected UsualBasis_A for bit encoding
10 GENE: STEP 1: 1 | Master Send Quantum State: [ [ (+0000000, +0000000i) ], [
       (+1000000, +0000000i) ] ]
11 GENE: STEP 1: 1 | Slave State Received: [ [ (+0000000, +0000000i) ], [
       (+1000000, +0000000i) ] ]
12 Selected UsualBasis_A for bit decoding
13 Measurament Probabilities Distribution: [ 0, 1 ]
```

```
14 GENE: STEP 1: 1 | Slave conversion into Bit: True
```

Listing 13.6: KeyDistributionBB84.Simple default output

Here the *master station* sends quantum states, that codify random bits, and using bases also selected randomly. The receiver also measures randomly (the measurement operator informs us about the probability distribution for each state). After the measurement the receiver interprets the state and obtains a certain *true*/*false* value.

```
1 GENE: STEP 2: 0 | Master end Classical State: 5
2 GENE: STEP 2: 0 | Slave State Receive: 5
3 GENE: STEP 2: 0 | Slave conversion into Bit: True
4 GENE: STEP 2: 1 | Master Send Classical State: 0
5 GENE: STEP 2: 1 | Slave State Receive: 0
6 GENE: STEP 2: 1 | Slave conversion into Bit: False
7 GENE: STEP 2:
8 ==========> Slave found coincidence in Basis at POS: 1
```

Listing 13.7: KeyDistributionBB84.Simple default output

In this other step the emitter sends classic states informing in which bases sent the quantum data, and the receiver (Bob) stores the proper quantum bases sequence.

```
1 GENE: STEP 3: 0 | Slave Send Classical State: 0
2 GENE: STEP 3: 0 | Master State Receive: 0
3 GENE: STEP 3: 0 | Master conversion into Bit: False
4 GENE: STEP 3: 1 | Slave Send Classical State: 5
5 GENE: STEP 3: 1 | Master State Receive: 5
6 GENE: STEP 3: 1 | Master conversion into Bit: True
7 GENE: STEP 3:
8 ==========> Master found coincidence in Basis at POS: 1
```

Listing 13.8: KeyDistributionBB84.Simple default output

Now Bob sends to Alice, in which basis he measured, in this way both can calculate the coincidences with the other, and use those bits codified and measured in the same bases to generate a intermediate key.

```
1 #--#--#--#--#--# End MasterClassicalCoincidences Receive #--#--#--#--#--#
2 GENE: MASTER SECRET SECURE KEY:
      1001101101000110011011010001100110110100011001101101000110011011010001100110
3 GENE: SLAVE SECRET SECURE KEY:
      1001101101000110011011010001100110110100011001101101000110011011010001100110
4 CHEC: #--#--#--#--#--#--#--#--#--#--#--#--#--#
```

Listing 13.9: KeyDistributionBB84.Simple default output

At this moment a verification of size is realised. This intermediate key must have a size of at least $2N$ bits, being N the number of bits that needs to be ciphered. Otherwise, the protocol is aborted.

```
1 CHEC: RANDOM POSITIONS FOR COMPARISON: [ 78, 90, 72, 26, 77, 59, 23, 98, 33, 46,
      102, 54, 113, 7,14, 75, 116, 97, 111, 44, 100, 36, 127, 66, 25, 95, 19, 41,
      29, 56, 108, 13, 2, 37, 32, 101, 9, 1, 28, 6, 79, 0, 67, 61, 109, 89, 112,
      117, 121, 12, 8, 123, 45, 38, 62, 115, 124, 10, 48, 43, 27, 81, 122, 106, 40,
       125, 69, 119, 128, 31, 71, 83, 104, 17, 57, 34, 73, 39, 76, 74, 120, 50, 60,
       20, 55 ]
2 CHEC: RANDOM POSITIONS FOR COMPARISON ORDERED: [ 0, 1, 2, 6, 7, 8, 9, 10, 12, 13,
       14, 17, 19, 20, 23, 25, 26, 27, 28, 29, 31, 32, 33, 34, 36, 37, 38, 39, 40,
      41, 43, 44, 45, 46, 48, 50, 54, 55, 56, 57, 59, 60, 61, 62, 66, 67, 69, 71,
      72, 73, 74, 75, 76, 77, 78, 79, 81, 83, 89, 90, 95, 97, 98, 100, 101, 102,
      104, 106, 108, 109, 111, 112, 113, 115, 116, 117, 119, 120, 121, 122, 123,
      124, 125, 127, 128 ]
```

