# Computación cuántica: autómatas finitos con inputs y outputs
# Quantum computing: finite automata with inputs and outputs

TRABAJO DE FIN DE GRADO
Curso 2020/2021


UNIVERSIDAD
COMPLUTENSE
MADRID

FACULTAD DE INFORMÁTICA

DOBLE GRADO EN MATEMÁTICAS E INGENIERÍA INFORMÁTICA

Director: Manuel Núñez García

Javier Gallego Gutiérrez

**Resumen**

Este trabajo sirve como introducción a la computación cuántica mediante autómatas finitos. Además, presenta un nuevo marco para trabajar con un modelo de autómatas finitos cuánticos que distinga entre inputs y outputs. Se considera como inputs los símbolos de un alfabeto, que determinan transformaciones unitarias, y como outputs las mediciones realizadas tras leer una cadena. También se presenta una relación de implementación para estos autómatas. A continuación, se da otra definición alternativa más apropiada para un marco de testing de caja negra. Finalmente, se ha desarrollado un simulador de autómatas finitos cuánticos que permite simular el comportamiento de autómatas a partir de una configuración establecida por el usuario. También permite comparar autómatas de acuerdo a la relación previamente mencionada.

**Palabras clave.** Computación cuántica, autómatas finitos cuánticos, testing basado en modelos, conformidad, simulador

**Abstract**

This Thesis provides an introduction to Quantum Computing via quantum automata. Additionally, it proposes a novel framework to define quantum finite automata that distinguish between inputs and outputs. Inputs will be symbols of an alphabet, inducing unitary transformations, while outputs will be measurements applied after processing a sequence of inputs. The Thesis gives an implementation relation for these automata. After that, it presents an alternative implementation relation more appropriate for black-box testing. Finally, a quantum finite automata simulator has been developed. This simulator allows users to provide an automata and simulate its behavior. It also allows users to compare two automata according to the previously mentioned relation.

**Palabras clave.** Quantum Computing, quantum finite automata, model based testing, conformance, simulator

# Índice general

# Capítulo 1

# Introducción

Este capítulo presenta una breve descripción de la situación actual de las materias sobre las que trataremos a lo largo de esta memoria: computación cuántica, autómatas finitos cuánticos y testing. Además, se plantean los objetivos de este trabajo y el plan seguido para conseguirlos.

## 1.1 Antecedentes

La mecánica cuántica es una disciplina de la Física que comenzó a desarrollarse a principios del siglo XX. Sin embargo, hasta los años ochenta no se empezó a intentar hacer uso de las leyes de la mecánica cuántica en el mundo de la computación. Desde entonces, la computación cuántica se ha ido desarrollando y ha ido despertando cada vez mayor interés. Esto último se debe a la obtención de algunos resultados en los que el paradigma cuántico supera al clásico, como puede ser el caso del algoritmo de Shor para la factorización de enteros [18].

En cuanto a los autómatas finitos cuánticos, se trata de un campo cuyo desarrollo comenzó a finales del siglo XX. Debido al auge de la computación cuántica, se deseaba construir modelos de computación análogos a los modelos clásicos existentes. El primer modelo de autómata finito cuántico fue presentado por Kondacs y Watrous [11]. Con el tiempo, este modelo ha acabado siendo conocido como MMQFA (*measure many quantum finite automata*). De manera independiente, Moore y Crutchfield [13] presentaron otro modelo alternativo: los autómatas MOQFA (*measure once quantum finite automata*). A partir de estos dos modelos han ido apareciendo otros diferentes. Sin embargo, en este trabajo trataremos sólo con los dos primeros modelos citados porque son los más interesantes para ser extendidos con la inclusión de inputs y outputs.

Otro de los campos que nos interesa en este trabajo es el *testing*. El *testing* es una rama de la Ingeniería del Software ocupada de controlar la calidad de un sistema software. En el caso clásico, se trata de una disciplina bastante desarrollada [14, 5].

