UNIVERSITY OF UDINE

Department of Mathematics, Computer Science and Physics

MASTER THESIS IN INFORMATICA

Quantum Finite Automata

CANDIDATE

Riccardo Romanello
Matricola 127934

SUPERVISOR

Prof.ssa Carla Piazza

Academic Year 2021-2022

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

To my niece, Ginevra.

Acknowledgements

I want to thank my family the most. It must have been an hard task to bear me during my path through this master degree. I must thank also some of my dearest friends: Brian, Christian, Federico, Michelangelo and Zelda (this is no challenge, just alphabetical order). If you hadn't been by my side at any time during these last years, I would have never been able to get where I am now. To all my other friends, please do not get mad for not being cited by name. You all are a primary building block of my life. Last, but not least, I want to thank all the professors I got to know through these years. Especially my supervisor Prof.ssa Carla Piazza.

Abstract

In this thesis we will describe theoretical foundations of Quantum Finite Automata, which emerged as the quantum counterpart of DFA. A focus will be given to automata allowed to scan the tape only in one direction (briefly called, *one way*). We will first start from an introduction on Linear Algebra and measurement theory, that we will use to define the first two models of Quantum Finite Automata: *Measure Once* and *Measure Many*.

Using these preliminary notions we will be able to study the expressive power of such classes of automata. In particular we will focus on different acceptance conditions that have been used throughout the years. After characterizing the expressive power we will be able to introduce some necessary and sufficient conditions for languages of the classic Chomsky Hierarchy to be accepted by a QFA. To conclude this discussion, we will move outside the class of one way automata considering 1.5 and 2 way automata. Using these last formalisms we will be able to introduce a complete hierarchy of expressive power of QFAs.

To conclude this thesis, as an original contribution to the state of the art, we will introduce a brand new family of automata inspired by the Heisenberg picture of Quantum Mechanics. We will study this formalism providing three different acceptance conditions. For each one of them we will try to characterize the expressive power it yields to. With the first acceptance condition we were able to prove that this new formalism is more powerful than Measure Once QFA. With the other two conditions we are able to show that at least they are as powerful as Measure Once QFA. While the other direction remains an open question.

Contents

1 Introduction

The first idea of Quantum Computer is usually related to Richard Feynman. When Quantum Mechanics was at its spike, because of its probabilistic nature, it was clear that classic computers wouldn't have been able to simulate Quantum Systems. In that environment Richard Feynman proposed to exploit Quantum Mechanics properties in order to create a faster model of computation: the quantum computer. From that very moment, a race to Quantum Computer started. The hard truth researchers faced in a really short amount of time was that QM is really counterintuitive, and they found themselves struggling with the creation of both a Quantum Hardware and Software. One of the first scientists that started working on the theoretical foundations of Quantum Computing was David Deutsch in [6]. He described the Church-Turing principle for Quantum Computing and introduced the idea of a quantum universal computer based on Quantum Turing machines. Deutsch idea was to replicate the steps made when classic theory of computation was developed, hoping to get to the same results. After that paper was published, a lot of other accomplishments were made such as the introduction of computational models like Quantum Finite Automata and Quantum Gates. Moreover, it has been proved that Quantum algorithms can obtain an exponential speed-up with respect to their classic counterpart. The most famous algorithms are Shor and Grover. The former can factorize numbers in a polynomial (quantum) number of steps. The latter is used to search an element in a database, with no preconditions on the input, with a running time of $\sqrt{N}$ where N is the size of the input as number of elements.

When talking about algorithms for Quantum Computers we explicitly refer to quantum running time. As for the Classic version of Turing Machine, also for the Quantum one a notion of running time and computational Complexity have been introduced. The most important class is BQP which consists of those problems that can be solved with bounded probability of error using a polynomial size quantum circuit. It's the quantum analog of BPP. The current known chain of inclusions is the following:

$$P \subseteq BPP \subseteq BQP \subseteq PSPACE$$

Of course, when talking about computational complexity one of the main concerns is the question 'is P equals to NP?'; actually, a similar question can be asked also for BQP: which is the relation between BQP and NP? Being able to answer such a question could help us understand the true relation between P and NP. Studying this kind of problems clearly requires going deep in the theoretical foundations of computation, just like it has been done for classic computers. Thus, it is mandatory to study lower-level formalisms, like Quantum Automata. Quantum Finite Automata are the Quantum Counterpart of the classic Deterministic Automata. Quantum Automata were first introduced by Kondacs and Watrous in [7]. A lot of other papers came out after that first one, and they followed two main approaches. The first approach tries to characterize the expressive power of Quantum Finite state Automata as they were introduced by Kondacs and Watrous. The two models used in this case are known as 'Measure Once QFA' and 'Measure Many QFA'. In the first one only one measurement is made at the end of the computation, while in the second one a measurement is made after each step. The characterization of languages accepted by MO-QFA and MM-QFA was done in [5] and [1]. From these papers, the most counterintuitive result we got is that both MO-QFA and MM-QFA accept a subset of regular languages. The second approach tries to introduce new models of Quantum Finite Automata. Unfortunately, none of these new models can overcome the regular language limitation. The only way to do so is to allow the automata a more 'Turing machine'-like behavior (i.e.: 2-Way QFA which are allowed to scan the tape in two directions). In this thesis we will then give an overview about the state of the art for Quantum Finite Automata and we will also present some original contribution with a new model of QFA.

Notice that from Chapter 4 to Chapter 9 we are going to introduce a serie of QFA models that are all based on the Shrödinger picture of Quantum Mechanics. While the brand new model we will introduce in Chapter 10 is based upon Heisenberg Picture. In Shrödinger Picture a quantum system is described by an initial state and a set of unitary matrices that are used to make the system evolve. The time dependent component in this picture is the state: in fact, the state at time $t > 0$ is different from the state at time 0. In this case the unitary matrices remain unchanged. On the other hand, Quantum Mechanics can also be described using the so called Heisenberg Picture. As in Shrödinger's we start from a state and a set of unitaries but in this case the time dependecy is moved from the state to the unitaries. Thus, the state remains fixed to its initial (i.e. time 0) value while the unitaries change through time. Physically speaking the equivalence between Shrödinger and Heisenberg picture has already been proved (in fact Copenhagen interpretation of QM usually refers to the Shrödinger picture). In a more Computer Science oriented approach, anyone has ever tried to exploit the features of Heisenberg picture while designing QFA models. This is what we aimed to do introducing QHFA.

Giving now an rapid overview on what we are going to discuss in this thesis,

We start from **Chapter 2** in which we will introduce and describe a series of mathematical tools. We will need them throughout the whole thesis to formalize the description of the topics.

In **Chapter 3** we will then move to an higher abstraction level. We will give a shallow introduction to Quantum Mechanics in a physical way. Then we will move to the description of Quantum Computing in its most basic components. In the final part we will also give a proof of the differences between probabilistic and quantum computing using the *Quantum Interference* effect.

In **Chapter 4** we will finally introduce two different version of Quantun Automata. It will then lead us to the definition of Measure Once and Measure Many QFA. Historically speaking, the two models were derived indipendently but roughly at the same time.

In **Chapter 5** we will introduce some notions of acceptance for Quantum Automata. Moreover, we will state and prove some theorems useful to create a link between diffent acceptance conditions.

In **Chapter 6** we will finally move into the description of the expressive power of Quantum Finite Automata. We will do it for both Measure Once and Measure Many QFA.

In **Chapter 7** we will introduce the 1-Way *General* QFA. This formalism was introduced with the goal of overcoming the expressive power of Measure Many and Measure Once QFA.
In **Chapter 8** we will the introduce other two formalisms for QFA. The reason for this chapter is to show to the reader in which way the researchers tried to overcome the limitation introduced by MO and MM QFA.
In **Chapter 9** we will move in the description of QFA that are allowed to move also right or even stay put in the input tape. After this descriptuion we will be able to introduce a complete expressiveness hierarchy for QFAs.
We will then use **Chapter 10** to introduce and describe the original contribution of this thesis about QFA. In particular, a brand new model of Quantum Finite Automata will be introduced. The main idea it exploits is the Heisenberg Picture of Quantum Mechanics. For this new model we will prove some basic results and we will of course give some unanswered questions that may be answered in the future.

2

Prerequisites

In this section we will introduce some Prerequisites. They will be necessary to understand all the operations we will use during the description of the various formalisms. We will start by giving some basic definitions about the mathematics we are going to need throughout the whole thesis. Then we will introduce the main lemmas about Quantum Mechanics. In the end we will switch to some more advanced mathematical topics like quantum measurament.

2.1 Complex Numbers

A complex number is any number of the form:

$$a + ib \tag{2.1}$$

Where $a, b \in \mathbb{R}$ are known as the *Real part* and the *Imaginary part* respectively. i is the imaginary unit such that $i^2 = -1$. The set of all complex numbers is called $\mathbb{C}$.

Given a complex number $w = a + ib$ we define the *norm* of w as:

$$|w| = \sqrt{a^2 + b^2}$$

while the *complex conjugate* of w is:

$$w^* = a - ib$$

A generic complex number $z = a + ib$ can also be described using polar coordinates. If in fact we see z as a vector on a complex plane, then the length of the vector is its own norm ($r = |z|$) and it forms with the real axis an angle ϕ. Using classic trigonometric rules, we can see that $a = r\cos\phi$ while $b = r\sin\phi$, thus $z = r\cos\phi + ir\sin\phi$. Rewriting the terms and using Euler formula $e^{i\theta} = \cos\theta + i\sin\theta$, z can be written as:

$$z = re^{i\phi} \tag{2.2}$$

2.2 Hilbert space

Definition 2.2.1 (Hilbert Space)**.** *Let V denote a Vector Space over $\mathbb{C}$. V is called Inner Product space if it has an Inner product defined on it. Namely, an inner product is a function $(\cdot,\cdot) : V \times V \to \mathbb{C}$ that satisfies the following:*

- $(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z) \ \ \forall x, y, z \in V, \alpha, \beta \in \mathbb{C}$
- $(x, x) \geq 0 \ \ \forall x \in V \setminus \{0\}$ *Moreover, it's true that:* $(x, x) = 0 \leftrightarrow x = 0$
- $(x, y) = (y, x)^* \ \ \forall x, y \in V$

*An **Hilbert Space** is an Inner Product space that is complete under the norm induced by $\|\cdot\|$. With complete under the norm we mean that all the Cauchy sequences of vectors in V focus to a limit in V : this proprerty is meaningfull with infinite-dimension spaces, because with finite-dimension spaces it is always true. In quantum computing, all the vector spaces have a finite number of dimensions: so, for our purposes, the term **Hilbert Space** is equal to Inner product Space.*

2.3 The $\mathbb{C}^2$ space

The space in which the computational basis is described is $\mathbb{C}^2$. The generic element $v \in \mathbb{C}^2$ has the form:

$$v = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

with $\alpha, \beta \in \mathbb{C}$.
Given a vector $v \in \mathbb{C}^2$ as just defined we will refer to its *norm* has:

$$\|v\| = \sqrt{|a|^2 + |b|^2}$$

and to its *complex conjugate* with the row-vector:

$$v^\dagger = (\alpha^*, \beta^*)$$

Definition 2.3.1. *A vector $v \in \mathbb{C}^2$ is said to be a unit vector if $\|v\| = 1$. If this condition holds, then is also true that v is normalized.*

If a vector v is *non-normalized* then the operation of dividing it by its norm is called *normalization*, namely

$$v = \tfrac{v}{\|v\|}$$

Given two vectors $v = \begin{pmatrix} \alpha_v, \\ \beta_v \end{pmatrix}, w = \begin{pmatrix} \alpha_w, \\ \beta_w \end{pmatrix} \in \mathbb{C}^2$ we define their inner product as:

$$(v, w) = v^\dagger w = (\alpha_v^*, \beta_v^*) \begin{pmatrix} \alpha_w \\ \beta_w \end{pmatrix} = \alpha_v^* \alpha_w + \beta_v^* \beta_w \tag{2.3}$$

The space $\mathbb{C}^2$ together with the inner product as defined above is an Hilbert Space.

2.3.1 Dirac Notation

To describe a generic element inside $\mathbb{C}^n$ (notice that all the definitions we gave for $\mathbb{C}^2$ are easily generalized for a generic $n \geq 3$) we will use the dirac notation:

- The general column vector $v \in \mathbb{C}^n$ is written as $|v\rangle$ (known as *ket*)
- Given $v \in \mathbb{C}^n$ then $v^\dagger$ is written as $\langle v|$ (known as *bra*)
- The inner product as defined in (2.3) between two vectors $v, w \in \mathbb{C}^n$ is written as $\langle v\,|w\rangle$ (*bra-ket* notation)
- If U is a linear operator on $\mathbb{C}^n$ then to apply U to a generic vector $v \in \mathbb{C}^n$ we write $U\,|v\rangle$
- If U is a linear operator on $\mathbb{C}^n$ and $|\psi\rangle\,, |\varphi\rangle \in \mathbb{C}^n$ then with $\langle\varphi|\,U\,|\psi\rangle$ is the inner product between $|\varphi\rangle$ and $U\,|\psi\rangle$

Using this notation we can now introduce the *orhtogonality* definition.

Definition 2.3.2. *$|\psi\rangle\,, |\varphi\rangle$ are ortoghonal if their inner product is 0.*

Moreover, the *norm* of a vector $|\psi\rangle$ can be rewritten as:

$$\|\,|\psi\rangle\,\| = \sqrt{\langle\psi|\psi\rangle}$$

We can also define that a set $\{|1\rangle\,, |2\rangle\,, \cdots |n\rangle\}$ of n vectors is said to be *orthonormal* if each vector is a unit vector vector and distinct vectors in the set are orthogonal that is

$$\langle i|j\rangle = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}$$

2.4 Complex matrices

We refer to a complex matrix as a generic element of the vector space $\mathbb{C}^{n\times n}$ (with $n > 1$). Given $A \in \mathbb{C}^{n\times n}$ we will use $A^\dagger$ to refer to its *Conjugate transpose*: $A^\dagger$ is obtained from A taking the transpose of A and then changing every complex element with its own complex conjugate.

Definition 2.4.1. *A complex matrix $A \in \mathbb{C}^{n\times n}$ is said to be an **Hermitian matrix**(or self-adjoint) if it's equal to its own conjugate transpose; namely, $A = A^\dagger$*

Notice that, on an Hermitian complex matrix the elements on the diagonal must be real (the only way for a complex number to be equals to its own complex conjugate is to have a 0 imaginary part).

Definition 2.4.2. *An Hermitian Matrix $A \in \mathbb{C}^{n\times n}$ is **positive semi-definite** if it's true that:*

$$x^* A x \geq 0 \;\; \forall x \in \mathbb{C}^n$$

Definition 2.4.3. *A complex matrix $A \in \mathbb{C}^{n\times n}$ is said to be **unitary** if $A^\dagger$ is also equal to the inverse of A. Namely,*

$$AA^\dagger = A^\dagger A = I$$

Definition 2.4.4. *Suppose W is a k-dimensional vector subspace of a d-dimensional space V. Then we can construct an orthonormal basis $|1\rangle, |2\rangle, \dots |d\rangle$ for V such that $|1\rangle, |2\rangle, \dots |k\rangle$ is an orthonormal basis for W. By definition, the operator P defined as:*

$$P = \sum_k |i\rangle \langle i|$$

is the projector onto the subspace W.

From the definition we gave above, it can be shown that P is Hermitian and moreover it holds that P is idempotent, namely $P^2 = P$

2.5 Projective measurement

To extract some information from a quantum state $|\psi\rangle$ a measurement has to be performed: the operation we will use for this kind of operation is the projective measurament. We will consider here only projective measurements that are defined by a set of so called *projective matrices* $\{P_m\}$ where the index m is the potential outcome we could get from the measurament. Not all set of matrices are allowed as projective measurament, but $\{P_m\}$ has to respect the constraint that:

$$P_i P_j = \begin{cases} P_i \; i = j \\ 0 \; i \neq j \end{cases}$$

and moreover it must hold that:

$$\sum_{i=1}^{n} P_i = I$$

where I stands for the identity matrix.

When a state $|\psi\rangle$ is measured using a projective measurement $\{P_m\}$ then the probability of getting m as an outcome is described by the equation:

$$p(m) = \langle\psi| P_m |\psi\rangle = \|P_m |\psi\rangle\|^2$$

and the state then consequentially collapses into a new state $|\psi'\rangle$:

$$|\psi'\rangle = \frac{P_m |\psi\rangle}{\sqrt{p(m)}}$$

Usually a projective measurement $\{P_m\}$ is described using an *observable* M, an Hermitian matrix that has a decomposition:

$$M = \sum_m m P_m$$

where m are different eigenvalues and P_m is the projector onto the eigenspace of M with eigenvalue m.

2.6 Recall of classic automata

Definition 2.6.1 (Reversible Finite Automata)**.** *A 1-Way reversible automata (RFA) is a DFA* $M = (Q, \Sigma, \delta, q_0, F)$ *in which, for every pair* $(q, a) \in Q \times \Sigma$ *there is at most one* $q' \in Q$ *such that* $\sigma(q', a) = q$ *(or, if there exists* q_1, q_2 *different states such that* $\delta(q_1, a) = \delta(q_2, a) = q$*, then it's true that* $\bigcup_{a \in \Sigma} \delta(q, a) = \{q\}$*. This state is also called a spin state cause once the automata reaches it, it never leaves it).*

Definition 2.6.2 (Probabilistic Reversible Finite Automata)**.** *A PRFA is a probabilistic finite automata such that for any* q_1 *and any* $\alpha \in A$ *there is at most one* q_2 *such that the probability of passing from* q_2 *to* q_1 *after reading* α *is non zero.*

We can then link che class of language accepted by RFAs with the one accepted by PRFAs using the following theorem:

Theorem 1. *If a language* L *is accepted by a RFA* M *then there exists a PRFA* M' *which accepts* L

Proof. The proof is trivial because M' is exactly M where all the edges have probability 1. □

3

Quantum Computing

3.1 Quantum Mechanics

Quantum Mechanics (QM for short) is the field of physics that studies the behaviour of sub-atomic particles. It is, togheter with General Relativity, the main block of modern physics. The father of QM is Max Planck that in the 1900 firstly theorized the idea of *quanta*: the smallest 'packet' a light radation can be splitted in. This theory was then formalized by Einstein in the 1905: from that point on, QM was officially born. The main contributors to the developement of the theory are physicists like Erwin Schrödinger, Werner Karl Heisenberg, Niels Bohr and so on. The first counter-intuitive phenomena on which the QM (the subatomic realm where the quantum mechanics rules act) relies on is the *wave-particle duality*: this principle states that photons or other subatomic particle does not behave only as particles but also as waves; the experiment that helped proving this statement is also known as the *double slit experiment* where a photon showed property that were clearly associated to waves (when particle-like behaviours were expected).

How to describe the result of this experiment? The most common interpretation (it's neither the only one nor the best one, just the most widely used) for QM is known as *Copenhagen interpretation* and it describes the subatomic realm using just a few definitions:

- A wave function which is commonly named Ψ represents the entire state of the system (so there are no hidden variables that we are, for an human error, not taking into account). It encapsulates everything that can be known about that system before an observation; the evolution during time of this wave function is actually deterministic, and it's described using the Schrödinger equation (if we suppose to take as basic 'picture' for quantum mechanics the Schrödinger one).