```
```

Listing 13.10: KeyDistributionBB84.Simple default output

For the verification phase, they are had to select $Z\ minus N$ positions between Z different positions that occupy Z elements that compose the intermediate key (being N the number of bits that needs to be ciphered).

```
1 CHEC: STEP 1: 2 | Master informs if POS 2 will be used for comparison. (True)
2 CHEC: STEP 1: 2 | Master Send Classical State: 5
3 CHEC: STEP 1: 2 | Slave State Received: 5
4 CHEC: STEP 1: 2 | Slave conversion into Bit: True
5 CHEC: STEP 1: 3 | Master informs if POS 3 will be used for comparison. (False)
6 CHEC: STEP 1: 3 | Master Send Classical State: 0
7 CHEC: STEP 1: 3 | Slave State Received: 0
8 CHEC: STEP 1: 3 | Slave conversion into Bit: False
```

Listing 13.11: KeyDistributionBB84.Simple default output

Now the master sends to the slave, what indices of the common key are going to be used for comparison.

```
1 CHEC: STEP 2: 2 | Master informs value of POS 2 ==> False
2 CHEC: STEP 2: 2 | Master Send Classical State: 0
3 CHEC: STEP 2: 2 | Slave State Receive: 0
4 CHEC: STEP 2: 2 | Slave conversion into Bit: False
5 CHEC: STEP 2: 2 | Slave stores for checking check POS 2 with VALUE False
6 CHEC: STEP 2: 3 | Master bypass value of POS 3
7 CHEC: STEP 2: 3 | Master bypass value of POS 4
```

Listing 13.12: KeyDistributionBB84.Simple default output

Next the master checks the intermediate key, bit by bit, if it position were not selected to compare, does not inform on the value. If on the contrary, it was were selected, it makes public the value that wanted to send by means of quantum channel.

```
1 CHEC: STEP 3: 2 | Slave informs value of POS 2 ==> False
2 CHEC: STEP 3: 2 | Slave Send Classical State: 0
3 CHEC: STEP 3: 2 | Master State Receive: 0
4 CHEC: STEP 3: 2 | Master conversion into Bit: False
5 CHEC: STEP 3: 2 | Master stores for checking check POS 2 with VALUE False
6 CHEC: STEP 3: 3 | Slave bypass value of POS 3
7 CHEC: STEP 3: 3 | Slave bypass value of POS 4
```

Listing 13.13: KeyDistributionBB84.Simple default output

The slave station conducts the symmetrical action, it sends to the master what it received measuring using the correct bases. Any dissonance will have to be noise, or to the existence of a spy.

```
1 CHEC: Master Error Level 0
2 CHEC: Slave Error Level 0
```

Listing 13.14: KeyDistributionBB84.Simple default output

As both stations, have what they wanted to send and was really received, they compare it. If they find that the error rate is over a pre-established limit, the transmission is aborted. They suspect the existence of a possible spy.