En el caso cuántico, aunque se encuentra todavía en sus etapas más preliminares, se ha ido incrementando el trabajo de los investigadores en los últimos años [12]. Entre los distintos tipos de testing que existen, en este trabajo estamos interesados en el *testing basado en modelos*. Más concretamente, se pretende que este trabajo represente una contribución a la hora de determinar si el comportamiento de una determinada implementación es acorde a una especificación. Tomaremos como referencia el marco más usado en la actualidad [19], así como algunas extensiones con información probabilística del mismo [17].

## 1.2 Organización y plan de trabajo

Este trabajo tiene dos objetivos fundamentales. El primero de ellos es plantear un marco en el que, a partir de los modelos de autómatas finitos cuánticos existentes, sea posible distinguir entre inputs y outputs. El segundo objetivo es el de desarrollar un simulador de autómatas finitos cuánticos.

A la hora de desarrollar el trabajo, los primeros pasos han estado centrados en adquirir los conocimientos necesarios sobre computación cuántica y sobre autómatas cuánticos. Para ello, se ha hecho uso de algunos libros introductorios sobre computación cuántica [16, 8], así como de múltiples artículos sobre autómatas. Además, también se ha realizado una aproximación al *testing* mediante artículos relacionados con el *testing* cuántico y con el *testing* basado en modelos. A partir de estos conocimientos, el siguiente paso se ha centrado en la elaboración del Trabajo de Fin de Grado realizado para la Facultad de Matemáticas, en el que se desarrollan en mayor profundidad los conceptos sobre computación cuántica y autómatas que aquí se usan. Una vez dispuesto todo lo anterior, se ha procedido a desarrollar el simulador de autómatas y a plantear el marco de autómatas con inputs y ouputs.

La memoria presenta, en primer lugar, una breve introducción a la computación cuántica en la que se proponen los conceptos necesarios para el desarrollo posterior. Después, se muestran las definiciones de los dos modelos de autómatas en los que estamos interesados. En el siguiente capítulo, se presenta el marco de autómatas con inputs y outputs. Por último, comentamos las características principales del simulador de autómatas y cómo ha sido desarrollado.

# Capítulo 2

# Introduction

This chapter presents a brief introduction to the topics that we will treat along this Thesis: Quantum Computing, quantum finite automata and testing. We also set the goals of this project and the plan followed to achieve them.

## 2.1 Previous work

Quantum mechanics is a branch of Physics that appeared in the beginning of the XX century. However, it was not until the 1980s when scientists started to apply quantum phenomena in computing. Since then, Quantum Computing has been growing and gaining increasing attention. This attention is due to some results that show *quantum supremacy*, like Shor's algorithm for integer factorization [18].

Quantum finite automata theory started in the late XX century. Due to the rise of Quantum Computing, computer scientists started to think about developing computing models similar to the the classical ones. The first quantum finite automata model was presented by Kondacs and Watrous [11]. After the years, this model has been known as MMQFA (measure many quantum finite automata). Moore and Crutchfield [13] presented a different model: MOQFA (measure once quantum finite automata). After these two approaches, other different models have been proposed. Nevertheless, we will focus our attention on these first two models, because they are more interesting for an extension where inputs and outputs are taken into account.

The other topic we are interested in for this Thesis is testing. Testing is a part of Software Engineering that tries to increase the quality of software systems. In classical computing, testing has been extensively developed [14, 5]. In Quantum Computing, however, it is still in a starting stage, but the work of researchers in this area has grown in recent years [12]. Among all the different existing types of testing, we are particularly interested in *model based testing*. Specifically, it is intended that this Thesis represents a contribution to determine whether the behaviour of an implementation complies to a specification. We take as a reference the most used

framework [19] and a probabilistic extension of it [17].