- The properties of the system follow a principle of incompatibility, defined through the Heisenberg uncertarnty principle; it states for example, that more precisely we measure the position of a particle, less precisely we can measure its momentum (the pairs of properties that are constrainted under the uncertanty principle are known as Complementary)

- When the system is measured, the wave function of the system is said to collapse, or irreversibly reduce to an eigenstate of the observable that is seen.

- Even if the system is described using a wave function and the uncertanty principle does not allow us to measure all the properties with the same precision, it is widely accepted that after a measurement is made the result is totally a 'classic' result (for the position we will have some coordinates, for the speed a value that has correct measurement unit and so on).
- The description given by the wave function is probabilistic, and so, nature is probabilistic (from here the famous quote from Einstein *God does not play dice*). The probability of an event (using the description given by a wave function) is the square of the absolute value of the amplitude of its wave function.
- The wave function expresses and prove the wave–particle duality.
- When quantum numbers are large, they refer to properties which closely match those of the classical description.

The first and the third point in this list are the most useful ones when understanding quantum computing; using a mathematical description, the first point can be described as:

Definition 3.1.1. *Associated to any isolated physical system is a complex vector space with an inner product, namely an Hilbert Space, known as the state space of the system. The current state of the system is completely described by its state vector which is a unit vector in its state space.*

Moreover, the evolution of a closed system is described using an unitary transformation. That is, the state $|\varphi\rangle$ of a system at time t_1 is related to the state $|\varphi'\rangle$ at time t_2 by an unitary operator U that only depends on t_1, t_2 namely:

$$|\varphi'\rangle = U(t_1, t_2)\,|\varphi\rangle$$

Obviously, during our disssertation we will not use any description related to the well accepted notion of time: we can, anyway, describe the process of reading two consecutive chars in a string as two consecutive processes happening in two consecutive moments in time. Therefore, we can create a biijection between a generic string $\sigma = \sigma_1\sigma_2\dots\sigma_n$ (namely, the input of our Finite automata) and the consecutive moments in time $t_1, t_2, \dots t_n$ where t_i, t_{i+1} are consecutive moments in time $\forall i = 1, 2 \dots n-1$ and at the moment t_i only the char σ_i is eventually provided as input to the Quantum Finite Automata. In this way the equation we introduced above to describe the evolution in time of a state vector can be described using the notion of read character, where during the computation for an n-length string σ, given two moments in time t_i, t_j with $i \in \{1, 2, \dots n-1\}$ and $j \in \{i+1, i+2, \dots n\}$ we can state the system evolves according to the rule:

$$|\varphi'\rangle = U(t_i, t_j)\,|\varphi\rangle = U(\sigma_i, \sigma_{i+1} \dots \sigma_j)\,|\varphi\rangle$$

Relating U to σ and not anymore to the notion of time.

The other concept that is useful in either Quantum Mechanics and Quantum Computing is the measurement.

Definition 3.1.2. *Quantum measurements are described by a collection $\{M_m\}$ of measurement operators. These are operators acting on the state space of the system being measured. The index m refers* *to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement then the probability that result m occurs is given by:*

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle$$

and the state after the measurement is:

$$\frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}}$$

The measurement operator also has to satisfy the relation:

$$\sum_m M_m^\dagger M_m = I$$

Notice that if to the last definition we add a constraint in which we require the M_m to be orthogonal projectors, that is, the M_m are Hermitian and $M_m M_m' = \delta_{m,m'} M_m$, then this definition of measurement is exactly the definition of *projective measurement* we already introduced. (the operator $\delta_{m,m'}$ is known as Kronecker delta and it is 1 when $m = m'$, 0 otherwise).

From this definition of measurement we can see that not only the 'statistics' is given, but we also have a definition of how the state changes after the measurement; in many cases, as we will also see in this disseration, the *post* measurement state is not important beacuse the measurement is only made once. For this kind of exepriments, the measurement formalism used is neither the general one we just introduced nor the projective measurement, but is the *POVM* measurement. In this formalism, given a collection $\{M_m\}$, for each m it is defined an operator E_m as:

$$E_m = M_m^\dagger M_m$$

It's pretty straightforward to see that $\sum_m E_m = I$ and that $p(m) = \langle\psi| E_m |\psi\rangle$ for a given state $|\psi\rangle$. Thus the colletion $\{E_m\}$ is enough to compute the outcomes probabilities: the operators E_m are known as POVM *elements* while the entire collection $\{E_m\}$ is known as the POVM. Notice that if we define a POVM starting from a projective measurament, we will obtain that $E_m = P_m, \forall m$

As we already introduced, Einstein did not agree with QM. Moreover after the introduction of the so-called *Quantum Entaglement*: it's a phenomena that allows two or more particles to be put in a state known as *entangled*, where the particle are someway *bonded* together. Until now, it looks something normal but, if we split an entalged pair of particles keeping one here on the planet earth and sending the other in the deep space (for example close to Betelgeuse) a *spooky* action pops up: if we measure the particle we kept here on earth, the effect of the measurement is instantly seen also in the other particle that is lightyears away from here! Quantum Entanglement is a really useful effect in quantum computing, for example for the teleportation circuit.

3.2 Handbook for Quantum Computing

The bit is the fundamental concept of classical computation and classical information. Quantum computation is built upon an analogous concept, the quantum bit, or qubit for short.

The main difference between Quantum and classic computing is that the bit can be either in the state 0 or in 1 while the *Qubit*, exploiting the laws of the quantum realm, namely *Quantum Superposition*, can be both 0 and 1 together until it is measured.

So, while a bit can be described using only one digit, 0 or 1, how do we describe a qubit that, how we just said, can be 0 and 1 but can also be in a superposition state?

To answer this question we will use the space $\mathbb{C}^2$.

The quantum counterpart of the bit 0 and 1 are the two column vector $|0\rangle, |1\rangle \in \mathbb{C}^2$ defined as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The states $|0\rangle, |1\rangle$ are also known as *computational basis states* and they form an orthogonal base for $\mathbb{C}^2$.

A generic quantum bit $|\psi\rangle$ is a *superposition* of $|0\rangle, |1\rangle$, defined as:

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

where $\alpha, \beta \in \mathbb{C}$ are called the *amplitude* of $|0\rangle$ and $|1\rangle$. What's the meaning of those two complex numbers? They actually *hide* the probabilistic nature of quantum computing: in fact, given a *qubit* $|\psi\rangle$ as defined above, when it is measured we have a probability $|\alpha|^2$ to see 0 as a result and $|\beta|^2$ to see 1 (notice how after introducing the concept of measuring we dropped the Dirac notation and actually returned to classic 0 and 1). From that, it's easy to see that both α and β cannot get any casual values but they must satisfy the following:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3.2}$$

that can be described, geometrically, as the condition that the qubit's state be normalized to length 1. (A qubit state is a unit vector in a two-dimensional complex vector space)

The first thing implied by the constraint on α, β is that any operator applied on a *qubit* must preserve the constraint (3.2): to obtain that the main way is to use always *unitary* operators (operators which matrix is unitary) that also allow us to have reversible computations.

A natural question arising from the description we just gave of a qubit state is: if a bit can only be in two states, how many state can assume a generic qubit described as in (3.1)? Let's take a generic *qubit* $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$; as we introduced in (2.2), we know that $\alpha, \beta \in \mathbb{C}$ can be wrote using polar coordinates:

$$\alpha = r_\alpha e^{i\phi_\alpha}$$

$$\beta = r_\beta e^{i\phi_\beta}$$

Thus $|\psi\rangle$ becomes:

$$r_\alpha e^{i\phi_\alpha} |0\rangle + r_\beta e^{i\phi_\beta} |1\rangle \tag{3.3}$$

If we now apply the constraint (3.2) to the description we just introduced of $|\psi\rangle$ we obtain:

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

We will now show how we derived this formula from (3.2). Let $\alpha = a + ib$ (β is pretty the same):

$$\begin{aligned}
|\alpha| &= \sqrt{a^2 + b^2} \text{ Using polar coordinates} \\
&= \sqrt{r_\alpha^2 \cos^2 \phi_\alpha + r_\alpha^2 \sin^2 \phi_\alpha} \\
&= \sqrt{r_\alpha^2 \Big(\underbrace{\cos^2 \phi_\alpha + \sin^2 \phi_\alpha}_{=1} \Big)} \\
&= \sqrt{r_\alpha^2} \\
&= r_\alpha
\end{aligned}$$

We showed that the norm of α is equals to r_α, thus $|\alpha|^2 = r_\alpha^2$.

Moreover, the equation in (3.4) actually describes all the points in the 1-radius circle for $\mathbb{R}^2$ and we can then describe r_α, r_β using a generic angle ρ:

$$r_\alpha = \cos \rho$$
$$r_\beta = \sin \rho$$

Thus, setting $\rho = \theta/2$ we can rewrite (3.3) as:

$$|\psi\rangle = \cos(\theta/2) e^{i\phi_\alpha} |0\rangle + \sin(\theta/2) e^{i\phi_\beta} |1\rangle$$

We can now also gather the terms of the kind $e^{i\phi}$ and setting $\varphi = \phi_\beta - \phi_\alpha$, $\gamma = \phi_\alpha$ we obtain:

$$|\psi\rangle = e^{i\gamma} \big(\cos(\theta/2) |0\rangle + e^{i\varphi} \sin(\theta/2) |1\rangle \big)$$

From a physical point of view, the term $e^{i\gamma}$ is called *global phase* and it's known to have no visible effects on the measurement of a qubit (measuring $|\psi\rangle$ and measuring $e^{i\gamma} |\psi\rangle$ is exactly the same). So, there exists a one to one corrispondence between the description of a qubit $|\psi\rangle$:

$$\cos(\theta/2) |0\rangle + e^{i\varphi} \sin(\theta/2) |1\rangle$$

and the points on a particular sphere known as *Bloch Sphere*. Therefore, it looks like we can encode an infinite number of different combinations for the angle θ: that's true but, from the rules of quantum mechanics we know that the only way to obtain the informations from a superposition is through the measuration process which outcome is always one single classic bit that can be either 0 or 1! So it's crucial to remember that even if we can have an infinite number of possibile superpostion for a qubit, once it is measured the result is always one single classic bit.

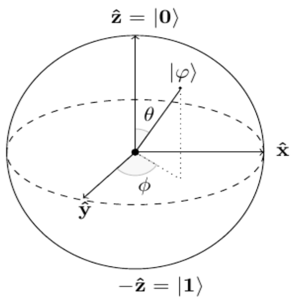

$\hat{\mathbf{z}} = |\mathbf{0}\rangle$
$|\varphi\rangle$
θ
$\hat{\mathbf{x}}$
ϕ
$\hat{\mathbf{y}}$
$-\hat{\mathbf{z}} = |\mathbf{1}\rangle$

Figure 3.1: The Bloch Sphere

The generic quantum state we will refer to during this paper can be described as:

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$$

where it must hold that:

$$\sum_{i=1}^{n} |\alpha_i|^2 = 1$$

where each $|\alpha_i|^2$ is the probability that when $|\psi\rangle$ is measured we get $|q_i\rangle$ as result.

3.3 Quantum and probabilistic computing

Before defining the differences between a classic, a Probabilistic and a quantum algorithm, let's see what is an algorithm. There are plenty of definitions, but the one we will use is:

An algorithm is a finite serie of instructions that, given some data in input, it transforms them in a output result

So, an algorithm is just like a function that, starting from a domain (input data), gives us a value inside a codomain (output data). The easiest algorithm is the one that creates the identity function: given a number $x \in \{0,1\}$ it gives us the value of x as result.

$$f : \{0,1\} \to \{0,1\}$$

We can think that any function can be computed with algorithm: actually, it has been proved in 1936 that not all the functions that we can think can be computed with an algorithm: for example, there is no algorithm that takes in input one other algorithm and tells us if the algorithm ends or no. We'll see that the set of the *computable* functions has well known bounds.

But, how many types of algorithm can we write? Before Quantum Computing there were two types:

1. **Deterministic algorithms**: the algorithm has one 'way' to solve a problem and the solution is correct in 100% of cases. A classic algorithm follows a finite sequence of instructions that gives us the result. If we describe every instruction of the algorithm as a vertex, the execution of classic

algorithm could be plotted as a one-way connected list of vertex: every vertex has one incoming edge (except for the Input vertex) and one outcoming edge (except for the Output Vertex)

2. **Probabilistic algorithms**: the algorithm has many paths it can follow, and each possible path has a probability to be followed. The result is not always right, we will see when a Probabilistic algorithm is defined as 'good' and when no. As before, let's imagine every instruction of the algorithm as a vertex but in this case, every vertex is not connected to just one another vertex, but it can be connected to many. So, each vertex can have many incoming edges (except for the input vertex) and can have many outcoming edges (except for the output vertex). Each vertex has a weight: if the vertex e goes from u to v, then the weight of e is the probability that, once the algorithm has executed the instruction described by node u, it executes, as next operation the instruction decribed by node v. And so, referring to a classic mathematical structure, the execution of a Probabilistic algorithm can be seen as a weighted graph.

When talking about a classic algorithm we know that, if we reach a result, it is correct in the 100% of cases, in the case of a Probabilistic algorithm the probability that we get the right result is given by probaility theory.

Let's take a generic Probabilistic algorithm, let's call it A. This algorithm has n different paths that it can take, but just m of them give us a correct result. Each of these paths is made by some edges, where each edge has a weight
We can know extend the definition of weigth: we will call *weight of a path* the product of the weights of all the edges that form the path (this is just the probability of following the entire path edge by edge). Given a path P, we will refer to its weight as $W(P)$.
So, we know that, in the A algorithm we have m paths that, during the execution can lead us to a correct output. What is the probability that we *actually* get a correct output? Let's enumerate all the m paths as $P_1, P_2, ..., P_m$, then the probability that the algorithm A gives us a correct answer is given by this formula:

$$\sum_{i=1}^{m} W(P_i) \tag{3.5}$$

It may look trivial, but, if at least one of the m path has a non-zero weight, then we get a non-zero probability of having the right output. Obviosly, before things gets creepy, it's important to say that:

In a probabilistic algorithm, there are many paths that can be followed, but during the execution just one of them is followed

Now that we introduced classic and probabilistic algorithm, let's see which changes quantum computing has brought to this matter. The execution of a Quantum algorithm can be seen (graphically) as the execution of a probabilistic one (in what concern probabilities, edges and vertex). But, there are some differences:

1. the weight of each edge is not a probability but it's a *probability amplitude* and, as we saw before, probability amplitudes are complex numbers. So, the function that associates to each edge a weight has a new codomine: $\mathbb{C}$.
   We will see that this change of codomine will give us an unexpected result because of minus sign that products between complex numbers generate.

2. while the *weight of a path* is always calculated as a product (in this case of probability amplitudes instead of simple probabilities) but the probability that we got with (3.5) changes its formula (we use the same hypotesis as for (3.5)) and becomes:

$$\left|\sum_{i=1}^{m} W(P_i)\right|^2 \tag{3.6}$$

3. Last (but not least) difference is that, whereas in a probabilistic algorithm, between all the paths only one is actually followed, in a quantum algorithm (because of quantum mechanincs rules) all the paths are followed and together executed: just in the end, when we watch the result we make the system collapse in one single result, but until we don't measure it all the possible results exist together!

.

Maybe, those three differences, taken one by one, could make you think that, beside some maths, there is no difference between a probabilistic and a quantum algorithm: but take a look to this 3 togheter. The consequence is a strange phenomena called 'quantum interference': while in a probabilistic algorithm if even only one of the probabilities P_i in (3.5) is non-zero then we have a non-zero probability to get the right output, in a quantum algorithm that's not true! and the reason is that sometimes the probabilities in (3.6) cancel out each other. This is due to the fact that for quantum algorithm we do not talk about simple probabilities, but we are using complex numbers and, multiply complex numbers generate minus signs that make separated paths to intefere with others. So, we can get a quantum algorithm where we have two paths (P_1, P_2) with non-zero probability to get to the output $(W(P_1), W(P_2) > 0)$, but (3.6) gives us a 0 result. This, because we have

$$W(P_1) = c$$

And

$$W(P_2) = -c$$

and if we put $W(P_1), W(P_2)$ in (3.6) we get

$$\begin{aligned}\left|\sum_{i=1}^{m} W(P_i)\right|^2 &= \text{with } m = 2\\ &= |W(P_1) + W(P_2)|^2\\ &= |c + (-c)|^2\\ &= |0|^2\\ &= 0\end{aligned}$$

Let's give an example to better understand this critical concept. Looking at picture (3.5), we can see a simple quantum machine with its *probability amplitudes*. We will use this notation:

c_{ij} is the probability amplitude to get to output j with input i.

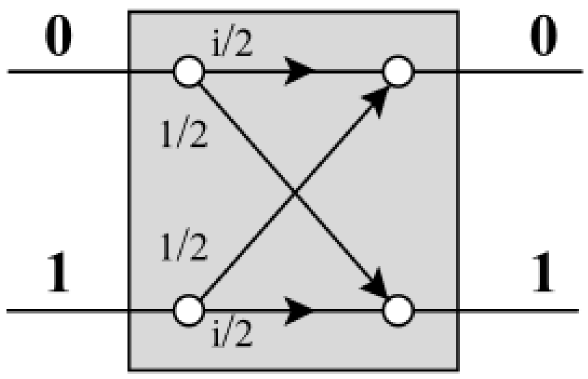


Figure 3.2: A simple Quantum machine decribed as a graph

In this case we have:

$$c_{00} = i/2, c_{01} = 1/2, c_{10} = 1/2, c_{11} = i/2$$

In this particular case, the machine at picture 4.1 is also known as *Random bit* because given in input a bit, it gives us in output a bit that is 0 or 1 with the same probability, so it's an example of true randomness in the world of Computer Science (true randomness cannot be achieved in classic computation).

So, let's imagine to put two copies of this machine in serie (connecting the output of the first as input for the second). Now, we want to compute the probability of getting a 0 output giving 0 as input in this new machine. If we look at it, we have two paths that can lead us to the wanted result:

1. $0 \to 0 \to 0$, we will call it P_1

2. $0 \to 1 \to 0$, we will call it P_2

So, the m we are gonna use for (4.2) is 2 (becase of two good paths).
Let's calculate P_1 and P_2.
P_1 is made of two edges: the one that brings from 0 to 0, and one other from 0 to 0, so, it's *weight* is

$$W(P_1) = c_{00} \cdot c_{00} = i/2 \cdot i/2 = -1/4$$

P_2 is made of two edges too: the one that brings from 0 to 1, and then the one that brings from 1 to 0

$$W(P_2) = c_{01} \cdot c_{10} = 1/2 \cdot 1/2 = 1/4$$

Now, applying (4.2) we get that, the probability of going from 0 input to 0 output is

$$\begin{aligned}\left|\sum_{i=1}^{m} W(P_i)\right|^2 &= \text{with } m = 2 \\ &= |W(P_1) + W(P_2)|^2 \\ &= |-1/4 + 1/4|^2 \\ &= |0|^2 \\ &= 0\end{aligned}$$

And, as we wanted to show, there are case in which, non-zero weights for single paths bring us to zero probability after applying (4.2).