```
1 MASTER SECRET SECURE KEY: 11000011000110110001010100100111010000110011110 1
2 SLAVE SECRET SECURE KEY: 11000011000110110001010100100111010000110011110 1
```

Listing 13.15: KeyDistributionBB84.Simple default output

To finalize, each station generates a new key with N remaining bits that have not been compared publicly. This key will be the one that will be used to cipher the data with the Vernam' algorithm.

13.2.6 Interactive.BB84Cryptosystem

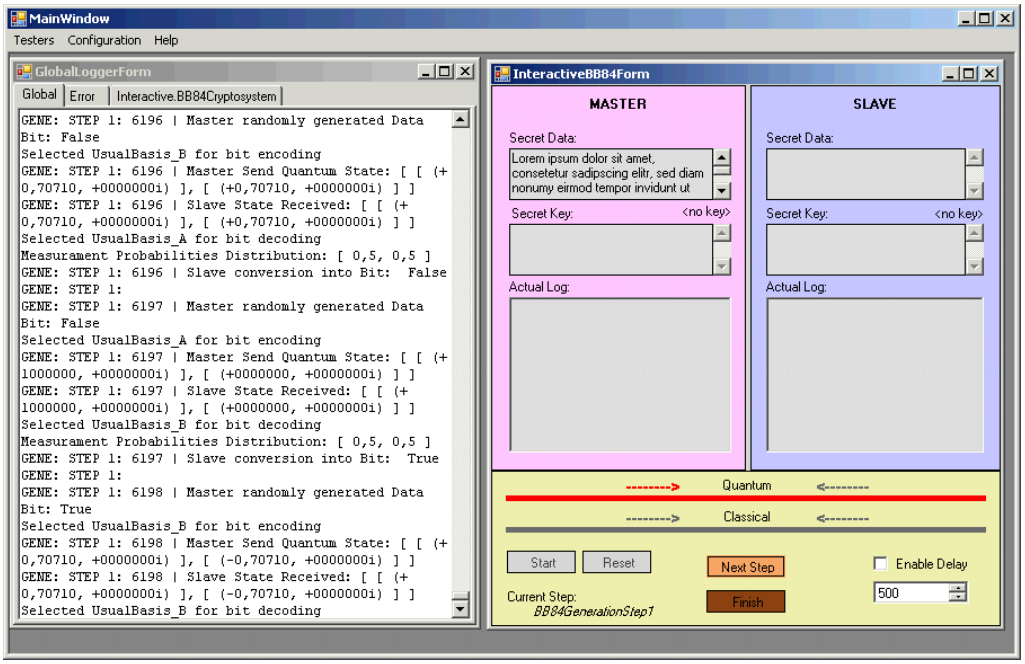

Figure 13.6: BB84 execution

It is like `KeyDistributionBB84.Simple` but interactive. Among each stage it makes a pause, showing one explanation than happens in each stage. Also presents in a graphical way the channels in use at any moment. In the same way, the text to transmit in a ciphered way is modifiable by the user.

Chapter 14

Requirements covering

14.1 Functional requirements

14.1.1 Randomness

- COMPLETE
- Managed by the component `Random`.
- Is one of the best performing components. Comparing among the entropy sources available on Internet, and the built-in random number generators the differences are quite appreciable. The .NET implementation start cycling the results. Each simulation consumes around 500 random numbers, after 8 BB84 simulations, the global results start being the same.

14.1.2 Matricial operations with complex numbers

- COMPLETE
- Managed by the component `Math`, depending on `Random`.
- Most of the mathematical operations for quantum mechanics are implemented. The most usual ones are the inverse matrix, and the determinant. If more operations are needed, they can be easily implemented.

14.1.3 Quantum states simulation: pure, mixed and entangled

- PARTIAL
- Managed by the component `Physics` depending on `Math`.
- Creating bigger states by joining simpler states is not fully implemented. This process of joining physical systems is formalized using the 'tensor product'. Only the pure states are implemented by doing the concatenation of the coefficients of input states. The tensor product of mixed and entangled states is not needed for BB84, for that reason this development is postponed.

14.1.4 Quantum states growing

- PARTIAL
- Managed by the component `Physics` and depending on `Math`.
- Is the module able to modify the internal states when performing the tensor product.

14.1.5 Physical states basis

- COMPLETE
- Managed by the component `Physics`.

14.1.6 Operators

- PARTIAL
- Managed by the component `Physics`, and depending on `Random` and `Math`.
- Only the operators for nose are implemented. For the simulation of logic gates more operators are needed.

14.1.7 Noise

- PARTIAL
- Managed by the module `Physics`, depending on `Math`.
- It would be interested to simulate several noise as described in [Pre98], and do comparison among them.

14.1.8 Conversion of physical states into information

- PARTIAL
- Managed by the component `Physics`, and depending on `Math`.
- There are methods for encoding bits and array of bits. The data type handling is generic enough for many other types. With high priority to be finalized are the *native* types. They can be quite useful to develop new algorithms, by debugging the conversion of classical into quantum information . Also encoding from the language data types must be provided.

14.1.9 Logical gates

- PARTIAL
- Only a rough design is done, almost nothing.

14.1.10 Communication channels: senders and receivers

- PARTIAL
- Managed by `Protocol` and depending on `Physics`.
- The data transmission among senders and receivers is implemented using *delegates* in C#. However this is a language specific feature, in case of porting to another language this has to be replaced by a list of callbacks.

14.1.11 Protocol logic

- PARTIAL
- BB84 is implemented using some FSM automates, but they do not fulfill the OSI specifications.

14.1.12 Classic cryptography

- COMPLETE
- Managed by `Cryptosystem`.

14.1.13 BB84 key distribution

- COMPLETE
- Managed by component `Protocol` while depending on `Data` and `Channel`.
- It could be improved with techniques as 'privacy enhancement'.

14.1.14 User interface

- PARTIAL