## 2.2 Objectives and work plan

This Thesis has two main goals. The first one is to propose a framework where one of the existing models of quantum automata is extended with the capability of distinguishing between inputs and outputs. The second one is to develop a quantum finite automata simulator.

To achieve these goals, the first steps had to do with acquiring the knowledge about Quantum Computing and quantum automata necessary for the rest of the work. To get this knowledge, we used some quantum computing introductory books [16, 8] and several quantum finite automata papers. Moreover, we also had a first contact with testing via quantum testing and model based testing papers. After all that, the next step was to do the project for the Faculty of Mathematics, where all the quantum computing and quantum automata concepts used here are discussed in greater depth. Once we completed this preliminary work, we started to define a framework where inputs and outputs were taken into account and to develop the automata simulator.

This Thesis starts with a brief introduction to Quantum Computing needed for the later sections. After that, we present the two models of quantum automata we are interested in. Next, we propose the framework of automata with inputs and outputs. Finally, we describe the main characteristics of the automata simulator and how it has been developed.

# Capítulo 3

# Introducción a la computación cuántica

Este capítulo está dedicado a realizar un resumen sobre los conceptos matemáticos básicos necesarios para introducirse en la computación cuántica. Es una introducción breve, pero útil, para comprender las definiciones de autómatas finitos cuánticos que presentaremos más adelante.

## 3.1 Espacios de Hilbert y estados cuánticos

Un sistema cuántico viene descrito por un estado cuántico. Este estado no es más que un vector de un tipo de espacio concreto llamado espacio de Hilbert. En esta sección trataremos ambas nociones planteando las definiciones necesarias.

### 3.1.1 Espacio de Hilbert

El objeto matemático más importante para explicar la mecánica cuántica y la computación cuántica es el de *espacio de Hilbert*. Antes de presentar la definición de espacio de Hilbert, cabe mencionar que se hará uso de la notación de Dirac. Así, $|\psi\rangle$ representa un vector columna denominado *ket*. El vector conjugado traspuesto de $|\psi\rangle$, $|\psi\rangle^*$, es el vector fila $\langle\psi|$ y recibe el nombre de *bra*. Por último, el producto interior de dos vectores $|\psi\rangle$ y $|\phi\rangle$ se expresa como $\langle\psi|\phi\rangle$ mientras que el producto exterior se representa como $|\psi\rangle\langle\phi|$.

**Definición 3.1.1.** Un *espacio con producto interior* $H$ sobre un cuerpo $\mathbb{K}$ (que puede ser $\mathbb{C}$ o $\mathbb{R}$) es un espacio vectorial equipado con un producto interior (o producto escalar), que es una función $\langle\cdot|\cdot\rangle : H \times H \to \mathbb{K}$ que satisface:

1. $\langle\psi|\psi\rangle \geq 0$. Además, $\langle\psi|\psi\rangle = 0 \Leftrightarrow |\psi\rangle = 0$.
2. $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$.

3. $\langle\psi|\alpha\phi + \beta\phi\rangle = \alpha\langle\psi|\phi\rangle + \beta\langle\psi|\phi\rangle$, para $\alpha, \beta \in \mathbb{K}$.

Este producto interior induce la norma $\|\cdot\| : H \to \mathbb{R}$

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

A su vez, la norma induce la distancia entre dos vectores $d : H \times H \to \mathbb{R}$

$$d(|\psi\rangle, |\phi\rangle) = \||\psi\rangle - |\phi\rangle\|$$

Un espacio con producto interior $H$ es *completo* si toda sucesión de Cauchy en $H$ es convergente.

Un *espacio de Hilbert* $H$ es un espacio con producto interior que es completo con respecto a la distancia inducida por su norma.