4

Quantum Finite Automata

4.1 Simulating a Quantum System

The first physicist who sayd that a Quantum Computer was able to simulate things that a normal computer could not was Richard Feynmann. During one of his lecture, Feynmann described how physicists use computers to do simulations: sometimes they want to do tests before trying the experiment for real, or maybe they don't have access to the system so they fake it using computers simulation. But, can a normal computer simulate every physical system? Feynmann supposed that the amount of memory and computing capability may be enough to simulate planets, or the motion of galaxies: but what about quantum systems? When talking about quantum realm, the main feature is the *Superposition Principle*: for example, a quantum system of electrons or protons can exist in a superpostion of observable states before the measure. So we must keep track of all the probabilities: usually, with a classical computers, to keep track of n particles in a quantum system we need an amount of memory that is in the order of 2^n and these number become really big in a really short time. The solution Feynmann gave was that, if the problem is to simulate quantum systems, why don't we use them as computing power? Qubits can keep track of the probabilities better than normal bits, because it's their nature to behave like quantum particles. So, if we suppose that each Qubit keep track of the probability of one quantum particle in the quantum system we described before, then we should need just n Qubit (one per particle). So, theorical physicists and computer scientists started to work on a model of Finite State Automata but based on Quantum Mechanics: *Quantum Finite State Automata*

4.1.1 Quantum Automata

The first, and most generic definition of Quantum Automata (notice that we are not assuming it to be Finite) was given in the 1997 in [10]. In this paper, a *Quantum Automata* is defined as *Real Time Quantum Automaton*
$Q = (H, s_{\text{init}}, H_{\text{accept}}, P_{\text{accept}}, A, U)$, where:

- H is a Hilbert Space
- s_{init} is the initial state vector (obviously $s_{\text{init}} \in H$) and $\|s_{\text{init}}\|^2 = 1$
- H_{accept} is a *subspace* of H

- P_{accept} is an operator projecting on H_{accept}
- A is the input alphabet
- $U = \{U_a \mid a \in A\}$ is a set of unitary operators, namely one for each symbol in A. Every $U_a \in U$ is called transition matrix.

Given a word $w = w_1 w_2 \dots w_n \in A^*$, we define U_w as:

$$U_w = U_{w_n} \dots U_{w_2} U_{w_1}, \; U_{w_i} \in U \quad \forall i = 1, 2, \dots n \tag{4.1}$$

This is the unitary transformation applied to s_{init} while reading the input w. The language accepted from a Quantum Automata Q is then defined as the function:

$$p_Q(w) = \|P_{\text{accept}} U_w s_{\text{init}}\|^2 = \|P_{\text{accept}} |s_w\rangle\|^2 \tag{4.2}$$

if we define $|s_w\rangle = U_w s_{\text{init}}$ The formula we just introduced can be described as: the computation starts with the state $|s_{\text{init}}\rangle$, then, while the input w is read, the Unitary Operators associated with the symbols are applied to the current state and in the, to compute the probability that the state obtained (that can, and pretty always be, a Superposition) is in H_{accept} we use the projection operator P_{accept} and then measure the norm.

If we look at the definition given in [10], we see that the definitions for (4.1) and (4.2) are different: the differences come from how (4.1) is defined. In [10], in fact, U_w is defined as the product $U_{w_1} U_{w_2} \dots U_{w_n}$; that kind of structure is used to make it look similar to the actual process of input reading from left to right. This help that is given to the reader has an impact on (4.2): the state used in the begginig is not $|s_{\text{init}}\rangle$, but is actually $\langle s_{\text{init}}|$: no matter of U_w is defined, the order of the operator applied has always to follow the order of the input symbols! (First has to be applied the operator for w_1, then the one for w_2 and so on).

After this little discussion, we can further analyze (4.2): the first main difference that has to be seen between a language accepted from a classic Automata and a Quantum One is its probabilistic Nature. In fact, while for a classic automata we know exactly if a word is either accepted or not (it's a true / false decisione), here we cannot talk absolutely but we have to talk in matter of probabilities due to the probabilistic nature of Quantum Mechanics.

The second main topic to discuss about (4.2) is: what does applying U_w to $|s_{\text{init}}\rangle$ do? The effect of the matrix product $U_w = U_{w_n} \dots U_{w_2} U_{w_1}$ (thinking about how matrix product is defined) is to sum over all possible path that the automata can take given the input word w (think about superposition: every possible path must have a probability to be taken). Each of this paths, as we saw in the section about differences between probabilistic and quantum computing, has a *complex amplitude* that is the product of the amplitudes at each step of computation.

We will use the notation $U_a^{s_i,s_j}$ to define the component of U_a that describes the probability amplitude of moving to state s_j while being in state s_i and the current input symbol is a. (Can be seen as the probability used to label edges in markov chains).

Given U_w for a generic word w, the initial state s_0 and a final state s_w we can state that:

$$U_w^{s_0,s_w} = \sum_{s_1,s_2,\dots,s_{|w|-1}} U_{w_1}^{s_0,s_1} U_{w_2}^{s_1,s_2} \dots U_{w_{|w|}}^{s_{|w|-1},s_{|w|}} \tag{4.3}$$

By giving a look a the formula above, we can see that the component for the pair s_0, s_w in U_w is actually the sum over all possible choices for $s_1, s_2, \dots, s_{|w|-1}$ (obviously given from the sum operator) of the amplitude of a chosen path: in fact, if we look at the body of the sum operator, we see that once a value for $s_1, s_2, \dots, s_{|w|-1}$ is chosen (namely $\hat{s}_1, \hat{s}_2, \dots \hat{s}_{|w|-1}$) then the body is actually the amplitude of the path $\hat{s}_1 \to \hat{s}_2 \to \dots \hat{s}_{|w|-1}$. What can happen in the computation of U_w is the *Quantum Interference* phenomena: in fact, as we saw in the section about quantum and probabilistic computations, the paths in a quantum computation can cancel out each other in the sum over all the possible paths.

It's pretty difficult to find resemblance from the definition we just gave and the automata we are used to see: in fact, the finite-state machine we always studied are, as in the name, finite. The quantum analog of finite-state machine are called *Quantum Finite Automata*, and are defined in [10] as:

Definition 4.1.1. *A quantum finite-state automaton (QFA) is a real-time quantum automaton where H, s_{init}, and the U_a all have a finite dimensionality n. A quantum regular language (QRL) is a quantum language recognized by a QFA.*

Here we have no big results about relationships between classical languages and languages recognized by QFA, but a lot of closure properties for QFA were provided.

Closure Properties of QRLs

In this first introduction to Quantum Automata, the function $p_Q(w)$ has been threated by its 'numerical' meaning and not translating it into a characteristic function for a language in a more computer science fashion.

First, as in [10], we define two operations on quantum automata that allow us to add and multiply quantum languages.

Definition 4.1.2. *If u and v are vectors of dimension m and n, respectively, their direct sum $u \oplus v$ is the $(m+n)$-dimensional vector $(u_1, u_2, \cdots u_m, v_1, v_2, \cdots v_m)$. If M and N are two matrices then their direct sum is defined as* $M \oplus N = \left(\begin{array}{c|c} M & 0 \\ \hline 0 & N \end{array}\right)$
Then if Q and R are quantum automata with the same input alphabet, and if a and b are complex numbers such that $|a|^2 + |b|^2 = 1$, the weighted direct sum $aQ + bR$ is a brand new quantum automata that has initial state $\hat{s}_{init} = as^Q_{init} \oplus bs^R_{init}$, projection operator $\hat{P}_{accept} = P^Q_{accept} \oplus P^R_{accept}$, and transition matrices $\hat{U}_{accept} = U^Q_{accept} \oplus U^R_{accept}$

Lemma 2. *If Q and R are two QFA and $|a|^2 + |b|^2 = 1$, then $aQ \oplus bR$ is a QFA and $f_{aQ \oplus bR} = |a|^2 f_Q + |b|^2 f_R$. Thus if $f_1, f_2, \dots, f_k$ are QRLs and $c_1, c_2, \dots, c_n$ are real constants such that $c_i > 0 \quad \forall i \in 1, 2, \dots n$ and $\sum_{i=1}^n c_i = 1$, then also $\sum_{i=1}^n c_i f_i$ is a QRL.*

Definition 4.1.3. *if u and v are vectors of dimension m and n respectively, then their tensor product $u \otimes v$ is the mn-dimensional vector w, where $w_{(i,j)} = u_i v_j$ and $(i,j) = n(i-1) + j$. If M and N are two* *matrices with dimension $m \times m$ and $n \times n$ respectively, then $M \otimes N$ is the $mn \times mn$ matrix O where each element is defined as $O_{(i,k),(j,l)} = M_{ij}N_{kl}$; then, if Q and R are quantum automata with the same input alphabet, $Q \otimes R$ is defined by taking the tensor products of their respective s_{init}, P_{accept} and the U_a.*

Lemma 3. *If Q and R are QFAs, then $Q \otimes R$ is a QFA and $f_{Q\otimes R} = f_Q f_R$. Therefore, the product of any number of QRLs is a QRL.*

Lemma 4. *For any $c \in [0, 1]$, the constant function $f(w) = c$ is a QRL.*

Lemma 5. *If f is a QRL, then $\bar{f} = 1 - f$ is a QRL.*

Theorem 6 (Pumping Lemma for QRLs)**.** *If f is a QRL, then for any word w and any $\epsilon > 0$ there is a k such that*

$$|f(uw^k v) - f(uv)| < \epsilon$$

for any pair of words u, v. Moreover, if the automaton for f has $n-$dimensionality (the transition matrices U_a has dimension n) there is a constant c such that $k < (c\epsilon)^{-n}$

We will then refer to finite version of this model of automata as *Measure Once* Quantum Finite Automata.

4.2 1-way Quantum Finite Automata

Independently from [10], Quantum Finite Automata were introduced in the same year also in [7].
In [7], a MM-QFA is defined as a 6-tuple $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ where:

- Q is a finite set of states
- Σ is the finite input alphabet
- δ is the transition function
- $q_0 \in Q$ is the initial state
- $Q_{\text{acc}} \subseteq Q$ is the set of accepting state
- $Q_{\text{rej}} \subseteq Q$ is the set of rejecting state

From Q_{acc} and Q_{rej} we can define the set Q_{non} as:

$$Q_{\text{non}} = Q \setminus (Q_{\text{acc}} \cup Q_{\text{rej}})$$

containing all the non-halting states; it's supposed that the element in $Q_{\text{acc}} \cup Q_{\text{rej}}$ are halting states, $Q_{\text{acc}} \cap Q_{\text{rej}} = \emptyset$ and that $q_0 \in Q_{\text{non}}$.
The Hilbert Space $\mathcal{H}_Q = \text{span}\{|q\rangle \mid q \in Q\}$ (with $\|v\| = 1 \ \forall v \in \mathcal{H}_Q$, (it's also called $l_2(Q)$) can be splitted in 3 subspaces:

$$E_{\text{acc}} = \text{span}\{|q\rangle \mid q \in Q_{\text{acc}}\}$$
$$E_{\text{rej}} = \text{span}\{|q\rangle \mid q \in Q_{\text{rej}}\}$$
$$E_{\text{non}} = \text{span}\{|q\rangle \mid q \in Q_{\text{non}}\}$$

and correspondingly there are three projectors $P_{\text{acc}}, P_{\text{rej}}, P_{\text{non}}$ onto the three subspaces, respectively. Thus $\{P_{\text{acc}}, P_{\text{rej}}, P_{\text{non}}\}$ is a projective measurement on $\mathcal{H}_Q$.

Moreover, we need to add to the alphabet Σ two symbols $\kappa, \$$ being the left and right end marker for the tape: the new alphabet $\Gamma = \Sigma \cup \{\kappa, \$\}$ is called the *working alphabet*.

A state $|\psi\rangle$ of M is a superposition of states in Q:

$$|\psi\rangle = \sum_{q_i \in Q} \alpha_i \, |q_i\rangle, \;\; \alpha_i \in \mathbb{C} \tag{4.4}$$

where, eventually, some α_i can be 0.

We can now introduce how δ is defined for a generic MM-QFA.
In [7], the definition of δ for MM-QFA is derived from the definition of the 2-way QFA version: to have a dependecy free description, we will take it from [1];
in [1], it is defined as a function $\delta : Q \times \Gamma \times Q \to \mathbb{C}$ where the value of $\delta(q_1, a, q_2)$ is the amplitude of the state $|q_2\rangle$ in the superposition of state that M reaches when it is in the state $|q_1\rangle$ and the input symbol is a. We can now define the operator V_a (with $a \in \Gamma$) that acts as the generalization of δ: while the latter takes as input both the starting and the ending state, the former is used to describe the whole superposition obtained starting from a specific state $|q_i\rangle$ and reading a specific symbol $a \in \Gamma$:

$$V_a(|q_i\rangle) = \sum_{q_j \in Q} \delta(q_i, a, q_j) \, |q_j\rangle \tag{4.5}$$

Its a basic requirement that all the V_a are unitary (to keep the norms sum up to 1).

Once we have defined V_a, we can see how it is used to create an actual computation. At the beginning, the state is $|q_0\rangle$. Let's suppose we are in a generic step of computation for an input word w, the actual state of M is a generic superposition $|\phi\rangle = \sum_{q_i \in Q} \alpha_i \, |q_i\rangle$ and we read the symbol $x \in \Gamma$;
first of all, V_x is applied to $|\psi\rangle$ obtaining a new state $|\psi'\rangle$ defined as:

$$|\psi'\rangle = V_x(|\psi\rangle) = \sum_{q_i \in Q} \alpha_i V_x(|q_i\rangle)$$

Before explaining how the computation step is ended, we have to introduce the notion of *Observable for a MM-QFA*.
An Observable O for MM-QFA is actually a decomposition of $\mathcal{H}_Q$ into m subspaces $E_1 \oplus E_2 \oplus \cdots \oplus E_m = \mathcal{H}_Q$, where the E_j are pairwise orthogonal: to each $j = 1, 2, \ldots m$ will correspond a different outcome. Let's suppose a generic MM-QFA is in the superposition $|\phi\rangle$, then observing it using an observable O will result in one of these outcomes and $|\phi\rangle$ will be modified; in particular, let $|\phi_j\rangle$ be the projection of $|\phi\rangle$ in E_j (if we suppose that each E_j has an associated projection matrix P_j, then $|\phi_j\rangle = P_j \, |\phi\rangle$), thus $|\phi\rangle$ can be rewritten as $\sum_{j=1}^{m} |\phi_j\rangle$; the result of the measuration of $|\phi\rangle$ using O has the following Properties:

- it is completely (and truly) random, each component $|\phi_j\rangle$ has a probability $|| \;\; |\phi_j\rangle \;\; ||^2$ to be seen
- Immediately after the observation, the M's superposition collapses to $\frac{1}{|| \, |\phi_j\rangle \, ||} \, |\phi_j\rangle$ where j is the the particular outcome observed

Returning to our computation, after obtaining $|\phi'\rangle$, it is measured (observed) using the Observable $E_{\mathrm{acc}} \oplus E_{\mathrm{rej}} \oplus E_{\mathrm{non}}$. This observation gives a result $x \in E_j$ with a probability that is equal to the amplitude of the projection of $|\phi'\rangle$ in the subspace E_j namely, $\|P_j\,|\phi'\rangle\,\|^2$;
after this operation is performed, the superposition collpses to the projection stated above: in this case, if we get $|\phi'\rangle \in E_{\mathrm{acc}}$ or $|\phi'\rangle \in E_{\mathrm{rej}}$ the input is accepted or rejected rispectively. If instead, $|\phi'\rangle \in E_{\mathrm{non}}$, then the operations are re-executed. If, wlog, the superposition of a MM-QFA is:

$$|\phi'\rangle = \sum_{q_i \in Q_{\mathrm{acc}}} \alpha_i\,|q_i\rangle + \sum_{q_j \in Q_{\mathrm{rej}}} \beta_j\,|q_j\rangle + \sum_{q_k \in Q_{\mathrm{non}}} \gamma_k\,|q_k\rangle \tag{4.6}$$

then after the observation we just described we could get an accepting state with probability $\sum |\alpha_i|^2$, a rejecting state with probability $\sum |\beta_j|^2$ and the computation continues with probability $\sum |\gamma_k|^2$. In the last case, the system collapses to the state $\frac{|\varphi\rangle}{\|\,|\varphi\rangle\,\|}$ where $|\varphi\rangle = \sum_{q_k \in Q_{\mathrm{non}}} \gamma_k\,|q_k\rangle$ (the operation we just made on $|\psi\rangle$ is called normalization and it's made so the probabilities sum up to 1).

To better explain the above notation, we will provide an example. Let $M = (Q, \Sigma, \delta, q_0, Q_{\mathrm{acc}}, Q_{\mathrm{rej}}))$ be a MM-QFA defined as just introduced, where:

- $Q = \{q_0, q_1, q_{\mathrm{acc}}, q_{\mathrm{rej}}\}$
- $\Sigma = \{a\}$
- q_0 is the initial state.
- $Q_{\mathrm{acc}} = \{q_{\mathrm{acc}}\}$
- $Q_{\mathrm{rej}} = \{q_{\mathrm{rej}}\}$

We shall now define the transition function δ. Since defining it case by case would make example pretty heavy, we will define it using the V_x notation:

$$\begin{aligned}
V_a(|q_0\rangle) &= \frac{1}{2}\,|q_0\rangle + \frac{1}{2}\,|q_1\rangle + \frac{1}{\sqrt{2}}\,|q_{\mathrm{rej}}\rangle \\
V_a(|q_1\rangle) &= \frac{1}{2}\,|q_0\rangle + \frac{1}{2}\,|q_1\rangle - \frac{1}{\sqrt{2}}\,|q_{\mathrm{rej}}\rangle \\
V_\$(|q_0\rangle) &= |q_{\mathrm{rej}}\rangle \\
V_\$(|q_1\rangle) &= |q_{\mathrm{acc}}\rangle
\end{aligned}$$

There are some transitions that we have not described, for example the cases were the input for V is $|q_{\mathrm{acc}}\rangle$. This because, when the measurement is made, if the state is projected in the E_{non} subspace, then all the components of $|q_{\mathrm{acc}}\rangle$ are dropped; so in the next step, we will have a superposition that presents only either q_0 or/and q_1. On the other hand, if the state is projected in the E_{acc} subspace, the computation ends and of course we don't need the transition V anymore. (the same idea can be moved to the case of $|q_{\mathrm{acc}}\rangle$).

Next, we show how this automaton behaves on input word $x = aa$. (which is trasformed to the word $x = aa\$$) The automaton starts in $|q_0\rangle$; then V_a is applied an what we obtain is $\frac{1}{2}\,|q_0\rangle + \frac{1}{2}\,|q_1\rangle + \frac{1}{\sqrt{2}}\,|q_{\mathrm{rej}}\rangle$. This state is observed and two outcomes are possible:

- with a probability $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$, a rejecting state is observed. In this case, the superposition collpses to $|q_{\text{rej}}\rangle$, the word is rejected and the computation terminates.
- otherwise, with a probability $(\frac{1}{2})^2 + (\frac{1}{2})^2 = \frac{1}{2}$, a non-halting state is observed and the superposition collapses to $\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle$ and the computation continues

If the computation continues from the previous step, V_a is applied again, obtaining the superposition:

$$\begin{aligned}
V_a(\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle) &= \frac{1}{2}V_a(|q_0\rangle) + \frac{1}{2}V_a(|q_0\rangle) \\
&= \frac{1}{2}(\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle + \frac{1}{\sqrt{2}}|q_{\text{rej}}\rangle + \frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle - \frac{1}{\sqrt{2}}|q_{\text{rej}}\rangle) \\
&= \frac{1}{2}(|q_0\rangle + |q_1\rangle) \\
&= \frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle
\end{aligned}$$

From which we can state that V_a maps the state $\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle$ to itself. Of course in this case we have 0 probability of both accepting and rejecting since there are no components of q_{rej} and q_{acc}, which means that the superposition remains unchanged. In the last step, we apply the operator $V_\$$, mapping the superposition to $\frac{1}{2}|q_{\text{acc}}\rangle + \frac{1}{2}|q_{\text{rej}}\rangle$. This is observed: with probability $\frac{1}{4}$ we observe the state q_{rej} and with probability $\frac{1}{4}$ we see the state q_{acc}. As we will see after this example, MM-QFA also comes with a concept of **total** state, that is used as an 'accumulator' for the acceptance and rejection probability that can be encountered during a computation; in this case, the total accepting probability is just $\frac{1}{4}$ (from the last step), while the rejection probability is $\frac{1}{2}$ (from the first step) + $\frac{1}{4}$ (from the last step) = $\frac{3}{4}$.