- The user interface would be much more friendly. However it shows all the internal configuration parameters of the framework. The debugging subsystem and the unitary test.

14.2 Nonfunctional requirements

- The distributed capabilities are not implemented, but the framework is multithread, so making it distributed should be something affordable.
- The memory management is good enough. The usage of *sparse* matrices has proofed to be satisfactory, and many of the object, such as the debuggers, are generated on demand. The basis are also developed by means of a repository, but the usage of a *flyweigth* code pattern would improve the memory usage, but it could have major impacts on the performance. The number of basis is an small set, and usually the same ones; so a repository is a good approach.
- An example of the advantage of modeling everything with interfaces is the matrix of complex numbers. Those numbers can be in the local machine or distributed among several computers. For this reason it is mandatory to isolate the data from the operations, and the quantum states from the quantum operators. The product of matrices can be done, by diving it in smaller matrices an using a distributed algorithm, and doing the partitions considering the distributed system, and where are stored each one of the numbers. This is not efficient on small matrices but on very big ones can be important.
- Also, as many random number generators as needed can be instantiated. One approach is to have one per component, another is to have one per machine, another is to have one in global. Those are performance tuning and implementation related researches. This is the purpose of this framework; to help in this kind of studies.
- The problems are in the data synchronization. The objects storing the data must be locked and unlocked for the whole system before doing modification into them. For example, a matrix multiplication should lock all the write operations into the input matrices. This implies a huge overhead for the calculus, as an easy approach, the multithreading only applies to whole states, not to individual numbers.

(This page is intentionally left blank)

Chapter 15

Possible improvements

- Enhanced graphical user interface
- Logical gates implementation
- Quantum reversibility simulations
- Stopping and resuming mathematical calculations
- Generation of quantum, classic an native data types
- Deployment as a distributed system
- Statistics about the memory usage and simulation speed
- Native code optimizations (mathematical libraries)
- Symbolic mathematical calculations
- Interaction with other simulators
- Quantum languages interpreter
- Simulation of memory registers and discrete electronics components

(This page is intentionally left blank)

Chapter 16

Comparison with other developments

16.1 PERL entanglement

It is an small class showing the behaviour of entanglement. It is very limited, and it is only useful for doing small examples. Cannot be used to simulate the evolution of those systems with operators.

It does not seems to be supported.

16.2 Libquantum++

It is a set of classes and utilities that try to simulate a quantum computer. It defines registries and a Arithmetic Logic Unit (ALU). It supports corrections codes. It is quite flexible, but it does simulated entangled states, nor communications channels. It does not emulate the measurement processes either.

It is quite coherent in its objective of simulating registries and an ALU. It also defines good interfaces for the simulation of the computation processes.

The current state is unknown.

16.3 QCL

It means Quantum Computing Language. It is a tool/simulator developed a few years ago in C/C++ for simulating algorithms such the one of Grover and Shor. It is widely cited in many research articles and books.

The program works, but from the conceptual point of view, it leaves much to be desired. The programming style does not follow any of the modern methodologies or uses any design pattern. Does not differentiate between classic data and quantum data in the simulation process, does not establish a difference between a 'quantum world' and a 'classic world'. It is only emulating results, but not doing an evolution of the system using physical concepts.