En el caso finito dimensional, todo espacio con producto interior es un espacio de Hilbert. En este caso, que es el único que nos interesará, dados dos vectores $|\psi_1\rangle = (\alpha_1, \ldots, \alpha_n)^T$ y $|\psi_2\rangle = (\beta_1, \ldots, \beta_n)^T$, el producto interior que consideramos es el usual

$$\langle\psi_1|\psi_2\rangle = \sum_{i=1}^{n} \alpha_i^* \beta_i$$

Además, el producto exterior es

$$|\psi_1\rangle\langle\psi_2| = \begin{pmatrix} \alpha_1\beta_1^* & \alpha_1\beta_2^* & \ldots & \alpha_1\beta_n^* \\ \alpha_2\beta_1^* & \alpha_2\beta_2^* & \ldots & \alpha_2\beta_n^* \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_n\beta_1^* & \alpha_n\beta_2^* & \ldots & \alpha_n\beta_n^* \end{pmatrix}$$

Usualmente, el espacio de Hilbert que utilizaremos es $\mathbb{C}^n$. Sin embargo, al trabajar con autómatas, será necesario otro espacio que pasamos a definir.

**Definición 3.1.2.** Dado un conjunto numerable $Q$, denotamos por $\ell_2(Q)$ al espacio de todas las funciones complejas $x : Q \to \mathbb{C}$ tales que $\left(\sum_{i\in Q} x^*(i)x(i)\right)^{1/2} < \infty$. Nótese que $\ell_2(Q)$ es un espacio de Hilbert respecto del producto interior

$$\begin{aligned} \ell_2(Q) \times \ell_2(Q) &\to \mathbb{C} \\ (x_1, x_2) &\mapsto \langle x_1 \mid x_2\rangle = \sum_{i\in Q} x_1^*(i)x_2(i) \end{aligned}$$

donde $x^*(i)$ representa el conjugado traspuesto de $x(i)$.

Esta definición vale para cualquier conjunto $Q$ numerable. Nosotros la utilizaremos únicamente con conjuntos finitos. En ese caso, se tiene que el espacio $\ell_2(Q)$ es isomorfo al espacio complejo $\mathbb{C}^n$ donde $n$ es el número de elementos de $Q$. Este

hecho es fácil de ver dado que si $Q = \{q_1, \ldots, q_n\}$ y tenemos una función $x : Q \to \mathbb{C}$, esta puede identificarse con el vector $|\psi\rangle = (x(q_1), \ldots, x(q_n))^T$ y, para dos vectores $|\psi_1\rangle = (\alpha_1, \ldots, \alpha_n)^T$ y $|\psi_2\rangle = (\beta_1, \ldots, \beta_n)^T$, se tiene

$$\langle\psi_1|\psi_2\rangle = \sum_{i=1}^{n} \alpha_i^* \beta_i = \sum_{q_i \in Q} x_1^*(q_i) x_2(q_i) = \langle x_1|x_2\rangle$$

donde $x_1$ y $x_2$ son las funciones con imágenes $x_1(q_i) = \alpha_i$, $x_2(q_i) = \beta_i$ para $1 \leq i \leq n$.

### 3.1.2 Estado cuántico

En un sistema cuántico, un estado vendrá dado por un vector de un espacio de Hilbert complejo $H$. Este espacio puede tener dimensión finita o infinita. A la hora de hablar de computación cuántica nos interesa exclusivamente el caso finito, donde un estado de un sistema cuántico es un vector

$$|\psi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

con $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{C}$.

Una condición que ha de cumplir $|\psi\rangle$ es la de que sus coeficientes (llamados *amplitudes*) satisfagan

$$|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_n|^2 = 1 \tag{3.1}$$

Esta condición es equivalente a decir que el vector tiene longitud 1, pues $\||\psi\rangle\|^2 = \langle\psi|\psi\rangle = |\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_n|^2 = 1$. Por tanto, un estado cuántico es un vector unitario con coeficientes complejos que se encuentra dentro de un espacio de Hilbert.