Moreover, if we define the set $\mathcal{V} = \mathcal{H}_Q \times \mathbb{R} \times \mathbb{R}$ then the **total** state of a MM-QFA can be described as a triple inside $\mathcal{V}$.
A Machine M with state $(\psi, p_{\text{acc}}, p_{\text{rej}}) \in \mathcal{V}$ has an acceptance probability p_{acc}, rejection probability p_{rej} and neither with probability $\|\psi\|^2$ (and the current superposition of internal states is $|\psi\rangle$). For each $\sigma \in \Gamma$ there exists an operator $T_\sigma : \mathcal{V} \to \mathcal{V}$ that describes the evolution of the machine from a generic state to a new one after reading the symbol σ defined as:

$$T_\sigma(\psi, p_{\text{acc}}, p_{\text{rej}}) = (P_{\text{non}}V_\sigma(\psi), \|P_{\text{acc}}V_\sigma(\psi)\|^2 + p_{\text{acc}}, \|P_{\text{rej}}V_\sigma(\psi)\|^2 + p_{\text{rej}})$$

For a word $x = \sigma_1\sigma_2\cdots\sigma_n \in \Gamma^*$, we define $T_x = T_{\sigma_n}T_{\sigma_{n-1}}\cdots T_{\sigma_1}$; so, if for example $(\psi, p_{\text{acc}}, p_{\text{rej}}) = T_{\kappa w\$}(|q_0\rangle, 0, 0)$ then the quantum automata M accepts w with probability p_{acc} and rejects it with probability p_{rej}. Therefore, we define a norm on $\mathcal{V}$ as

$$\|(\psi, p_{\text{acc}}, p_{\text{rej}})\| = \frac{1}{2}(\|\psi\| + p_{\text{acc}} + p_{\text{rej}})$$

and let $\mathcal{B} = \{v \in \mathcal{V} \mid \|v\| \leq 1\}$. A trivial calcucation reveals that there exists a fixed constant c such that $\|T_x v - T_x v'\| \leq c\|v - v'\|$, $\forall v, v' \in \mathcal{B}$ and $x \in \Gamma^*$.

5

About acceptance conditions

While introducing the defintions for Measure Once Automata (QFA) we deliberately avoided to introduce the definition of *language accepted.* In Quantum Automata theory a lot of definitions for language acceptance are used. We are going to introduce a serie of definitions and theorems to describe most of them [4].

Definition 5.0.1 (*cut-point*)**.** *A language L is said to be accepted by a QFA A with cut-point λ iff*

$$L = \{w \in \Sigma^* \mid p_A(w) > \lambda\}$$

Definition 5.0.2 (*Isolated cut point*)**.** *Let L be a language accepted with cut-point λ. If there exists $\epsilon \in (0, 1/2]$ such that*

$$|p_A(w) \geq \epsilon| \ \ \forall w \in \Sigma^*$$

then we say that λ is isolated by ϵ

The relevance of isolated cut point acceptance on finite automata is due to the fact that, in this case, we can arbitrarily reduce the classicification error probability of an input word w by repeating a constant number of times its parsing and taking the majority of the answers.

An acceptance model that is considered to be more reliable than the isolated cut point is the Monte Carlo.

Definition 5.0.3 (*Monte Carlo*)**.** *Let L be a language. L is said to be accepted by a QFA A in Monte Carlo mode iff there exists $\epsilon \in (0, 1/2]$ such that:*

$$w \in L \to p_A(w) = 1 \tag{5.1}$$

$$w \notin L \to p_A(w) \leq \epsilon \tag{5.2}$$

$$\tag{5.3}$$

Moreover, we can introduce one last definition of acceptance called *with bounded error.*

Definition 5.0.4 (*with bounder error*)**.** *a QFA A is said to accept a language L with a bounded error*

$\epsilon \in [0, 1/2)$ *iff*

$$w \in L \leftrightarrow p_A(w) \geq 1 - \epsilon \tag{5.4}$$

$$w \notin L \leftrightarrow p_A(w) \leq \epsilon \tag{5.5}$$

$$\tag{5.6}$$

There exists a clear and well defined equivalence between acceptance *with bounded error* and *isolated cut point*.

Theorem 7. *An acceptance with bounded error ϵ is equivalent to an acceptance with cutpoint 1/2 isoltated by 1/2 - ϵ.*

Proof. By definition, we know that with acceptance with cutpoint 1/2 isolated by 1/2 - ϵ these two following conditions hold:

$$L = \{w \in \Sigma^* \mid p_A(w) > 1/2\} \tag{5.7}$$

$$|p_A(w) - 1/2| \geq 1/2 - \epsilon \;\; \forall w \tag{5.8}$$

If we take the second condition, we know that $p_A(w) - 1/2 > 0$ iff $p_A(w) > 1/2$, which holds $\forall w \in L$. Under this assumptions 5.8 can be rewritten as:

$$p_A(w) - 1/2 \geq 1/2 - \epsilon = p_A(w) \geq 1 - \epsilon \; \forall winL$$

In the other case, if the take $p_A(w) - 1/2 \leq 0$, it holds iff $p_A(w) \leq 1/2$ leading to $w \notin L$. Thus 5.8 can be rewritten as:

$$-p_A(w) + 1/2 \geq 1/2 - \epsilon = p_A(w) \leq \epsilon \; \forall w \notin L$$

If we look at the results we got, they are exactly the condition necessary to define an acceptance with bounded error. □

Theorem 8. *An acceptance with cut point 1/2 isolated by ϵ coincide with the acceptance with bounded error 1/2 - ϵ.*

Proof. By definition, we know that in acceptance with bounded error the two following hold (doing a substitution of the actual error):

$$w \in L \leftrightarrow p_A(w) \geq 1/2 + \epsilon \tag{5.9}$$

$$w \notin L \leftrightarrow p_A(w) \leq 1/2 - \epsilon \tag{5.10}$$

$$\tag{5.11}$$

Looking at the first condition, we know that $\forall winL$:

$$p_A(w) \geq 1/2 + \epsilon = p_A(w) - 1/2 \geq epsilon$$

which also leads to $p_A(w) \geq 1/2 \; \forall winL$.

Looking at the second condition, $\forall w \notin L$ it holds to:

$$p_A(w) \leq 1/2 - \epsilon = -(p_A(w) - 1/2) \geq \epsilon$$

and it also leads to $p_A(w) < 1/2 \, \forall w \notin L$.

It's straightforward to see that it all yields to:

$$L = \{w \in \Sigma^* \mid p_A(w) > 1/2\}$$

which defines an acceptance with cut point 1/2. Moreover, we have that:

$$|p_A(w) - 1/2| \leq \epsilon$$

that holds for both $w \in L$ and $w \notin L$. Thus, we have achieved an acceptance with cut-point 1/2 isolated by ϵ □

Moreover, we can introduce a result about cut point isolated with $\lambda \neq 1/2$ taken from [9].

Theorem 9. *The acceptance with cut point λ isolated by ϵ is equivalent to an acceptance with cutpoint λ isolated by:* $\begin{cases} \frac{\epsilon}{(2(1-\lambda))} & if \lambda < 1/2 \\ \frac{\epsilon}{(2\lambda)} & if \lambda > 1/2 \end{cases}$

Using this theorem, we can conclude that

Theorem 10. *The acceptance with cut point λ isolated by ϵ is equivalent to an acceptance with a bounded error* $\begin{cases} \frac{1}{2} - \frac{\epsilon}{(2(1-\lambda))} & if \lambda < 1/2 \\ \frac{1}{2} - \frac{\epsilon}{(2\lambda)} & if \lambda > 1/2 \end{cases}$

Proof. Using 9 we know that we can move from an acceptance with cut point cut point λ isolated by ϵ to a cut point 1/2 isolated by ϵ'. Then, using 8 we can move from a cut point 1/2 isolated by ϵ' to a bounded error 1/2 - ϵ'. Doing an easy substitution it yields to the result of the theorem. □

A last definition is called *acceptance with probability*:

Definition 5.0.5. *A MM-QFA M recognizes a language L with propability p*

- *if it accepts any string $x \in L$ with a probability $> p$*
- *if it accepts any word $y \notin L$ with a probability that is $\leq p$.*

If we refer to the language L recognized by a MM-QFA M without specific p, then we refer L to a language accepted from M with some probability $\frac{1}{2} + \epsilon$, with $\epsilon > 0$.

6

Characterization of expressive power

In this chapter we will describe the results obtained throughout the years about acceptance power of both MO-QFA and MM-QFA.

6.1 Measure Once Quantum Finite Automata

6.1.1 MO-QFA

A **MO-QFA** M is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where Q is the set of states, Σ is the alphabet (united with the end marker \$), δ is the transition function, q_0 is the initial state and F the set of final state. δ is defined as a function going from $Q \times \Sigma \times Q$ to $\mathbb{C}$ and describes the probability of going from state q to q' after reading a symbol in Σ. The δ function has to be unitary, so it has to respect the condition:

$$\sum_{q' \in Q} \overline{\delta(q_1, a, q')} \delta(q_2, a, q') = \begin{cases} 1 \text{ if } q_1 = q_2 \\ 0 \text{ otherwise} \end{cases}$$

The δ function as defined above is actually obtained through a set of unitary matrices $\{U_a\}_{a \in \Sigma}$, where U_a represents the unitary transition of M applied after reading a. So, given a generic state $|\phi\rangle = \sum_{q_i \in Q} \alpha_i |q_i\rangle$ of M, reading a symbol a means (in term of U_a):

$$|\phi'\rangle = U_a |\phi\rangle = \sum_{q_i, q_j \in Q} \alpha_i \delta(q_i, a, q_j) |q_j\rangle$$

As for the definition of QFA provided in [10], the measurement is taken only at the end of the computation; the measurement is described through a projection matrix $P = \sum_{k \in F} |k\rangle \langle k|$ and the probability of M accepting a word x is given by:

$$p_M(x) = ||PU_x |\phi\rangle ||^2 = \sum_{kinF} |(U_x |\phi\rangle)_k|^2$$

where

- $U_x = U_{x_{|x|}} \dots U_{x_2} U_{x_1}$

- $(U_x |\phi\rangle)_k$ denotes the k-th component of the vector $|\phi'\rangle = U_x |\phi\rangle$

6.1.2 Splitting Languages accepted from MO-QFA

The class of languages accepted from a QFA in [5] is then splitted in classes. The first class introduced is the $\mathbf{RMO}_\epsilon$: it contains all the languages that can be accepted from a MO-QFA with cut-point isolated by a *margin* at least ϵ. From this definition we can also derive the definition of a new class, **RMO**, defined as:

$$\mathbf{RMO} = \bigcup_{\epsilon>0} \mathbf{RMO}_\epsilon$$

If we set $\epsilon = 0$ we obtain the language $\mathbf{RMO}_0 = \mathbf{UMO}$.

One of the results that were achieved in [5], is to define that **RMO** is actually equals to the set of languages acceptable from RFAs. Restricting MO-QFAs to accept with bounded error greatly reduces their accepting power; Since MM-QFAs can accept only a proper subset of the regular languages if they are required to accept with bounded error and since every MO-QFA can be simulated exactly by an MM-QFA, the class **RMO** is a proper subset of the regular languages.

Theorem 11. *The class **RMO** is exactly the class of languages accepted by RFAs.*

Two lemmas useful to prove this statement are:

Lemma 12. *Let U be an unitary matrix. For any $\epsilon > 0$ there exists an integer $n > 0$ such that for all vectors x, with $||x|| \leq 1$, it is true that $||(I - U^n)x|| < \epsilon$*

Lemma 13. *Let $|\phi\rangle$ and $|\psi\rangle$ be two complex vectors such that $||\ |\phi\rangle\ || = ||\ |\psi\rangle\ || = 1$, and $||\ |\phi\rangle - |\psi\rangle\ || \leq \epsilon$. The total variation distance between the probability distributions resulting from the measurement of $|\phi\rangle$ and $|\psi\rangle$ is at most 4ϵ.*

From the theorem 11, it follows that $\mathbf{RMO}_\epsilon = \mathbf{RMO}_{\epsilon'}$ for all $\epsilon, \epsilon' > 0$. So, we can conclude that the class of languages recognizable from a MO-QFA are actually two: **RMO** and $\mathbf{RMO}_0 = \mathbf{UMO}$. From theorem 11 it also follows that:

Theorem 14. ***RMO** class is closed under Boolean operations, inverse homomorphisms and word quotient. It is not closed under homomorphisms.*

Now, the question that follows pretty naturally is:
which is the relation between **RMO** and **UMO**? Let's consider the language $L = \{x \in \{a,b\}^* |\ |x_a| \neq |x_b|\}$: it's shown in [5] that there exists a MO-QFA that accepts L with *cutpoint* $= 0$.

Proof. The matrix U_a is defined as a rotation matrix for the initial state $|q_0\rangle$; U_b is defined through the relation $U_b^{-1} = U_a$ so that applying U_b would 'undo' a rotation madre through U_a (and viceversa). So, if the operators U_a and U_b are applied the same number of times ($|x_a| = |x_b|$), then the state of the machine would remain $|q_0\rangle$ with a 0 probability of acceptance. While, if the number of times U_a is applied is bigger than the times U_b is applied (or viceversa) we will have a non-zero probability that once the measuration is done we will get the state $|q_1\rangle$ □

From that proof we can derive that inside **UMO** there are also languages that are not regular. Therefore, the relation between **RMO** and **UMO** is as follows:

$$\mathbf{RMO} \subset \mathbf{UMO}$$

In the end, we can link the MO-QFAs to the PRFAs using the following theorem:

Theorem 15. *Let L be a language accepted by MO-QFA M with cut-point γ:*

1. *Then there exists a PRFA accepting same language with cut-point γ'*
2. *If M accepts L with bounded error, then there exists a PRFA accepting L with bounded error*

6.1.3 UMO expressive power

As shown in 6.1.2, there are non regular languages which can be recognized by MO-QFA if they are allowed to accept without bounded error. (i.e. inside the class **UMO**) A natural question arises: what is the exact characterization of the class **UMO**?

This question has been answered in [3]. To prove the theorem we will need the following lemma:

Lemma 16. *if M is any unitary matrix of order m over the complex field and I_m is the identity over C^m, then for any ϵ there exists $v \in N$ such that it holds*

$$\|M^v - I_m\| \leq \epsilon$$

The Characterization of **UMO** can be done using the following:

Lemma 17. *if L is a language accepted by a MO-QFA A, then for each $x \in \Sigma^*$ and for every $w \in L$, there exists a positive integer v such that*

$$wx^v \in L$$

Proof.

$$\begin{aligned}
|p_A(w) - p_A(wx^v)| &= \left| \sum_{k \in F} (|(U_w |\phi\rangle)_k|^2 - |(U_{wx^v} |\phi\rangle)_k|^2) \right| \\
&\leq 2 \sum_{k \in F} \big|\, |(U_w |\phi\rangle)_k| - |(U_{wx^v} |\phi\rangle)_k| \,\big| \\
&\leq 2 \sum_{k \in F} |(U_w |\phi\rangle)_k - (U_{wx^v} |\phi\rangle)_k| \\
&= 2 \sum_{k \in F} |(((I - (U_x)^v) U_w) |\phi\rangle)_k| \\
&\leq 2 \sum_{k \in F} \|(I - (U_x)^v)\| \\
&= 2|F| \|(I - (U_x)^v)\|
\end{aligned}$$

Since $w \in L$, then $p_A(w) > \lambda$ and we can set $p_A(w) - \lambda = \delta > 0$. By lemma 16 we know that there exists some $\hat{v}$ such that:

$$\|I - (U_x)^v\| \leq \frac{\delta}{4|F|}$$

which yields to

$$|p_A(w) - p_A(wx^v)| \leq \frac{\delta}{2}$$

Thus, we can conclude that $wx^v \in L$ since:

$$p_A(wx^v) - \lambda \geq p_A(w) - \frac{\delta}{2} - \lambda \geq \frac{\delta}{2} \geq 0$$

□

From the last lemma we can derive two useful theorems

Theorem 18. *MO-QFA accepting with cut point not isolated can accept only the empty language or languages containing an infinite number of words.*

Theorem 19. ***UMO** is not closed under complementation.*

6.2 Measure Many Quantum Finite Automata

6.2.1 MM-QFA

As we said above, the definition of MM-QFA is exactly the same as the one we described in 4.2.

6.2.2 Properties for MM-QFA

A first result that was given inside [7] about language accepted from MM-QFA is the following:

Theorem 20. *Let L be any language recognized by a MM-QFA with bounded error. Then L is regular.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ be a MM-QFA which recognizes L with probability of error bounded by $1/2 - \epsilon$. We will write $w \equiv_L w'$ if $\forall y \in \Sigma^*$, we have $wy \in L \leftrightarrow w'y \in L$. The relation L is an equivalence relation, and partitions Σ^* into finitely many equivalence classes if and only if L is regular. Let $W \subseteq \Sigma^*$ be any set of strings which are pairwise equivalent with respect to $\equiv_L$. In order to prove the proposition, if suffices to show that W must be finite. For $w \neq w'$ (both in W) there must exist a string $y \in \Sigma^*$ such that $wy \in L \leftrightarrow w'y \notin L$. Hence, let $v = T_{\kappa w}(|q_0\rangle, 0, 0)$ and $v = T_{\kappa w'}(|q_0\rangle, 0, 0)$; since M has error propability bounded away from $\frac{1}{2}$ by ϵ, we have that $\|T_{y\$}v - T_{y\$}v' > 2\epsilon$. Consequently, we have $\|v - v'\| > 2\epsilon/c$ so that the set

$$\{T_{\kappa w}(|q_0\rangle, 0, 0) \mid w \in W\} \tag{6.1}$$

must be finite. Therefore, W must be finite as well. □

The other way of the theorem is not true (regular → recognizable by a MM-QFA with bounded error), in fact in [7] it is shown that

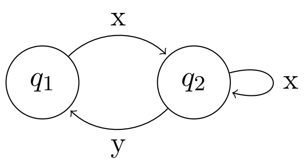


Figure 6.1: The new forbidden structure

Theorem 21. *The language $L = \{a,b\}^*a$ cannot be recognized by MM-QFA with bounded error*

From the two theorems above we can conlude that:

Theorem 22. *The set of languages accepted from a MM-QFA with bounded error is a proper subclass of regular languages*

Proof. Let $\mathcal{L}$ be the set of languages accepted from a MM-QFA with buonded error. From theorem 20, we know that $L \in \mathcal{L} \to L$ is regular, so for sure $\mathcal{L}$ is a subset of the regular languages. Moreover, $\hat{L} = \{a,b\}^*a \notin \mathcal{L}$ and $\hat{L}$ is clearly regular; so $\hat{L}$ is the witness of the difference between the set of regular languages and $\mathcal{L}$. □

The first link we can create between MM-QFA and classic automata is:

Theorem 23. *If a language L is accepted by a PRFA then it is accepted by a MM-QFA with the same probability of acceptance.*

6.2.3 Splitting languages recognized by MM-QFA

From the definitions we gave of **RMO** and **UMO**, it's easy to derive a definition for their counterpart for MM-QFA, **RMM** and **UMM** respectively.