To emulate logic doors within this scenario supposes relatively little effort. It is quite more complicated to simulate a communication channel and a protocol logic that a sequence of quantum operators, that is how a circuit is modeled.

It does not have graphic interface and all the simulation is done through a command line.

The current state is unknown.

16.4 Quasi

It is quantum circuits simulator. It defines a structure of archives to save them. It also includes on a graphical interface quite attractive. It does not realise measurement simulations, it does not simulate communication channels neither ciphering.

It is currently supported.

Chapter 17

Conclusions

17.1 Considerations

1. It is a research branch **quite extensive**. Deals with many subjects, that are not allocated in any previous science branch. It mixes together information theory, noise, error correction codes,...It also includes discrete mathematics techniques (Shor algorithm) and infinitesimal calculus with data structures and storage analysis. All this without mentioning all the theoretical formalisms and practical implementations that they are discussed for each one of the concepts. All this creates a great fuss that difficulties the research objectives, a great amount is needed time to decide that article or book can be useful to the line of investigation.
2. The current **programming tools** are not able to handle quantum information. All the current developments make the assumption that unlimited copies of the data can be done. Therefore a quite complex abstraction layer is needed, so each time that the information is accessed, somehow the original system gets disturbed. This has been a challenge in this project, no other previous development or bibliographic reference makes a clear proposal for this. This is one of the main contributions of my work. I have created the basic stuff to be able to handle properly this new programming model.
3. Previous developments does not simulate states, they only track high level evolutions of the system. As far they do not simulate the states, they cannot **measure**, there are no probabilities distributions, and they are limited in features. They can only emulate the output of a set of states that were considered in their design, they are not general purpose.
4. Besides being able to handle quantum states it is necessary to be able **codify** the information. To establish a criterion to transform classic data to quantum data and reciprocally. It is something similar to codify a text in binary. It is possible to uses codes (tables of conversion of character to binary) ASCII of 7 bits, 8 bits, multitude of codes of page for each country, Unicode Transformation Format (UTF)-8,...All these are systems of codification in binary relatively standardized. But in computation quantum there are not even no proposals. There is only the 'common practice' to transform 1 qubit into 1 bit, but all the others datatypes have not been explored. There is not bibliography or researches regarding the codification of quantum data with a relatively generic approach.
5. These previous points are needed for comparing with QCL (Quantum Computing Language)

Section 16.3. It has not been possible to reuse anything of this development. As explained previously, the objectives of this work are quite more refined, form the physically simulation point of view. However at the current state this work is not able to simulate common algorithms in a easy way. The novel way of approaching the simulation (defining operators and states, define a in detail simulation of the measurement process,...)

6. Another great problem has been to find formal models for the **quantum communication channel**. I refer to a model possible to implement, with well defined emitters and receivers, enumeration the attributes of each part of the system, or how information is altered while circulating through them. I have only found a reference to channels with more of one receiver [AS97], titled 'Multiaccess Quantum Channels'.