En el contexto de un espacio de Hilbert $H$ podemos considerar una base ortonormal $B = \{|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_n\rangle\}$. Es posible trabajar en esta base y escribir un estado como

$$|\psi\rangle = \alpha_1 |\phi_1\rangle + \alpha_2 |\phi_2\rangle + \ldots + \alpha_n |\phi_n\rangle$$

Los vectores $|\phi_1\rangle, |\phi_2\rangle, \ldots, |\phi_n\rangle$ reciben el nombre de *estados básicos* y la base $B$ el de *base computacional*. En esta situación, suele decirse que el sistema se encuentra en una *superposición* de los estados $|\phi_1\rangle, \ldots, |\phi_n\rangle$.

#### Qubits

Si tenemos un espacio de Hilbert de dimensión 2, podemos considerar una base del mismo $B = \{|0\rangle, |1\rangle\}$. Un estado cuántico tendrá la forma $|\psi\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$. Este sistema cuántico recibe el nombre de *qubit*. El término qubit se usa por su analogía

con el bit de la computación clásica. La diferencia radica en que un bit tiene sólo dos posibles estados, 0 y 1, mientras que un qubit se encuentra en una superposición de estos dos estados.

Como sucede en el caso clásico, también podemos trabajar con sistemas cuánticos de más de un qubit. Un sistema de varios qubits recibe el nombre de *registro cuántico* y para definirlo es necesaria la noción de *producto tensorial*.

**Definición 3.1.3.** Dados dos espacios de Hilbert $H_1$ y $H_2$ de dimensiones $m$ y $n$, respectivamente, el *producto tensorial* de ambos espacios es el espacio

$$H = H_1 \otimes H_2$$

de dimensión $m \cdot n$ donde un vector $|\psi\rangle \in H$ es el producto tensorial de dos vectores $|\psi_1\rangle = (\alpha_1, \ldots, \alpha_m)^T \in H_1$ y $|\psi_2\rangle = (\beta_1, \ldots, \beta_n)^T \in H_2$. Este producto tensorial viene dado por el vector

$$|\psi_1\rangle \otimes |\psi_2\rangle = (\alpha_1\beta_1, \ldots, \alpha_1\beta_n, \alpha_2\beta_1, \ldots, \alpha_2\beta_n, \ldots, \alpha_m\beta_1, \ldots, \alpha_m\beta_n)^T$$

En particular, si tenemos dos bases $B_1 = \{|\phi_1\rangle, \ldots, |\phi_m\rangle\}$ y $B_2 = \{|\varphi_1\rangle, \ldots, |\varphi_m\rangle\}$ de $H_1$ y $H_2$, respectivamente, puede obtenerse la base de $H$:

$$B = \{|\phi_i\rangle \otimes |\varphi_j\rangle \mid 1 \leq i \leq m,\ 1 \leq j \leq n\}$$

Retomando el apartado anterior, si tenemos dos qubits y se considera la base ortonormal $\{|0\rangle, |1\rangle\}$, tomaremos como base del registro cuántico $B = \{|0\rangle, |1\rangle\} \otimes \{|0\rangle, |1\rangle\}$. Esta base se suele escribir como $B = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ donde $|ij\rangle = |i\rangle \otimes |j\rangle$.

Entonces, a partir de dos qubits $|\psi_1\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ y $|\psi_2\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle$, el producto tensorial de ambos qubits será

$$\begin{aligned} |\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) \\ &= (\alpha_0, \alpha_1) \otimes (\beta_0, \beta_1) = (\alpha_0\beta_0, \alpha_0\beta_1, \alpha_1\beta_0, \alpha_1\beta_1)^T \\ &= \alpha_0\beta_0 |00\rangle, \alpha_0\beta_1 |01\rangle, \alpha_1\beta_0 |10\rangle, \alpha_1\beta_1 |11\rangle \end{aligned}$$

Esta noción puede generalizarse para el caso de registros de $n$ qubits. En ese caso, tendremos el espacio de Hilbert $2^n$-dimensional con la base $B = \{|i\rangle \mid\ i \in \{0,1\}^n\}$. Nótese que la dimensión del espacio de Hilbert crece exponencialmente con el número de qubits.