Unlike the closure properties of the classes **RMO** and **UMO**, which can be derived easily, the closure properties of the classes **RMM** and **UMM** are not as evident and in one important case are unknown: in fact

Definition 6.2.1. *It is still unknown if the class **RMM**, **UMM** are closed under boolean operations.*

But for a fixed $\epsilon > 0$ it's true that:

Definition 6.2.2. *The class $\boldsymbol{RMM}_\epsilon$ is closed under complement*

As we introduced in 25, in [5] a condition for a Language to be inside **RMM** is derived too. A language L is said to satisfy the partial order condition if the minimal DFA for L satisfies the partial order condition.

Definition 6.2.3. *A DFA is said to satisfy the partial order condition if it does not contain two distinguishable states $q_1, q_2 \in Q$ such that there exists two words $x, y \in \Sigma^+$ where $\delta(q_1, x) = \delta(q_2, x) = q_2$ and $\delta(q_2, y) = q_1$. (Two states are said to be distinguishable if there exists a word $z \in \Sigma^+$ such that $\delta(q_1, z) \in F$ and $\delta(q_2, z) \notin F$).*

From this new forbidden structure, we can define a condition for a language to be inside **RMM**:

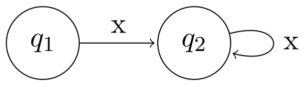


Figure 6.2: The 'forbidden construction' from the theorem [25].

Theorem 24. *If the minimal DFA for a Language L does not satisfy the partial order condition, then $L \notin \textbf{RMM}$*

(The condition is named partial order because once q_2 is visited, there is no way back from it, so we can introduce a partial order for the states of a DFA) The partial order condition was also proposed as a method to prove the closure of **RMM** under intersection. Actually, as we will see in Theorem [35], the class **RMM** results to be not closed under any binary boolean operation where the both arguments are significant.

6.2.4 MM-QFA expressive power and minimal automaton

In [1] a lot has been done to study the relation between acceptance of MM-QFA and minimal automaton. We will introduce some of them that will then lead us the definition of *partial order condition.*

Theorem 25. *Let L be a language and M be its minimal automaton. Assume that there is a word x such that M contains state q_1, q_2 satisfying:*

- $q_1 \neq q_2$
- *If M starts in q_1 and reads x, it passes to q_2*
- *If M starts in q_2 and reads x, it passes to q_2*
- *q_2 is neither an all-accepting nor an all-rejecting state.*

Then L cannot be recognized by a MM-QFA with probability at least $\frac{7}{9} + \epsilon$ for any fixed $\epsilon > 0$.

Something similar can also be achieved for RFA, in fact:

Theorem 26. *Let L be a language and M the minimal automaton accepting it. If M does not contain the forbidden structure introduced in [25], then L can be recognized by a RFA.*

The last theorem, united with the one introduced in [25] leads to a new theorem about language acceptance for MM-QFA:

Theorem 27. *A language L can be recognized by a MM-QFA with probability $\frac{7}{9} + \epsilon$ if and only if it can be recognized by a RFA*

Of course, we just introduced results for acceptance probabilities that are pretty high ($\frac{7}{9} + \epsilon$), but what about smaller ones? Always in [1], it shown that:

Theorem 28. *The language a^*b^* can be recognized by a MM-QFA with probability $p = 0.68$ where p is the root of $p^3 + p = 1$*

Proof. We will provide a description of a MM-QFA accepting this language. Let $M = (Q, \Sigma, \delta, \psi, Q_{\text{acc}}, Q_{\text{rej}})$ that automaton, defined as:

- $Q = \{q_0, q_1, q_{\text{acc}}, q_{\text{rej}}\}$
- $\Sigma = \{a, b\}$
- For δ function we will provide a more general definiton using operators V_a, V_b
- The initial state ψ is defined as $\sqrt{1-p}\,|q_0\rangle + \sqrt{p}\,|q_1\rangle$
- $Q_{\text{acc}} = \{q_{\text{acc}}\}$
- $Q_{\text{rej}} = \{q_{\text{rej}}\}$

The V_a and V_b operators are defined as:

- $V_a(|q_0\rangle) = (1-p)\,|q_0\rangle + \sqrt{p(1-p)}\,|q_1\rangle + \sqrt{p}\,|q_{\text{rej}}\rangle$
- $V_a(|q_1\rangle) = \sqrt{p(1-p)}\,|q_0\rangle + p\,|q_1\rangle - \sqrt{1-p}\,|q_{\text{rej}}\rangle$
- $V_b(|q_0\rangle) = |q_{\text{rej}}\rangle$
- $V_b(|q_1\rangle) = |q_1\rangle$
- $V_\$(|q_0\rangle) = |q_{\text{rej}}\rangle$
- $V_\$(|q_1\rangle) = |q_{\text{acc}}\rangle$

Notice that, for the first time we defined the V operator also for the symbol \$ of the working alphabet Γ, used to define how the Automata has to behave once that right end marker is read (In this case, it can be read as 'if at the end of the word, the state is $|q_0\rangle$, then reject, if it is $|q_1\rangle$ accept)

Let's prove that this automata actually accepts a^*b^* with probability p. We can split the possible input word in 3 classes:

1. if the input word $x \in a^*$. In that case, the V_b definition becomes useless and only the V_a and $V_\$$ operators are applied. If we start from ψ, and we apply V_a we get:

$$\psi = \sqrt{1-p}V_a(|q_0\rangle) + \sqrt{p}V_a(|q_1\rangle) \text{ with}$$
$$\sqrt{1-p}V_a(|q_0\rangle) = \sqrt{1-p}\Big((1-p)\,|q_0\rangle + \sqrt{p(1-p)}\,|q_1\rangle + \sqrt{p}\,|q_{\text{rej}}\rangle\Big)$$
$$\sqrt{p}V_a(|q_1\rangle) = \sqrt{p}\Big(\sqrt{p(1-p)}\,|q_0\rangle + p\,|q_1\rangle - \sqrt{1-p}\,|q_{\text{rej}}\rangle\Big)$$

If we look carefully to that equation we see that the terms with $|q_0\rangle$ can be rewrote like:

$$\begin{aligned}&(1-p)\sqrt{1-p}\,|q_0\rangle + \sqrt{p}\sqrt{p(1-p)}\,|q_0\rangle =\\&(1-p)\sqrt{1-p}\,|q_0\rangle + \sqrt{p^2(1-p)}\,|q_0\rangle =\\&(1-p)\sqrt{1-p}\,|q_0\rangle + p\sqrt{(1-p)}\,|q_0\rangle =\\&\sqrt{1-p}\,|q_0\rangle\,(1-p+p) =\\&\sqrt{1-p}\,|q_0\rangle\end{aligned}$$

The same thing can be made for $|q_1\rangle$ (we will not show all the steps, it's more like an exercise about radicals) and we obtain that the terms with $|q_1\rangle$ actually sum up to $\sqrt{p}\,|q_1\rangle$.

Therefore ψ after applying $V_a(|q_0\rangle)$ and $V_b(|q_1\rangle)$ is now:

$$\psi = \sqrt{1-p}\,|q_0\rangle + \sqrt{p}\,|q_1\rangle + (\sqrt{1-p}\sqrt{p} - \sqrt{p}\sqrt{1-p})\,|q_{\text{rej}}\rangle$$

The last term then cancels out due to its 0 coefficient ($\sqrt{1-p}\sqrt{p} - \sqrt{p}\sqrt{1-p}$), and then we have eventually proved that the operator V_a actually maps the initial state to itself. So, after reading the whole word $x \in a^*$, our state would always be: $\sqrt{1-p}\,|q_0\rangle + \sqrt{p}\,|q_1\rangle$. When the right end marker is read after x, using $V_\$$, $|q_0\rangle$ becomes $|q_{\text{rej}}\rangle$ and $|q_1\rangle$ becomes $|q_{\text{acc}}\rangle$. The ending state being measured is: $\sqrt{1-p}\,|q_{\text{rej}}\rangle + \sqrt{p}\,|q_{\text{acc}}\rangle$ and we will have an accepting state with probability p.

Note that in this case we took for granted that the observation step always results in the subspace E_{non} during the computation because until we reach the right end marker, we do not have any state either in the E_{acc} or in the E_{rej} subspace.

2. If the input is of the type a^*b^+. As we learned above, the operator V_a just maps the initial state to itself, and when we read the first b, the current state is actually the initial state ψ. Now let's apply the V_b operators as we did above for the V_a:

$$\begin{aligned}\psi &= \sqrt{1-p}V_b(|q_0\rangle) + \sqrt{p}V_b(|q_1\rangle) \text{ with}\\ \sqrt{1-p}V_b(|q_0\rangle) &= \sqrt{1-p}\,|q_{\text{rej}}\rangle\\ \sqrt{p}V_b(|q_1\rangle) &= \sqrt{p}\,|q_1\rangle\end{aligned}$$

And, this case, after applying the V_b operator, we explicitly apply the Observable: in fact, we have a $1-p$ probability that the word is rejected before the whole word is read. If instead, the observation goes in the E_{non} subspace, the new state of the machine would be $\sqrt{p}\,|q_1\rangle$. From here, if we keep applying the $V_b(|q_1\rangle)$ operator, we would just map the state to itself. So when we reach the \$, $|q_1\rangle$ is replaced with $|q_{\text{acc}}\rangle$ and we have a p acceptance probability.

3. if the input is not in the language a^*b^*. If the word is not inside this language, then its starting segment has the form $a^*b^+a^+$. We know that, when we read the first b, we have a $1-p$ rejecting probability. After that first b, the state is $\sqrt{p}\,|q_1\rangle$ for all the other bs. When we read the first a

after those bs, the state $\sqrt{p}\,|q_1\rangle$ becomes

$$\sqrt{p}\Big(\sqrt{p(1-p)}\,|q_0\rangle + p\,|q_1\rangle - \sqrt{1-p}\,|q_{\text{rej}}\rangle\Big)$$

applying $V_a(|q_1\rangle)$. In this case, we have a $p(1-p)$ rejecting probability; otherwise, the non-halting state of the machines become $|\psi\rangle = p\sqrt{1-p}\,|q_0\rangle + p\sqrt{p}\,|q_1\rangle$ (I actually found an error here in the proof inside [1]). All the other times we apply the V_a operator the state $|\psi\rangle$ remains unchanged. Eventually, we will find either a right end marker \$ or a b and $|q_0\rangle$ is mapped to $|q_{\text{rej}}\rangle$: in this case, we have a $p^2(1-p)$ rejecting probability. Summing up all the rejecting probabilities we found, we get that the machine M rejects a word $\notin a^*b^*$ with probability:

$$(1-p)+p(1-p)+p^2(1-p) = (1-p)(1+p+p^2) = \frac{1-p^3}{1-p}(1-p) = 1-p^3 = p$$

□

If we now look at the Minumal automata (RFA) that accepts a^*b^*, we see that it contains the 'forbidden structure' we introduced before; we can then state that:

Theorem 29. *There is a language that can be recognized from a MM-QFA with probability 0.68... but not with probability* $\frac{7}{9}+\epsilon$

Using the theorem we just introduced and the theorem 27, we can introduce one other theorem:

Theorem 30. *There is a language that can be recognized by a MM-QFA with probability 0.68... but not by a RFA.*

This last theorem can be improved showing that neither PRFA can accept that language; to do that we will introduce, as we did for RFA, a *forbidden structure* for PRFA too.

Definition 6.2.4. *Let L be a language and M the minimal automaton accepting it. Then if there exitsts words x, y and states $q_1, q_2 \in Q_M$ that satisfy the following:*

- *neither q_1 nor q_2 are all-accepting or all-rejecting states*
- *reading x in q_1 leads to q_1*
- *reading y in q_1 leads to q_2*
- *reading y in q_2 leads to q_2*
- *There is no $i > 0$ such that reading x^i from q_2 leads to q_2*

Then L cannot be recognized with probability $1/2+\epsilon$ $(\epsilon > 0)$ by a PRFA

It's pretty straightforward to see that the minimal automaton accepting a^*b^* contains this last forbidden construction too, leading to:

Theorem 31. *There is a language that can be recognized by a MM-QFA with probability 0.68... but not by a PRFA with probability $1/2+\epsilon$ for any $\epsilon > 0$.*

We then proved that MM-QFAs are actually more powerful than classic probabilistic automata.

In the 2001 other conditions (both sufficient and necessary) for the quantum languages where given in [2]. First of all, it is shown that the condition we introduced in 25 and its relaxed version 24, can be generalized to obtain a *necessary* condition for a language L to be recognized by a MM-QFA.

A Necessary condition

Theorem 32. *Let L be a a language. Assume that there are words x, y, z_1, z_2 such that its minimal automaton M contains states q_1, q_2, q_3 satisfying:*

- $q_2 \neq q_3$
- *if M starts in q_1 and reads x, it passes to q_2*
- *if M starts in the state q_2 and reads x, it passes to q_2*
- *if M starts in the state q_1 and reads y, it passes to q_3*
- *if M starts in the state q_3 and reads y, it passes to q_3*
- *for any word $t \in (x|y)^*$ there exists a word $t_1 \in (x|y)^*$ such that if M starts in the state q_2 and reads tt_1, it passes to q_2*
- *for any word $t \in (x|y)^*$ there exists a word $t_1 \in (x|y)^*$ such that if M starts in the state q_3 and reads tt_1, it passes to q_3*
- *if M starts in the state q_2 and reads z_1, it passes to an accepting state*
- *if M starts in the state q_2 and reads z_2, it passes to a rejecting state*
- *if M starts in the state q_3 and reads z_1, it passes to a rejecting state*
- *if M starts in the state q_3 and reads z_2, it passes to an accepting state*

Then, L cannot be recognized by a MM-QFA

A necessary and sufficient condition

After this *necessary* condition, in [2] a *necessary and sufficient* condition is introduced.

Theorem 33. *Let U be the class of languages whose minimal automaton does not contain "two cycles in a row". A language that belongs to U can be recognized by a MM-QFA if and only if its minimal deterministic automaton does not contain the 'forbidden construction' drawed in 6.1 and the 'forbidden construction' from Theorem 32.*

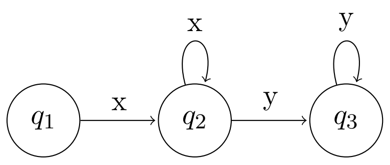


Figure 6.3: Two cycle in a row

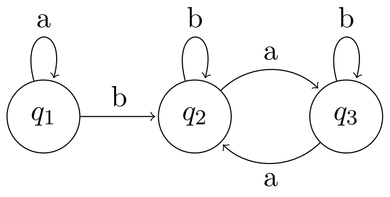


Figure 6.4: Automaton recognizing L_1

Not closure under union

From the Theorem 32 we can derive that the class of languages recognized by QFAs is not closed under union. Let L_1 be the language consisting of all words that start with any number of letters a and after first letter b (if there is one) there is an odd number of letters a.
The minimal automaton recognizing L_1 is shown in figure 6.4. That automaton satisfies the condition introduced in Theorem 32, so L_1 cannot be recognized by a QFA. But, we can also create two languages L_2 and L_3 defined as follows. L_2 consists of all words which start with an even number of letters a and after first letter b (if there is one) there is an odd number of letters a. L_3 consists of all words which start with an odd number of letters a and after first letter b (if there is one) there is an odd number of letters a. It's pretty straightforward that $L_1 = L_2 \cup L_3$. If we create the minimal automaton that accepts L_2, and the one accepting L_3 (their construction is pretty easy: just add a state q_4 connected to q_1 used to see that the number of a is either even for L_2 or odd for L_3), we would see that neither the first nor the second satisfy the condition in 32, so there actually exist two QFA accepting L_2 and L_3 respectively. But, as we showed above, there is no QFA accepting the union between $L_2 \cup L_3$ (namely, L_1), so:

Theorem 34. *The class of languages recognized by QFAs is not closed under union*

We can also see that L_2 and L_3 are disjoint, thus $L_1 = L_2 \Delta L_3$ and we can conclude that the class of languages accepted by QFA is not closed under symmetric difference. Being that class closed under complementation, we can assume that:

Theorem 35. *The class of languages recognizable by a QFA is not closed under any binary boolean operation where both arguments are significant*

More complex forbidden structures

If in the theorem 33 we allow the structure 'Two cicles in a row', then the theorem 33 is not true anymore. So, a new and more complex forbidden structure can be wrote:

Theorem 36. *Let L be a language and M be its minimal automaton. If M satisfies the following conditions (where $a, b, c, d, e, f, g, h, i \in \Sigma^*$):*

- *If M reads $x \in \{a, b, c\}$ in the state q_0, its state changes to q_x*
- *If M reads $x \in \{a, b, c\}$ in the state q_x, its state again becomes q_x*
- *If M reads any string consisting of a, b and c in the state $q_x(x \in \{a, b, c\})$, it moves to a state from which it can return to the same q_x by reading some (possibly, different) string consisting of a, b and c*
- *If M reads $y \in \{d, e, f\}$ in the state $q_x(x \in \{a, b, c\})$, it moves to q_{xy}*
- *If M reads $y \in \{d, e, f\}$ in the state q_{xy}, its state again becomes q_{xy}*
- *If M reads any string consisting of d, e and f in the state q_{xy} it moves to a state from which it can return to the same state q_{xy} by reading some (possibly, different) string consisting of d, e and f*
- *Reading h in the state q_{ad}, i in the state q_{be} and g in the state q_{cf} lead to accepting states. Reading g in q_{ae}, h in q_{bf} and i in q_{cd} lead to rejecting states*

then L is not recognizable by a QFA.

If we look at the Automata strucuture implied by the last Theorem, it's pretty like a tree: in fact, if we look carefully the structure of the Theorem 36 it can actually be generalized to any number of levels and any number of branchings at one level as long as every arc from one vertex to other is traversed the same number of times in paths leading to accepting states and in paths leading to rejecting states.

- Level 1 of a construction consists of a state q_1 and some words a_{11}, $a_{12}, \dots$
- Level 2 consists of the states q_{21}, $q_{22}, \dots$ where the automaton goes if it reads one of words of Level 1 in a state in Level 1. We require that, if the automaton starts in one of states of Level 2 and reads any string consisting of words of Level 1 it can return to the same state reading some string consisting of these words. Level 2 also has some words $a_{21}, a_{22}, \dots$
- Level 3 consists of the states $q_{31}, q_{32}, \dots$ where the automaton goes if it reads one of words of Level 2 in a state in Level 2. We require that, if the automaton starts in one of states of Level 3 and reads any string consisting of words of Level 2 it can return to the same state reading some string consisting of these words. Again, Level 3 also has some words $a_{31}, a_{32}, \dots$.
- ...
- Level n consists of the states $q_{n1}, q_{n2}, \dots$ where the automaton goes if it reads one of words of Level $n-1$ in a state in Level $n-1$.