7. **Separation of concepts**. At the time of designing some parts, such as the protocol simulation, the majority of the bibliographical references does not establish a clear difference in the entities that interacting, such as the channel, the stations, the protocol logic,...Although they are complementary concepts a better approach would be define properly the physical behavior of the channel and in top of it apply the protocol logic. This could be expressed as 'I want to send X, has arrived Y, but in fact should have arrived Z'. And referring with X, Y and Z to datatypes (bit, string, integers) rather than quantum states.

17.2 Technical

A protocol is used to communicate data. These data are codified with the codification system that settles down in states of a physical system (already preset and agreed $|\uparrow\rangle$ or $|\downarrow\rangle$, $0Volts$ or $5Volts$,...). These states are transmitted through a channel of communication and has to deal with the noise. When they are received have to be measured, this means, to compare itself with values of reference, the so-called quantum bases or electrical potential. This measurement has to be interpreted using the same the same coding criteria of the sender. And finally human-readable data is obtain

Perhaps many of these concepts and steps that I understand as necessary, for people who this used to this topics are not totally necessary. But when doing a in-detail simulation, everything needs to be well-defined; both the entities and their attributes.

17.3 Scientific

The work that I present has innovating characteristics that surpass 'state of the art' in many aspects. In certain areas there are better developments, but given it approach of framework it is highly flexible, and new features can be added and will be seamlessly integrated with the rest of the simulations.

This new conceptual way to approach the subject (to define operators, and states, to perform a simulation of the measurement process...). I believe that it is the great contribution of this work and reason why desire to give the greater possible diffusion. Sincerely I believe that it can suppose before and later in the simulation of quantum information (mainly for research, but also an unquestionable aid for the industry). Features to emulate logic quantum gates are still missing.

Part IV

Appendices

(This page is intentionally left blank)

Part V

References

(This page is intentionally left blank)

Bibliography

[AS97] A. E. Allahverdyan and D. B. Saakian. 'Multi-access channels in quantum information theory'. In: *arXiv:quant-ph/9712034v1* (1997).

[Ben73] Charles H. Bennett. 'Logical reversibility of computation'. In: *IBM Journal of Research and Development* 17 (1973), pp. 525–532.

[Bri56] Leon Brillouin. *Science and Information Theory*. Mineola, 1956. isbn: 0-486-43918-6.

[Buh00] Harry Buhrman. 'Quantum Computing and Communication Complexity'. In: *arxiv:quant-ph* (2000).

[Lan61] Rolf Landauer. 'Irreversibility and heat generation in the computing process'. In: *IBM Journal of Research and Development* 5 (1961), pp. 183–191.

[Max71] James C. Maxwell. *Theory of Heat*. Dover Publications, 1871. isbn: 0-486-41735-2.

[Moo65] Gordon E. Moore. 'Cramming more components onto integrated circuits'. In: *Electronics Magazine* 38.8 (May 1965).

[Pre98] John Preskill. *Lecture notes for physics 229: Quantum Information and Computation*. California Institute of Technology. 1998.

[Sha48] Claude E. Shannon. 'A Mathematical Theory of Communication'. In: *Bell System Technical Journal* 27 (July 1948), pp. 379–423.

[Szi29] Leo Szilard. 'On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings'. In: *Zeitschrift für Physik* 53 (1929), pp. 840–856.

(This page is intentionally left blank)

Glossary

AES	Advanced Encryption Standard
ALU	Arithmetic Logic Unit
ASCII	American Standard Code for Information Interchange
BPP	Bounded-Error Probabilistic Polynomial-Time
BQP	Bounded-Error Quantum Polynomial-Time
CORBA	Common Object Request Broker Architecture
EPR	Eintein-Podolsky-Rosen
FSM	Finite State Machine
OSI	Open Systems Interconnection
PKI	Public Key Infrastructure
QCL	Quantum Computing Language
QKD	Quantum Key Distribution
QIT	Quantum Information Technologies
QTM	Quantum Turing Machine
RMI	Remote Method Invocation
UTF	Unicode Transformation Format
VS	Visual Studio

(This page is intentionally left blank)