Sin embargo, no todos los estados cuánticos pueden ser representados mediante el producto tensorial de estados de dimensión inferior. Un ejemplo de ello es el conocido como *estado de Bell*:

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

Intentemos expresar este estado como el producto tensorial de dos qubits

$$\begin{aligned}\frac{|00\rangle + |11\rangle}{\sqrt{2}} &= (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) \\ &= \alpha_0\beta_0 |00\rangle , \alpha_0\beta_1 |01\rangle , \alpha_1\beta_0 |10\rangle , \alpha_1\beta_1 |11\rangle\end{aligned}$$

En primer lugar, debe cumplirse que $\alpha_0\beta_1 = \alpha_1\beta_0 = 0$. Sin embargo, para que se cumpla esta condición es necesario que o bien $\alpha_0$ o $\beta_1$ sean 0 y que al menos uno de los dos, $\alpha_1$ o $\beta_0$, también sea igual a 0. Esto llevaría a que la amplitud de $|00\rangle$ o la de $|11\rangle$ fuera igual a 0, lo cual es imposible. Por tanto, no se puede expresar el estado de Bell como producto tensorial de dos qubits. Un estado como este, que no puede ser expresado mediante un producto tensorial, recibe el nombre de *estado entrelazado*.

#### Estados mixtos

Los estados que hemos definido anteriormente suelen denominarse *estados cuánticos puros*. También existen los llamados *estados mixtos*. Un estado mixto es una combinación probabilista de estados cuánticos puros. De este modo, un estado mixto es de la forma

$$p_1 |\psi_1\rangle + p_2 |\psi_2\rangle + \ldots + p_k |\psi_k\rangle$$

donde $|\psi_1\rangle , |\psi_2\rangle , \ldots , |\psi_k\rangle$ son estados cuánticos y $p_1, p_2, \ldots, p_k \in \mathbb{R}$ verifican $p_1 + p_2 + \ldots + p_k = 1$. Un estado mixto se suele escribir de la forma $\{(p_i, |\psi_i\rangle) \mid 1 \geq i \geq k\}$.

## 3.2 Transformaciones unitarias

La evolución de un sistema cuántico viene dada por un operador lineal $T : H \to H$. Este operador tiene una matriz asociada $U$ que ha de ser unitaria. Por tanto, una transformación está determinada por una *matriz unitaria* $U$, que es aquella que cumple

$$U^* = U^{-1} \tag{3.2}$$

El requisito de que la transformación sea unitaria se debe a que se ha de seguir cumpliendo la condición expresada en la ecuación 3.1, es decir, si $|\psi'\rangle = U |\psi\rangle$ entonces se debe cumplir

$$\| |\psi'\rangle \|^2 = \| |U\psi\rangle \|^2 = \langle \psi | U^* U | \psi \rangle = \langle \psi | \psi \rangle = 1$$

Nótese que una condición equivalente a la que hemos expresado en la ecuación 3.2 es que las columnas de $U$ formen una base ortonormal de $H$.

Un ejemplo de transformación unitaria sencillo de entender es el de las *puertas cuánticas*. Cuando se trabaja en computación clásica, un bit es modificado mediante

una puerta lógica. En el caso cuántico, cuando trabajamos con qubits, la evolución de estos también viene dada por puertas. Una puerta cuántica aplica una transformación unitaria sobre un estado. Por ejemplo, si tenemos un bit clásico, podemos transformarlo mediante una puerta NOT. Esta puerta modifica el valor del bit de 0 a 1 o de 1 a 0 según sea el valor inicial. En el caso cuántico, si tenemos un qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, podemos plantearnos aplicar una transformación análoga de tal forma que obtengamos el estado $\beta\,|0\rangle + \alpha\,|1\rangle$. Esto puede hacerse mediante la puerta NOT, cuya matriz es

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Claramente $XX^* = I$ y $X$ es una matriz unitaria.