Let us denote all different words in this construction as $a_1, a_2, a_3, \dots, a_m$. For a word a_i and a level j we construct sets of states B_{ij} and D_{ij}. A state q in level $j+1$ belongs to B_{ij} if the word a_i belongs to level j and M moves to q after reading a_i in some state in level j. A state belongs to D_{ij} if this state belongs to the Level n and it is reachable from B_{ij}.

Then, a last theorem about QFA acceptance using the minimal automaton can be described using the definition we just gave:

Theorem 37. *Assume that the minimal automaton M of a language L contains the 'forbidden construction' of the general form described above and, in this construction, for each D_{ij} the number of accepting states is equal to the number of rejecting states. Then, L cannot be recognized by a QFA.*

6.2.5 Space efficiency

Until now, the main focus was about the recognizing power of various QFA formalisms. But what about their effiency? In [1], two theorems concerning this matter were introduced:

Theorem 38. *Let L be a language recognized by a MM-QFA with n states. Then it can be recognized by a 1 way deterministic automaton with $2^{\mathcal{O}(n)}$ states.*

Moreover, an interesting result is introduced about *primes.* Let p be a prime number, consider the language $L_p = \{a^i | \ i \equiv 0 (\text{mod } p)\}$; it's easy to see that any deterministic automaton needs at least p states to recognize L_p: QFA can do better, in fact

Theorem 39. *For any $\epsilon > 0$, there is a MM-QFA with $O(\log p)$ states that recognizes L_p with probability $1 - \epsilon$*

The same language can be analyzed for PRFAs too.

Theorem 40. *Any PRFA recognizing L_p with probability $\frac{1}{2} + \epsilon$, for a fixed $\epsilon > 0$, has at least p states*

From the last two theorems, we can derive a third one:

Theorem 41. *For the language L_p, the number of states needed by a classical (RFA or PRFA) automata is exponential in the number of states of a MM-QFA*

The question that now arises is: are MM-QFA always more efficient in terms of space, with respect to the classical automata? In [1], a theorem answers to this question:

Theorem 42. *Let $L_m = (xy|xy)^m \cup \{(xy|zy)^i xx \mid 0 \leq i \leq m-1\}$. Then*

1. *L_m can be recognized by a 1-way deterministic automaton with $3m + 2$ states*
2. *L_m can be recognized by a 1-way reversible automaton but requires at least $3(2^m - 1)$ states.*

7

1-Way General QFA

In 2012 more questions about the power of MO-QFA and MM-QFA were made and in [8] two new formalism were added to try answer those questions. The paper starts from the fact that the languages recognized by MM-QFA with bounded error are more than those recognized by MO-QFA, but still a proper subset of regular languages. From the study on MO-QFA and MM-QFA, two observation are then made:

1. the number of times the measurement is performed in the computation affects the computational power of 1 Way QFA (Either MO or Measure Many)
2. by considering just unitary transformations one limits the computational power of 1 Way QFA in such a way that the two typical models of 1 Way QFA (MO-QFA and MM-QFA) are less powerful than their classical counterparts.

Trying to overcome those problems, in [8] all the formalisms introduced does not use unitary operators as evolution operators, but the most general operator allowed in quantum mechanics is used: trace-preserving quantum operations.

7.1 MO-1gQFA

In [8] a MO-1gQFA is defined a 5-tuple $\mathcal{M} = (\mathcal{H}, \Sigma, \rho_0, \{\mathcal{E}_\sigma\}_{\sigma\in\Sigma}\}, P_{\text{acc}})$ where:

- $\mathcal{H}$ is a finite Hilbert Space (the space of the states)
- Σ is the working alphabet
- ρ_0 is the initial state and it's described as a Density operator on $\mathcal{H}$
- $\mathcal{E}_\sigma$ (fixed a $\sigma \in \Sigma$) it a trace-preserving quantum operator acting on $\mathcal{H}$
- P_{acc} is a projector on the subspace called *accepting subspace of H*

Denote $P_{\text{rej}} = I - P_{\text{acc}}$ and $\{P_{\text{acc}}, P_{\text{rej}}\}$ form a projective measurament on $\mathcal{H}$.

On a generic word $\sigma = \sigma_1\sigma_2\ldots\sigma_n(\sigma_i \in \Sigma \quad \forall i \in 1, 2, \ldots n)$ the automata $\mathcal{M}$ acts as follows: the operators $\mathcal{E}_{\sigma_i}$ are permormed on the state ρ_0 in succession. When the word σ has been completely read, the projective measurement $\{P_{\mathrm{acc}}, P_{\mathrm{rej}}\}$ on the current state of the machine obtaining an accepting result with some probability. So, as just described, the automaton $\mathcal{M}$ induces a function $f_{\mathcal{M}}(\sigma) : \Sigma^{*} - > [0,1]$ defined as:

$$f_{\mathcal{M}}(\sigma) = Tr(P_{\mathrm{acc}}\mathcal{E}_{\sigma_n} \circ \cdots \circ \mathcal{E}_{\sigma_2} \circ \mathcal{E}_{\sigma_1}\rho_0)$$

From this definition, we can see that $f_{\mathcal{M}}(\sigma)$ is actually the function describing the probability for some word $\sigma \in \Sigma^{*}$ to be accepted from $\mathcal{M}$.

The first results we can give about this new formalism concern the function induced by MO-1gQFAs:

- If f is a function induced by an MO-1gQFA, then $1 - f$ is also induced by an MO-1gQFA
- If $f_1, f_2, \ldots, f_k$ are functions induced by MO-1gQFA, then $\sum_{i=1}^{k} c_i f_i$ is also induced by an MO-1gQFA for any real constants $c_i > 0$ such that $\sum_{i=1}^{k} c_i = 1$
- If $f_1, f_2, \ldots, f_k$ are functions induced by MO-1gQFA, then $f_1 f_2 \ldots f_k$ defined as $f_1 f_2(w) = f_1(w) f_2(w)$ is also induced by an MO-1gQFA

About Language acceptance for MO-1gQFA, we have:

Definition 7.1.1. *A language L is said to be recognized by a MO-1gQFA $\mathcal{M}$ with bounded error $\epsilon > 0$ if for some $\lambda \in (0,1]$ it holds that $f_{\mathcal{M}}(x) \geq \lambda + \epsilon$ for all $x \in L$ and that $f_{\mathcal{M}}(x) \leq \lambda - \epsilon$ for all $x \notin L$*

So, if we resamble the reason why MO-1gQFA where introduced, it was beacuse it was assumed that using unitary operator on classic MO-QFA could limit their accepting power; actually, even using trace-preserving operator, the result obtained is that:

Theorem 43. *The class of languages recognized with bounded error by MO-1gQFAs is exactly the class of regular languages*

Moreover, a relation with the PRFA can be introduced:

Theorem 44. *For every regular language recognized by PFA, there exists a MO-1gQFA recognizing it with certainty*

Furthermore, it has been proved that:

Theorem 45. *MO-1gQFA recognize all regular languages with certainty.*

7.2 MM-1gQFA

The MM-1gQFA formalism introduced in [8] is pretty similar to the one used in 4.2 but, as for the MM-1gQFA, the evolution operator does not have the limitation to be unitary but they can be any quantum trace-preserving operator. So, a generic MM-1gQFA is defined as $\mathcal{M} = (\mathcal{H}, \Sigma, \rho_0, \{\mathcal{E}_\sigma\}_{\sigma\in\Sigma}\}, \mathcal{H}_{\mathrm{acc}}, \mathcal{H}_{\mathrm{rej}})$ where:

- $\mathcal{H}$ is a finite Hilbert Space (the space of the states)
- Σ, united with the symbols $\{\kappa, \$\}$ is the working alphabet

- ρ_0 is the initial state and it's described as a Density operator on $\mathcal{H}$
- $\mathcal{E}_\sigma$ (fixed a $\sigma \in \Sigma$) it a trace-preserving quantum operator acting on $\mathcal{H}$
- $\mathcal{H}_{\text{acc}}$ is the *accepting subspace of H*
- $\mathcal{H}_{\text{rej}}$ is the *rejecting subspace of H*

Note that, together with a subspace $\mathcal{H}_{\text{non}}$, the spaces $\mathcal{H}_{\text{acc}}$, $\mathcal{H}_{\text{rej}}$ span the full $\mathcal{H}$, namely $\mathcal{H} = \mathcal{H}_{\text{acc}} \oplus \mathcal{H}_{\text{rej}} \oplus \mathcal{H}_{\text{non}}$. Moreover, there is a measurement $\{P_{\text{acc}}, P_{\text{rej}}, P_{\text{non}}\}$ where the general element P_i is the projector on the subspace $\mathcal{H}_i$ ($i \in \{\text{acc, rej, non}\}$).

The input string of a generic MM-1gQFA $\mathcal{M}$ has the form $\kappa x\$$ (the symbols $\kappa, \$$ are the left and right endmarker respectively). Then, the behaviour of the machine is pretty the same as the one we introduced in subsection 4.2:

1. If we assume the read symbol is σ, then $\mathcal{E}_\sigma$ is applied to the current state ρ, obtaining $\rho' = \mathcal{E}_\sigma(\rho)$
2. Then, the measurement $\{P_{\text{acc}}, P_{\text{rej}}, P_{\text{non}}\}$ is performed on ρ'. If an accepting / rejecting result is observed, then $\mathcal{M}$ halts return the observed result; otherwise, with a probability $Tr(P_{\text{non}}\rho')$ the machine reads the next symbol.

The generic state of a MM-1gQFA $\mathcal{M}$ can be described as an element inside the set $\mathcal{V} = \mathcal{H} \times \mathbb{R} \times \mathbb{R}$. Given an element $(\rho, p_{\text{acc}}, p_{\text{rej}}) \in \mathcal{V}$, the machine $\mathcal{M}$ has an accepting probability p_{acc}, a rejecting probability p_{rej} and neither with probability $Tr(\rho)$. So, the evolution of $\mathcal{M}$ as described above can be formalized as an operator $\mathcal{T}_\sigma : \mathcal{V} \to \mathcal{V}$ (after reading a symbol $\sigma \in \Sigma \cup \{\kappa, \$\}$), defined as:

$$\mathcal{T}_\sigma : (\rho, p_{\text{acc}}, p_{\text{rej}}) \to (P_{\text{non}}\mathcal{E}_\sigma(\rho)P_{\text{non}}, \text{Tr}(P_{\text{acc}}\mathcal{E}_\sigma(\rho)) + p_{\text{acc}}, \text{Tr}(P_{\text{rej}}\mathcal{E}_\sigma(\rho)) + p_{\text{rej}})$$

If we use as for MO-1gQFA the function $F_{\mathcal{M}}(x)$ to describe the accepting probability for a word x, then it accumulates all the accepting probabilities produced on reading each symbol in the input string $\kappa x\$$.

It is known that MM-QFA can recognize more languages with bounded error than MO-QFA. From this fact, we tend to believe that the number of times of the measurement performed in the computation affects the computational power of QFA. Encouraged by this belief, the MM-1gQFA was defined, a measure-many version of MO-QFA, with hope to enhance the computational power of 1gQFA. However, the result obtained is that:

Theorem 46. *The set of Languages recognized by MM-1gQFA with bounded error is exactly the class of regular languages*

8 Other models: Latvian and with control language MM-QFA

We will give here the definition of two more formalisms for MM-QFA: the Latvian and the one with control language

8.1 Latvian QFA

It works similarly as MO-QFA except that the transition function δ, is a combination of projective measurement and unitary matrix. This alteration increases the power of LQFA (Latvian QFA) for acceptance of languages.

Definition 8.1.1. *A LQFA M is defined as a 6-tuple* $(Q, \Sigma, \{A_\sigma\}_{\sigma\in\Sigma}, \{P_\sigma\}_{\sigma\in\Sigma}, q_0, Q_{acc}, Q_{acc})$ *where:*

- *Q is the set of states*
- *Σ is the alphabet. As usual, we can define the working alphabet $\Gamma = \Sigma \cup \{\#, \$\}$.*
- *A_σ is the unitary transformation associated to each symbol and P_σ is a set of ortoghonal subspaces*
- *q_0 is the initial state*
- *Q_{acc} is the set of accepting states. We can then define $E_{acc} = span\{|q\rangle \, | q \in Q_{acc}\}$*
- *Q_{rej} is the set of rejecting states. We can then define $E_{rej} = span\{|q\rangle \, | q \in Q_{rej}\}$*

The computing process of a LQFA on a word $\sigma = \#\sigma_1\sigma_2 \ldots \sigma_n\$$, starts in the state q_0 and for each $\sigma_i \in \Sigma$ the unitary matrix A_{σ_i} and the projective measurement P_{σ_i} are performed. When the right end marker is found, \$, the projective measurement $P_\$$ gives the result that states if the word σ is accepted or rejected.

It has been proved that LQFA accepts, with bounded error, a proper subset of regular languages.

8.2 QFA with control language

Definition 8.2.1. *A CL-1QFA M is defined as a 6-tuple*
$M = (Q, \Sigma, \pi, \{U(\sigma)\}_{\sigma \in \Sigma}, O, L)$ *where:*

- *Q is the finite set of states*
- *Σ is the alphabet, to which we will add also a right-end marker \$*
- *$\pi \in \mathbb{C}^{1 \times n}$ is the set of initial amplitude satisfying $\|\pi\|^2 = 1$. If $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and $q = (q_1, q_2, \ldots, q_n)$ then πq^T (where T stands for the transpose) we will get the initial state*

$$|\psi_0\rangle = \sum_{i=1}^{n} \pi_i |q_i\rangle$$

- *$U(\sigma) \in \mathbb{C}^{n \times n}$, $\sigma \in (\Sigma \cup \$)$ is the unitary matrix associated to the symbol σ*
- *O is an observable with set of possible results $C = \{c_1, c_2, \ldots, c_s\}$ and with the projector set $\{P(c_i)|\ i = 1, 2, \ldots, n\}$ where $P(c_i)$ represents the projector onto the eigenspace corresponding to c_i*
- *$L \subseteq C^*$ is a regular language called Control language*

The computation of input $\sigma = \sigma_1\sigma_2 \ldots \sigma_n\$$ starts with the state $|\psi_0\rangle$ we defined above and for each σ_i that is read the operations made are:

1. $U(\sigma_i)$ is applied with the new state $|\psi'\rangle = U(\sigma_i) |\psi\rangle$
2. The observable O is measured on $|\psi'\rangle$ and according to the principles of QM this leads to a result $c_k \in C$ with a probability $p_k = \|P(c_k) |\psi'\rangle \|^2$ and the states of the machine then collpases to $P(c_k) |\psi'\rangle / \sqrt{p_k}$.

Thus, the computation on the word $\sigma = \sigma_1\sigma_2 \ldots \sigma_n\$$ leads to a sequence $y = y_1 y_2 \ldots y_{n+1} \in C^*$ with a probability $p(y|\sigma)$ computed as:

$$p(y|\sigma) = \left\| \prod_{i=1}^{n+1} (P(c_i)U(\sigma_i)) |\psi_0\rangle \right\|^2$$

where we define $\sigma_{n+1} = \$$ and hypotizing that the product $\prod_{i=1}^{n} A_i = A_n A_{n-1} \ldots A_1$ (remember the matrix rules on product) then the accepted language is defined as:

$$P_M(\sigma) = \sum_{y_1 y_2 \cdots y_{n+1} \in C^*} p(y_1 y_2 \ldots y_{n+1}|\sigma)$$

Further works on this model showed that the class of languages accepted by CL-1QFA with bounded error is closed under Boolean operations. Moreover, the relation between **RMO**, **RMM** as defined above and let **RQC** be the class of languages accepted by CL-1QFA with bounded error is:

$$\mathbf{RMO} \subset \mathbf{RMM} \subset \mathbf{RQC}$$

9

Overcoming the head movement limit

In this chapter we will introduce some formalisms that will be useful to define a conclusive language hierarchy; in this two last models, as for Turing machines, we are allowed to move right, stay put and eventually move left on the input tape after each symbol is read. This obviously increases the acceptance power of 1.5-Way QFA and 2-Way QFA.

9.1 1.5-Way QFA

In a 1.5-Way QFA R/W head is allowed to remain stationary or move towards the right direction of the input tape, but it cannot move towards the left of input tape. So, a generic 1.5-Way QFA $M = (Q, \Sigma, q_0, \delta, Q_{\text{acc}}, Q_{\text{rej}})$ where:

- Q is the finite set of states
- Σ is the input alphabet, united with the left and right endmarker $\{\kappa, \$\}$
- q_0 is the initial state
- δ is the transition function defined as $\delta : Q \times (\Sigma \cup \{\kappa, \$\}) \times Q \times \{0, 1\} \to \mathbb{C}$, where the last argument is used to defined the moviment of the reading head. (0 stands for a *stay* movement, while 1 is for *right* movement)
- Q_{acc} is the set of accepting states
- Q_{rej} is the set of rejecting states

9.2 2-Way QFA

As mentioned above, the 2-Way QFA formalism was introduced in [7]; from the name, it's easy to see that it allows the reading head to move both left and right on the input tape and eventually even act 'stationary' (no movement on the tape). A 2-Way QFA M is defined as $(Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ where:

- Q is the finite set of states. Moreover, it must be true that $Q = Q_{\text{acc}} \cup Q_{\text{rej}} \cup Q_{\text{non}}$, where Q_{non} is the the set of non-halting state ($Q_{\text{non}} = Q \setminus (Q_{\text{acc}} \cup Q_{\text{rej}})$)

- Σ is the input alphabet, united with the left and right endmarker $\{\kappa, \$\}$ (we will refer to this enhanced alphabet with Γ)
- q_0 is the initial state
- δ is the transition function defined as $\delta : Q \times (\Sigma \cup \{\kappa, \$\}) \times Q \times \{\leftarrow, \uparrow, \rightarrow\} \rightarrow \mathbb{C}$, where the last argument is used to defined the moviment of the reading head.
- Q_{acc} is the set of accepting states
- Q_{rej} is the set of rejecting states

Unlike the usual definition of 2-way automata, we will assume that the tape of any 2QFA in circular in the sense that if the machine is scannin the last tape square and subsequently moves its tape head right (or, in the same way, it is scanning the first tape square and moves left), the tape head will then be scanning the first tape square (last tape square, respectively). The content of any tape can be described by a mapping $x : \mathbb{Z}_n \rightarrow \Gamma$, with n being the the number of distinct tape squares in the tape; such a mapping will itself be referred to as a *tape.* The number of configurations of a 2QFA M on any tape x of length n is $n|Q|$, since there are n possible locations for the tape head and $|Q|$ internal states; we will denote the set of configuration for a fixed M with C_n, and identify $Q \times \mathbb{Z}_n$ in the obvious way.
A superposition of M on a tape x of length n is any norm 1 element of the finite Hilbert Space $\mathcal{H}_n = l_2(C_n)$ (the space of mappings from C_n to $\mathbb{C}$ with the usual inner product). For each $c \in C_n$, $|c\rangle$ denotes the unit vector which takes value 1 at c and 0 elsewhere: all the other elements inside $\mathcal{H}_n$ can be expressed with the usual superposition $\sum_{c \in C_n} \alpha_c |c\rangle$.

After this definitions, we can now introduce the transition function $\delta : Q \times \Gamma \times Q \times \{-1, 0, 1\} \rightarrow \mathbb{C}$, which has to be interpreted as the function that taken as inputs two states $q, q' \in Q$, a symbol $\sigma \in \Gamma$ and a direction $d \in \{-1, 0, 1\}$ (1 is a movement in right direction of the tape head, -1 for a left one and 0 is the *stay* direction) (written as $\delta(q, \sigma, q', d)$), returns the probability amplitude with which a machine in state q, reading in input symbol σ will change its state to q' and move the tape head in direction d. For any tape x, the transition function induces an operator U_δ^x (the *evolution operator*) on $\mathcal{H}_{|x|}$ as follows:

$$U_\delta^x |q, k\rangle = \sum_{q', d} \delta(q, x(k), q', d) \left|q', k + d(mod|x|)\right\rangle$$

for each element $c = (q, k) \in C_{|x|}$. The operator U_δ^x is extended to all $\mathcal{H}_{|x|}$ by linearity, so we are allowed to define $(U_\delta^x)^t |\psi\rangle$ as the superposition obtained if a machine M on tape x, starting from superposition $|\psi\rangle$ was left to run for t steps (without observing, otherwise the superposition would collapse).

Notice that the computation in $2-WayQFA$ is very similar to the one made in MM-QFA, since after every step the state is observed using the observable $\mathcal{O}$, which is defined as the observable corresponding to the decomposition $\mathcal{H}_n$ into $E_{\text{acc}} \oplus E_{\text{rej}} \oplus E_{\text{non}}$, where the three subspaces are defined as:

$$\begin{aligned}
E_{\text{acc}} &= span\{|c\rangle \mid c \in (Q_{\text{acc}} \times \mathbb{Z}_n)\} \\
E_{\text{rej}} &= span\{|c\rangle \mid c \in (Q_{\text{rej}} \times \mathbb{Z}_n)\} \\
E_{\text{non}} &= span\{|c\rangle \mid c \in (Q_{\text{non}} \times \mathbb{Z}_n)\}
\end{aligned}$$

which clearly recalls the definition given for MM-QFA, we just have to add the component for $\mathbb{Z}_n$.

9.2.1 Well-formed 2-Way QFA

In order for a superposition to be valid, it must be, as we already know, of unit form. A 2-Way QFA which guarantees that any valid superposition will evolve into another valid superposition is said to be *well-formed*. This requirement can be seen as the request for all the U_δ^x to be *unitary*; we will now prove a Proposition which describes the properties an evolution operator must have in order to be unitary (and, consequentially, the 2-Way QFA will be well-formed).

Theorem 47. *A 2-Way QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ is well-formed iff for every choice of $q_1, q_2 \in Q$ and $\sigma, \sigma_1, \sigma_2 \in \Gamma$, the 3 following holds:*

$$\sum_{q',d} \overline{\delta(q_1, \sigma, q', d)}\delta(q_2, \sigma, q', d) = \begin{cases} 1 \text{ if } q_1 = q_2 \\ 0 \text{ if } q_1 \neq q_2 \end{cases}$$

$$\sum_{q'} \left(\overline{\delta(q_1, \sigma_1, q', 1)}\delta(q_2, \sigma_2, q', 0) + \overline{\delta(q_1, \sigma_1, q', 0)}\delta(q_2, \sigma_2, q', -1) \right) = 0$$

$$\sum_{q'} \left(\overline{\delta(q_1, \sigma_1, q', 1)}\delta(q_2, \sigma_2, q', -1) \right) = 0$$

Proof. For each x, U_δ^x is unitary if and only if the vectors $U_\delta^x |q, k\rangle$ for $q \in Q, z \in \mathbb{Z}_{|v|}$ are ortonormal. The first condition is a 'Kronecker product', it requires that $\| U_\delta^x |q, k\rangle \| = 1$, $\forall q, k$, while if $q_1 \neq q_2$, then $U_\delta^x |q_1, k\rangle \perp U_\delta^x |q_2, k\rangle$. The second condition is equivalent to the requirement that $U_\delta^x |q_1, k\rangle \perp U_\delta^x |q_2, k+1\rangle$, while the third one ensures that $U_\delta^x |q_1, k\rangle \perp U_\delta^x |q_2, k+2\rangle$, for each tape x, states q_1, q_2 and position k. Thus, the vectors $U_\delta^x |q, k\rangle$, with $q \in Q$ and $k \in \mathbb{Z}_{|x|}$ are orthonormal for every x only if all three conditions are satisfied. □

The above theorem provides an easy way to check if a 2-Way QFA is well-formed or not. However, there exists an easier method to specify the transition function: the method is to decompose the transition function δ into two parts: one for transforming the state and the other which takes care of moving the tape head. Given a 2-Way QFA M with set of states Q, consider the Hilbert space $l_2(Q)$ and suppose to have a linear operator $V_\sigma : l_2(Q) \to l_2(Q)$ defined for each $\sigma \in \Gamma$ and a function $D : Q \to \{-1, 0, 1\}$. Using these two functions we can define the transition function δ as:

$$\delta(q, \sigma, q', d) = \begin{cases} \langle q' | V_\sigma | q \rangle & D(q') = d \\ 0 & D(q') \neq d \end{cases} \tag{9.1}$$

Where $\langle q' | V_\sigma | q \rangle$ is the coefficient of $|q'\rangle$ in the superposition generated by $V_\sigma(|q\rangle)$: M is *well-formed* when every V_σ is unitary.

9.2.2 Language Recognized by a 2-Way QFA

We can now discuss the concept of *language accepted* by a 2-Way QFA. Given an inpu string $w \in \Sigma^*$ we define the corresponding tape x_w as follows:

$$x_w(i) = \begin{cases} \kappa & \text{if } i = 0 \\ \$ & \text{if } i = |w| + 1 \\ w_i & \text{if } 1 \leq i \leq w \end{cases}$$

Of course x_w is not defined for $i > |w| + 1$. Let M be a 2-Way QFA, we will say that M *runs on input* w if:

1. the tape of M is defined by x_w
2. The computation begins with M in the state $|q_0, 0\rangle$
3. after each computation step (after each application of U_δ^x), the reached superpostion is observed used the observable $\mathcal{O}$. As for MM-QFA the computation continues until the result of an observation is either *accept* or *reject*, at which time the computation halts.

After this definitions, we can treate the computation of a 2-Way QFA as a for a probabilistic machine: if input w results in *accept* with probability greater than $\frac{1}{2}$, then w is an element of the language recognized by M, otherwise not.

9.2.3 A pratical example

To clarify a bit the notation which may look a bit messy, we will now provide an example of 2-Way QFA which recognizes the language a^*b^*. Let M be a 2-Way QFA defined as follows:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- $Q_{\text{acc}} = \{q_3\}$
- $Q_{\text{rej}} = \{q_4\}$

We will define the δ function using the method described in 9.1. Since no superposition are needed in this simple automaton, we will provide a table where we will put in the rows the current state q_i, in the columns the input symbol σ and then in the table cell the new state q_j such that $V_\sigma(q_i) = q_j$.

	κ	a	b	$\$$
q_0	q_0	q_0	q_1	q_1
q_1	q_2	q_2	q_0	q_0
q_2	q_4	q_4	q_2	q_3
q_3	q_3	q_3	q_3	q_2
q_4	q_1	q_1	q_4	q_4

The function D is then defined as

$$\begin{aligned} D(q_0) &= D(q_2) = +1 \\ D(q_3) &= D(q_4) = 0 \\ D(q_1) &= -1 \end{aligned}$$

Consider $w = aba$ as input string, then the tape x is defined as: $x(0) = \kappa, x(1) = a, x(2) = b, x(3) = a, x(4) = \$$. Then the sequence of steps done by M is:

1. The initial state is $|q_0, 0\rangle$. M reads the input symbol $x(0) = \kappa$ and the new state becomes $V_\kappa |q_0\rangle = |q_0\rangle$; We compute D on the new state, and we get $+1$. Then the new state is $|q_0, 1\rangle$.

2. The input symbol is now a. $V_a |q_0\rangle$ is equal to $|q_0\rangle$ and $D(q_0)$ is again $+1$. So we move to state $|q_0, 2\rangle$.

3. We now read the symbol b; in this case, by definition of V, $V_b |q_0\rangle$ is equal to $|q_1\rangle$ and since $D(|q_1\rangle) = -1$, the new state is $|q_1, 1\rangle$

4. Since we 'got back' to symbol in position one of the tape due to the definition of D, the machine reads again the input $x(1) = a$. The new state then becomes $V_a |q_1\rangle = |q_2\rangle$ while the movement of the tape head is $D(|q_2\rangle) = +1$; so the current state becomes $|q_2, 2\rangle$

5. M then reads again $x(2) = b$ while being in state $|q_2, 2\rangle$: it then moves to state $|V_b |q_2\rangle, D(V_b |q_2\rangle)\rangle = |q_2, 3\rangle$

6. The current input then becomes $x(3) = a$, which causes the machine to move into the state $|q_4, 4\rangle$

After this last step, the machine M halts with certainty with a *reject* result: in fact, after each symbol is read, we are also supposed to apply the observable $\mathcal{O}$ to the state of the machine; but, from step 1 to step 4, we were always in states that if measured, would yield to a non-halting state with certainty (so the state is not modified by the application of $\mathcal{O}$). But in step 5, we reach a state $|q_4, 4\rangle$, which is observed using $\mathcal{O}$ and leads to rejection with certainty.

We can use this example to introdce a key concept about the description of V: many of the values of V define transitions which are never encountered in a computation for any string w. These values have been defined in such a way that V_σ is unitary; in general, we are intrested in specify only the values that matter and as long as the vectors are orhonormal, the remaining values can always be assigned in arbitrary fashion so that the resulting V is unitary.

9.2.4 2-Way QFA for $\{a^m b^m \mid m \geq 1\}$

We will now build a 2-Way QFA that recognizes the non-regular language $L = \{a^i b^i \mid m \geq 1\}$ in linear time. For each $N \in \mathbb{N}^+$, define the machine $M_N = (Q, \Sigma, \delta, q_0, Q_{\text{acc}}, Q_{\text{rej}})$ where:

- $\Sigma = \{a, b\}$

- $$\begin{aligned} Q = &\{q_0, q_1, q_2, q_3\} \\ &\cup \{r_{j,k} \mid 1 \le j \le N, 0 \le k \le max(j, N-j+1)\} \\ &\cup \{s_j \mid 1 \le j \le N\} \end{aligned}$$

- $Q_{\text{acc}} = \{s_N\}$

- $Q_{\text{rej}} = \{q_3\} \cup \{s_j \mid 1 \le j < N\}$

The delta function is described using the easier method we introduced above (so giving the definition of V and D); we first introduce V:

$V_\kappa \lvert q_0\rangle = \lvert q_0\rangle$	$V_\$ \lvert q_0\rangle = \lvert q_3\rangle$
$V_\kappa \lvert q_1\rangle = \lvert q_3\rangle$	$V_\$ \lvert q_2\rangle = \frac{1}{\sqrt{N}} \sum_{j=i}^{N} \lvert r_{j,0}\rangle$
$V_\kappa \lvert r_{j,0}\rangle = \frac{1}{\sqrt{N}} \sum_{l=1}^{N} \exp\left(\frac{2\pi i}{N} jl\right) \lvert s_l\rangle, 1 \le j \le N$	$V_b \lvert q_0\rangle = \lvert q_1\rangle$
$V_a \lvert q_0\rangle = \lvert q_0\rangle$	$V_b \lvert q_2\rangle = \lvert q_2\rangle$
$V_a \lvert q_1\rangle = \lvert q_2\rangle$	$V_b \lvert r_{j,0}\rangle = \lvert r_{j,N-j+1}\rangle, 1 \le j \le N$
$V_a \lvert q_2\rangle = \lvert q_3\rangle$	$V_b \lvert r_{j,k}\rangle = \lvert r_{j,k-1}\rangle, 1 \le k \le N-j+1, 1 \le j \le N$
$V_a \lvert r_{j,0}\rangle = \lvert r_{j,j}\rangle, 1 \le j \le N$	
$V_a \lvert r_{j,k}\rangle = \lvert r_{j,k-1}\rangle, 1 \le k \le j, 1 \le j \le N$	

and then the definition of D:

$D(q_0) = 1$	$D(r_{j,0}) = -1, 1 \le j \le N$
$D(q_1) = -1$	$D(r_{j,k}) = 0, 1 \le j \le N, k \ne 0$
$D(q_2) = +1$	$D(s_j) = 0, 1 \le j \le N$
$D(q_3) = 0$	

Theorem 48. *Let $w \in \{a,b\}^*$. For every $N \in \mathbb{N}^+$, if $w \in \{a^m b^m \mid m \ge 1\}$ then M_N accepts w with probability 1 and otherwise M_N rejects w with probability at least 1 - $\frac{1}{N}$. In both cases the machine M_N halts after $O(N|w|)$ steps.*

Proof. The computation of each M_N consists of two phases. The first phase rejects any input not of the form $a^u b^v$ for $u, v \ge 1$, and the second phase rejects, with some probability, those inputs for which $u \ne v$.

The first phase is really similar to the computation we showed in the previous example: for each word that is not of the form $a^u b^v$, the computation will eventually enter a state were the machine M rejects with probability 1 and it stops.

Otherwise the second phase begins with the machine in state q_2 with the tape reading the right end marker \$. At the start of the second phase, the computation branches into N paths, indicated by the states $r_{1,0}, r_{2,0}, \cdots r_{N,0}$ each with amplitude $\frac{1}{\sqrt{N}}$ and the tape head moves 1 square to the left for each state in the superposition (this because the symbol read is \$, the state is q_2, so we apply $V_\$(\lvert q_2\rangle)$; for what concerns the tape head movement, we move to the left since $D(r_{j,0}) = -1$, for all the js, so all the states of the superposition are effected). For each of these paths, the tape head moves deterministically to the left end-marker in the following way; in the j-th path

- if the tape head reads the symbol a, it remains stationary for j steps (always reading the same a) and then moves left. This happens because when the a is first read, V_a maps the current state $|r_{j,0}\rangle$ to $|r_{j,j}\rangle$ and the function D for $r_{j,j}$ is defined as 0 (j, j is a special case of j, k in the definition of D). From that point on, the machine M_N keeps on reading the same a and it keeps applying the rule of V_a that maps $|r_{j,k}\rangle$ to $|r_{j,k-1}\rangle$ (and keeping being stationary on the same a because of D) until we reach the state $|r_{j,1}\rangle$. When the a is read for the jth time, we reach the state $|r_{j,0}\rangle$ and using $D(r_{j,0})$ we move the tape head one char to the left. And this goes on for all the as

- if the tape head reads the symbol b, it remains stationary for $N - j - 1$ steps and then it moves left. The explanation for this behavior is the same as the a case, just notice that V_b maps $|r_{j,0}\rangle$ to $|r_{j,N-j-1}\rangle$ and then proceed as above.

Thus, on input $a^u b^v$, the tape head requires precisely $(j + 1)u + (N - j + 2)v + 1$ steps to move from the right end marker \$ to the left one κ; notice that when the left end marker is reached, each j-th path is in the state $|r_{j,0}\rangle$ (Because while getting from right to left, we always reach only $r_{j,k}$ states, and the only one allowing us a left movement is $r_{j,0}$). Moreover, since all js in the superposition created from $V_\$$ (applied to $|q_2\rangle$ at the beginning) are different, we have that

$$(j_1 + 1)u + (N - j_1 + 2)v + 1 = (j_2 + 1)u + (N - j_2 + 2)v + 1,\ j_1, j_2 \in \{1, 2, \cdots N\}, j_1 \neq j_2$$

holds iff $u = v$. Namely, any two different (so with two different js) computational paths started from $V_\$(|q_2\rangle)$ will reach the left end marker at the same time iff $u = v$ (so the input is of the form $a^m b^m$ with $m = u = v$). Consider now what happens when all the N computational paths reaches the left end marker: the operator $V_\kappa(|r_{j,0}\rangle)$ needs to be applied; we split in two cases:

1. in the first one, the input is of the form $a^m b^m$: from what we stated above, we know that in this case all the N paths reaches the left end marker at the same time. So here we have the state that we will call, for sake of readability, $|\psi\rangle$:

$$\frac{1}{\sqrt{N}} \sum_{j=1}^{N} |r_{j,0}\rangle$$

and if we apply to it the operator V_κ (which is actually a QFT) we get:

$$\begin{aligned} V_\kappa |\psi\rangle &= \frac{1}{\sqrt{N}} \sum_{j=1}^{N} V_\kappa |r_{j,0}\rangle \\ &= \frac{1}{N} \sum_{j=1}^{N} \sum_{l=1}^{N} \exp\left(\frac{2\pi i}{N} jl\right) |s_l\rangle \qquad = |s_N\rangle \end{aligned}$$

in which the observable $\mathcal{O}$ leads to acceptance with probability 1.

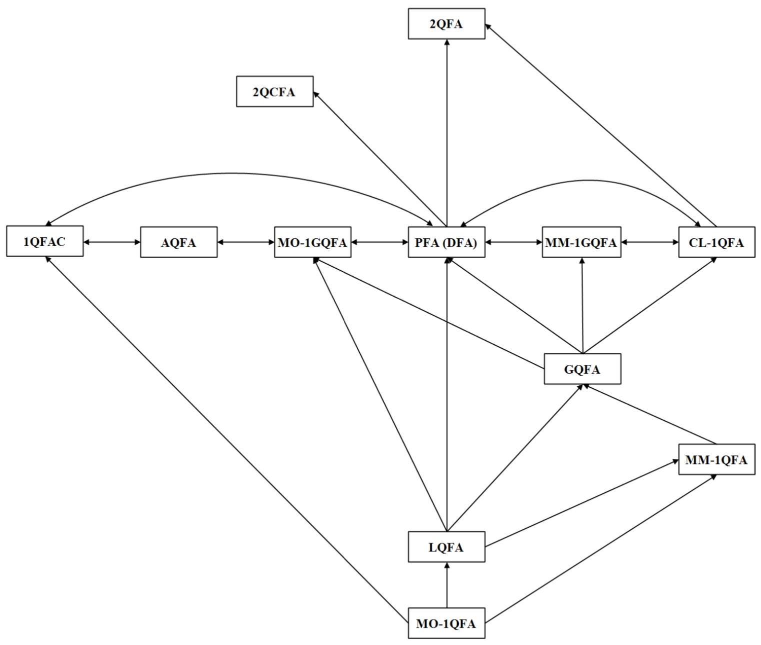


Figure 9.1: Inclusion Hierarchy

2. suppose now that the input is not of the form $a^m b^m$: then each of then N computation paths reaches the κ symbol at a differente time, and so there is no cancellation between the rejecting states introduced in the superposition by V_κ. For each of the N possible path lengths, the conditional probability that an observation results in accept at the the time corresponding to that path length is $\frac{1}{N}$ (the case that from the Fourier Transormation introduced by V_κ, if measured, gives as outcome the state $|s_N\rangle$). It follows that the total probability that an observation results in accept is also $\frac{1}{N}$, and consequently the input is rejected with prob 1 - $\frac{1}{N}$.

Each computation path has length that can be bounded by $O(N|w|)$ (in the equation for the number of steps to get from the right to the left endmarker, just notice that both $j+1$ and $N-j+2$ can be over estimated as $O(N)$, then we get $O(N)u + O(N)v = O(N)(u+v) = O(N|w|)$ since $w = a^u b^v$), so M_N must halt after $O(N|w|)$ steps.

□

9.3 A final recap

In the image 9.1 we introduce a language acceptance hierarchy: notice that a one-directional line shows containment relation and bidirectional lines show equivalence relation

10

Heisenberg-inspired Quantum Automata

We will now introduce a new family of automata, which will resul in an original contribution of this thesis.

Without going to deeply on physics matter, the most widely used mathematical description of Quantum Mechanics is also known as *Schrödinger picture*: in this version, the state vectors are time-dependent while the operators remain unmuted; the evolution in time of the system is then described using the Schrödinger equation. But, for QM there exists another *picture* known as *Heisenberg picture* in which the state vectors are time-indipendent and always remain fixed to their value at time 0, while the time-dependency is *moved* to the operators.
In terms of Quantum Finite State automata all the models use the first picture, where the initial state evolves through time using some unitary, but these unitaries never changes. Some work has been done for Quantum Cellular Automata, the quantum counterpart of Cellular Automata introduced by Von Neumann, where the equivalence between Schrödinger picture - based model and Heisenberg picture - based model has been proved in [**?**].

To exploit the Heisenberg picture, we formalize a new model for QFA, named *Quantum Heisenberg Picture Finite State Automata (QHFA)* defined as follow;

Definition 10.0.1 (Quantum Heisenberg Picture Finite State Automata)**.** *A QHFA M is 6-tuple $M = (Q, \Sigma, \psi, \{U_\sigma\}_{\sigma\in\Sigma}, \{E_{\sigma,\sigma'}\}_{\sigma,\sigma'\in\Sigma}, P)$ where:*

- *Q is a finite set of states, with $|Q| = m$*
- *Σ is the input alphabet*
- *ψ is the initial state of the Automata*
- *$\{U_\sigma\}_{\sigma\in\Sigma}$ is a set of unitaries ($U_\sigma \in \mathbb{C}^{m\times m}$), namely one for each $\sigma \in \Sigma$, which describes the initial amplitudes for the edges labelled with σ (using a graphic point of view).*
- *$\{E_\sigma^{\sigma'}\}_{\sigma,\sigma'\in\Sigma}$ is a set of $|\Sigma|^2$ unitaries. Let $\alpha, \beta \in \Sigma$, then $E_\alpha^\beta \in \mathbb{C}^{m\times m}$ is the unitary describing how the current unitary for α has to change when the input symbol is β.*
- *P is a projection matrix.*

10.1 Computation in QHFA

Let M be a QHFA as just defined and $x = x_1 x_2 \cdots x_n$ be an input string for the automata. Then, given $\sigma \in \Sigma$ we define:

$$\begin{aligned} U_\sigma^\epsilon &= U_\sigma \\ U_\sigma^{x_1} &= E_\sigma^{x_1} U_\sigma^\epsilon \\ U_\sigma^{x_i x_{i-1} \cdots x_1} &= E_\sigma^{x_i} U_\sigma^{x_{i-1} x_{i-2} \cdots x_1} \; \forall i \geq 2 \end{aligned}$$

A generic matrix U_σ^y, with $y \in \Sigma^*$ can be interpreted as the evolution of σ updated knowing that up to now the string y has been read. The idea is that at each step of computation we *reshape* the probability amplitudes in U_σ according to the input read, as if the automata *responds* to the input and adapts itself to it to get an higher accepting probability.

Let $x_i (i \geq 2)$ (the other cases are trivial) be the current input char, the operation we perform is:

$$U_\sigma^{x_i x_{i-1} \cdots x_1} = E_\sigma^{x_i} U_\sigma^{x_{i-1} x_{i-2} \cdots x_1} \;, \; \forall \sigma \in \Sigma$$

Then, when the last char x_n is read, the accepting probability is defined as:

$$\| P \hat{U}_x \left| \psi \right\rangle \|^2$$

For what concerns the definition of U_x we will provide three possible definitions and study the different expressive power the model gets whenever one of the three is used.

The three definitions are:

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} \tag{10.1}$$

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_{n-1}}^{x_{n-1} x_{n-2} x_{n-3} \cdots x_1} \ldots U_{x_i}^{x_i x_{i-1} x_{i-2} \cdots x_1} \ldots U_{x_2}^{x_2 x_1} U_{x_1}^{x_1} \tag{10.2}$$

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_{n-1}}^{x_n x_{n-1} x_{n-2} \cdots x_1} \ldots U_{x_i}^{x_n x_{n-1} x_{n-2} \cdots x_1} \ldots U_{x_2}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_1}^{x_n x_{n-1} x_{n-2} \cdots x_1} \tag{10.3}$$

The acceptance criterion we will use is the one we defined as *with cut-point.*

10.2 Expressive power

We will now further analyze how the expressive power of QHFAs change accordingly to the definition of $\hat{U}_x$ we use.

10.2.1 First case

The first definition for $\hat{U}_x$ namely:

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1}$$

seems to be the most promising.

Theorem 49. *Any MO-QFA M can be simulated by a QHFA $\mathcal{M}$*

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a MO-QFA (together with a projector P) and $x = x_1 x_2 \cdots x_n$ an input string. We will denote with $\overline{U}_\sigma$ the unitary description of δ. The computation for x in M produces a state that is:

$$\overline{U}_x |q_0\rangle = \overline{U}_{x_n} \overline{U}_{x_{n-1}} \cdots \overline{U}_{x_1} |q_0\rangle$$

We now define $\mathcal{M} = (Q, \Sigma, \psi, \{U_\alpha\}_{\alpha\in\Sigma}, \{E_\alpha^\beta\}_{\alpha,\beta\in\Sigma}, P)$ as follows:

- Q, Σ and P are the same as M
- $\psi = q_0$
- $U_\alpha = I_m \;\; \forall \alpha \in \Sigma$. (as before, $m = |Q|$)
- $E_\alpha^\beta = \overline{U}_\beta \; \forall \alpha, \beta \in \Sigma$.

Suppose $x \in \Sigma^*$ is given as input to $\mathcal{M}$: by definition of QHFA, the computation yields to a matrix $\hat{U}_x$ that is then applied to the initial state $|\psi\rangle$

$$\hat{U}_x |\psi\rangle = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} |\psi\rangle$$

Using the definition we can rewrite the last equation as:

$$\hat{U}_x |\psi\rangle = E_{x_n}^{x_n} E_{x_n}^{x_{n-1}} E_{x_n}^{x_{n-2}} \cdots E_{x_n}^{x_2} E_{x_n}^{x_2} E_{x_n}^{x_1} U_{x_n}^{\epsilon} |\psi\rangle$$

which yields by definition to:

$$\begin{aligned}
\hat{U}_x |\psi\rangle &= E_{x_n}^{x_n} E_{x_n}^{x_{n-1}} E_{x_n}^{x_{n-2}} \cdots E_{x_n}^{x_2} E_{x_n}^{x_2} E_{x_n}^{x_1} U_{x_n}^{\epsilon} |\psi\rangle \\
&= \overline{U}_{x_n} \overline{U}_{x_{n-1}} \overline{U}_{x_{n-2}} \cdots \overline{U}_{x_2} \overline{U}_{x_1} I_m |\psi\rangle \\
&= \overline{U}_{x_n} \overline{U}_{x_{n-1}} \overline{U}_{x_{n-2}} \cdots \overline{U}_{x_2} \overline{U}_{x_1} |q_0\rangle
\end{aligned}$$

that is exactly the equation of 'acceptance' for M. □

To prove that with this definition the expressive power of QHFAs is bigger than MO-QFA we will use as a witness the language:

$$L = \{a, b\}^* a$$

Firstly we will prove that this language cannot be accepted by a MO-QFA with any cutpoint.

Theorem 50. *Any MO-QFA cannot accept the language $L = \{a, b\}^* a$ with cut-point.*

Proof. Let's suppose the theorem is false. Then there exists an MO-QFA M that accepts L. Then using theorem 17 it must hold that for each $x \in \Sigma^*$ and for every $w \in L$ there exists a positive integer v such that $wx^v \in L$. But this is an absurd since for example taking $x = b$ and $w = a$, then $wx^v = ab^v$ and therefore it will never be in L. □

Theorem 51. *There exists a QHFA $\mathcal{M}$ that accepts $L = \{a, b\}^* a$ with certainty.*

Proof. Let $\mathcal{M}$ be a QHFA defined as follows:

- $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$ and $P = |q_1\rangle \langle q_1|$
- $psi = q_0$
- $U_a = X$, $U_b = I_2$ (where X is the matrix for the X gate).
- $E_{\sigma}^{\sigma'} = I_2 \ \forall \sigma, \sigma' in \Sigma$

where it must hold that $|q_1\rangle = X |q_0\rangle$. It is easy to see that for any $x = x_1 x_2 x_3 \cdots x_n$ used as input, the matrix $\hat{U}_x$ will be

$$\begin{aligned}\hat{U}_x &= E_{x_n}^{x_n} E_{x_n}^{x_{n-1}} E_{x_n}^{x_{n-2}} \cdots E_{x_n}^{x_2} E_{x_n}^{x_2} E_{x_n}^{x_1} U_{x_n}^{\epsilon} \\ &= U_{x_n}^{\epsilon} = U_{x_n}\end{aligned}$$

So, since $\Sigma = \{a, b\}$ then x_n can be either a or b. Therefore, in the case $x_n = a$ the computation yields to:

$$\| P \hat{U}_x |\psi\rangle \|^2 = \| P U_a |\psi\rangle \|^2 = \| P X |q_0\rangle \|^2 = \| \, |q_1\rangle \langle q_1 | q_1\rangle \|^2 = 1$$

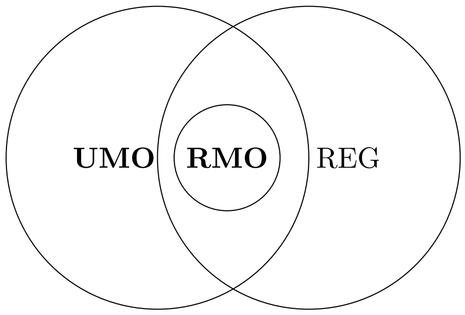
Otherwise, in the case $x_n = b$ the computation yields to:

$$\| P \hat{U}_x |\psi\rangle \|^2 = \| P U_b |\psi\rangle \|^2 = \| P I_2 |q_0\rangle \|^2 = \| \, |q_0\rangle \|^2 = 0$$

So this automaton not only recognizes with cut-point, but with certainty. □

From this theorem we can then conclude that the class of languages accepted by QHFA with the first definition of $\hat{U}_x$ is a proper superset of the class **UMO**.

The current relation between **UMO**, **RMO**, and regular languages is shown below.



The question that arises from the result we just got about QHFA is if the set of languages they recognize contains all the regular languages or not. In the first case then QHFAs would create a sort of a 'box' around the above diagram. In the second case QHFAs would accept a set of languages that is clearly bigger than **UMO** but that does not contain all regular languages.

10.2.2 Second case

We will now try to do the same thing we just did for the first defition of $\hat{U}_x$ with the second one, namely:

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_{n-1}}^{x_{n-1} x_{n-2} x_{n-3} \cdots x_1} \cdots U_{x_i}^{x_i x_{i-1} x_{i-2} \cdots x_1} \cdots U_{x_2}^{x_2 x_1} U_{x_1}^{x_1}$$

Theorem 52. *Any MO-QFA M can be simulated by a QHFA $\mathcal{M}$*

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a MO-QFA (together with a projector P) and $x = x_1 x_2 \cdots x_n$ an input string. We will denote with $\overline{U}_\sigma$ the unitary description of δ. The computation for x in M produces a state that is:

$$\overline{U}_x \left|q_0\right\rangle = \overline{U}_{x_n} \overline{U}_{x_{n-1}} \cdots \overline{U}_{x_1} \left|q_0\right\rangle$$

We now define $\mathcal{M} = (Q, \Sigma, \psi, \{U_\alpha\}_{\alpha \in \Sigma}, \{E_\alpha^\beta\}_{\alpha,\beta \in \Sigma}, P)$ as follows:

- Q, Σ and P are the same as M
- $\psi = q_0$
- $U_\alpha = \overline{U}_\alpha \; \forall \alpha \in \Sigma$.
- $E_\alpha^\beta = I_m \; \forall \alpha, \beta \in \Sigma$. (as before, $m = |Q|$)

Suppose x is given as input to $\mathcal{M}$: by definition of QHFA, the computation yields to a matrix $\hat{U}_x$ that is then applied to the initial state $\left|\psi\right\rangle$

$$\hat{U}_x \left|\psi\right\rangle = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_{n-1}}^{x_{n-1} x_{n-2} x_{n-3} \cdots x_1} \cdots U_{x_i}^{x_i x_{i-1} x_{i-2} \cdots x_1} \cdots U_{x_2}^{x_2 x_1} U_{x_1}^{x_1} \left|\psi\right\rangle$$

and, expanding all the terms, this is equal to:

$$E_{x_n}^{x_n} E_{x_n}^{x_{n-1}} \cdots E_{x_n}^{x_1} U_{x_n}^{\epsilon} E_{x_{n-1}}^{x_{n-1}} E_{x_{n-1}}^{x_{n-2}} \cdots E_{x_{n-1}}^{x_1} U_{x_{n-1}}^{\epsilon} \cdots E_{x_2}^{x_2} E_{x_2}^{x_1} U_{x_2}^{\epsilon} E_{x_1}^{x_1} U_{x_1}^{\epsilon} \left|\psi\right\rangle$$

Moreover applying QHFA's defition and the definition of $\mathcal{M}$, we know that $U_\sigma^\epsilon = U_\sigma = \overline{U}_\sigma$. Thereferore we can rewrite the above equation as:

$$\underbrace{I\,I \cdots I}_{n \text{ times}} \overline{U}_{x_n} \underbrace{I\,I \cdots I}_{n-1 \text{ times}} \overline{U}_{x_{n-1}} \cdots \underbrace{\;I\;}_{2 \text{ times}} \overline{U}_{x_2} I \overline{U}_{x_1} \left|q_0\right\rangle$$

and it's trivial to see that the final state reached by $\mathcal{M}$ is the same as M; hence, by generality of x, we can state that any machine M can be simulated by a QHFA $\mathcal{M}$. □

For the other direction we tried to prove lemma 17 but without success since the unitary matrices involved do not allow to apply the same techniques as in the proof of 17. Thus it leaded us to try the other way round checking if maybe 17 does not hold in this second case. The toy example we tried to solve was for a language L such that:

$$a \in L$$
$$a(bc)^i \notin L \forall i \in \mathbb{N}^+$$

but without any success.

10.2.3 Third case

The last case is the following:

$$\hat{U}_x = U_{x_n}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_{n-1}}^{x_n x_{n-1} x_{n-2} \cdots x_1} \cdots U_{x_i}^{x_n x_{n-1} x_{n-2} \cdots x_1} \cdots U_{x_2}^{x_n x_{n-1} x_{n-2} \cdots x_1} U_{x_1}^{x_n x_{n-1} x_{n-2} \cdots x_1}$$

Theorem 53. *Any MO-QFA M can be simulated by a QHFA $\mathcal{M}$*

Proof. The proof is the same as in the last case, adding some $E_\sigma^{\sigma'}$ that are all mapped to the identity matrix. □

For this third case we tried to check if the same lemma as 17 holds. Trying to follow the same proof, we encoutered a situation which is pretty similar but not exactly the same as in 17. In particular, let's suppose $w = a \in L$ while $x = bc \in \Sigma^*$. When looking at the matrix $\hat{U}_w$, by definition we get:

$$\hat{U}_w = U_a^a = E_a^a U_a^\epsilon$$

While $\hat{U}_w x^i$ with $i \geq 1$ is :

$$\hat{U}_w x^i = U_c^{a(bc)^i} U_b^{a(bc)^i} U_c^{a(bc)^i} U_b^{a(bc)^i} \cdots U_c^{a(bc)^i} U_b^{a(bc)^i} U_a^{a(bc)^i}$$

if we write $M = U_c^{a(bc)^i} U_b^{a(bc)^i}$ then $\hat{U}_w x^i$ can be written as:

$$\underbrace{MMM \cdots M}_{i \text{ times}} U_a^{a(bc)^i} = M^i U_a^{a(bc)^i}$$

Going even deeper, we can see that $U_a^{a(bc)^i}$ can be written as:

$$U_a^{a(bc)^i} = E_a^c E_a^b E_a^c E_a^b \cdots E_a^c E_a^b E_a^a U_a^\epsilon = E_a^c E_a^b E_a^c E_a^b \cdots E_a^c E_a^b U_a^a$$

We can then set $M' = E_a^c E_a^b$ and rewriting the above statement as:

$$U_a^{a(bc)^i} = \underbrace{M'M'M' \cdots M'}_{i \text{ times}} U_a^a = (M')^i U_a^a$$

Therefore, $\hat{U}_w x^i$ can be written as:

$$\hat{U}_w x^i = M^i (M')^i U_a^a$$

Resambling the proof of 17, at some point we will have to do the following operation:

$$\hat{U}_w - \hat{U}_w x^i$$

which yields to:

$$\begin{aligned} \hat{U}_w - \hat{U}_w x^i &= U_a^a - M^i (M')^i U_a^a \\ &= (I - M^i (M')^i) U_a^a \end{aligned}$$

which does not allow us to apply 16 as in 17. We just used a toy example for sake of readability. The reader can easily check that this situation is generated by any choice of x and w that satisfy 17 preconditions. Therefore we will end this part by stating that further investigations have to be done. One of the main path follower will be to check if 16 holds also for matrices of the form $(I - M^i(M')^i)$.

11

Conclusions

As we saw, a lot of different formalisms were introduced during the last 50 years to study the quantum counterpart of the Finite state Automata. Each formalism brings advantages and disadvantages. In some cases the closure properties are easier to prove, while other proofs are harder. What should be by now clear to the reader is the following: until quantum automata are allowed to move only in one direction, they are less powerful than their classical counterparts. This is a clearly counterintuitive result. What seems to be a good candidate to overcome this limitation is the QHFA formalism we introduced (in particular with the first acceptance condition). Proving that any regular language can be accepted by a QHFA would demonstrate a larger expressive power than classical automata.

Bibliography

[1] Andris Ambainis and Rusins Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 332–341. IEEE, 1998.

[2] Andris Ambainis, Arnolds KĶikusts, and Māris Valdats. On the class of languages recognizable by 1-way quantum finite automata. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 75–86. Springer, 2001.

[3] Alberto Bertoni and Marco Carpentieri. Analogies and differences between quantum and stochastic automata. *Theoretical Computer Science*, 262(1-2):69–81, 2001.

[4] Maria Paola Bianchi, Carlo Mereghetti, and Beatrice Palano. Quantum finite automata: Advances on bertoni's ideas. *Theoretical Computer Science*, 664:39–53, 2017.

[5] Alex Brodsky and Nicholas Pippenger. Characterizations of 1-way quantum finite automata. *SIAM Journal on Computing*, 31(5):1456–1478, 2002.

[6] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[7] Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 66–75. IEEE, 1997.

[8] Lvzhou Li, Daowen Qiu, Xiangfu Zou, Lvjun Li, Lihua Wu, and Paulo Mateus. Characterizations of one-way general quantum finite automata. *Theoretical Computer Science*, 419:73–91, 2012.

[9] Carlo Mereghetti, Beatrice Palano, and Giovanni Pighizzini. Note on the succinctness of deterministic, nondeterministic, probabilistic and quantum finite automata. *RAIRO-Theoretical Informatics and Applications*, 35(5):477–490, 2001.

[10] Cristopher Moore and James P Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237(1-2):275–306, 2000.