## 3.3 Medición de un estado

Dentro del mundo cuántico, uno de los conceptos más importantes tiene que ver con la obtención de información sobre el estado de un sistema. Este proceso de obtención de información es lo que se denomina una medición. Existen diferentes tipos de mediciones, pero nos centraremos exclusivamente en una aproximación, la de las mediciones proyectivas.

### 3.3.1 Medición proyectiva

A la hora de obtener información sobre el estado de un sistema cuántico, como puede ser un qubit, habrá que observarlo. Sin embargo, pese a que un estado cuántico es una superposición de estados básicos, no es posible observar dicha superposición.

Por ejemplo, en el caso del qubit, no es posible observar la superposición de estados básicos $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. Sólo es posible observar uno de los dos estados $|0\rangle$ o $|1\rangle$, ya que al realizar la observación el estado *colapsa* sobre alguno de los dos vectores de la base. Además, una vez realizada la observación, el proceso no es reversible y no se puede volver al estado $|\psi\rangle$.

Si en lugar de un espacio de dimensión 2, tenemos un espacio de dimensión $n$ con base $B = \{|\phi_1\rangle, \ldots, |\phi_n\rangle\}$ y un estado del sistema $|\psi\rangle = \alpha_1\,|\phi_1\rangle + \ldots + \alpha_n\,|\phi_n\rangle$, tampoco será posible observar la superposición de estados de la base. Cada estado $|\phi_i\rangle$ será observado con probabilidad $|\alpha_i|^2$. Esta medición recibe el nombre de *medición en la base computacional*.

La noción de medición en la base computacional puede ser generalizada de la siguiente manera:

Consideramos un conjunto de subespacios de $H$ disjuntos y ortogonales entre sí $\mathcal{O} = \{E_1, E_2, \ldots, E_k\}$ de modo que $H = E_1 \oplus E_2 \oplus \ldots \oplus E_k$. Este conjunto $\mathcal{O}$ recibe

el nombre de *observable*. Podemos ahora escribir el estado $|\psi\rangle$ como

$$|\psi\rangle = |\psi_1\rangle + |\psi_2\rangle + \ldots + |\psi_k\rangle$$

donde $|\psi_i\rangle \in E_i$ es la proyección ortogonal de $|\psi\rangle$ sobre $E_i$. Al medir con respecto a $\mathcal{O}$, obtendremos el subespacio $E_i$ con probabilidad

$$\|\psi_i\|^2$$

y el estado colapsará en

$$\frac{|\psi_i\rangle}{\|\psi_i\|}$$

Veamos ahora cómo es la proyección sobre un espacio $E_i$. Si tenemos un subespacio $E_i$ del espacio de Hilbert $H$ anterior, podemos considerar una base del mismo $B_i = \{|\phi_1\rangle, \ldots, |\phi_m\rangle\}$, donde $m \leq n = \dim(H)$. Esta base puede ser ampliada a una base de $H$, $B = \{|\phi_1\rangle, \ldots, |\phi_m\rangle, |\phi_{m+1}\rangle, \ldots, |\phi_n\rangle\}$, añadiendo una serie de vectores adicionales adecuadamente elegidos. Entonces, el *operador proyección* $P_i$ sobre el subespacio $E_i$ vendrá dado mediante la expresión

$$P_i = \sum_{j=1}^{m} |\phi_j\rangle \langle\phi_j|$$

Es fácil ver que, efectivamente, se trata de una proyección sobre $E_i$. Si tenemos un vector $|\psi\rangle = (\alpha_1, \ldots, \alpha_n)^T \in H$ escrito en la base $B$, como $\langle\phi_j|\psi\rangle = \alpha_j$, se cumple que

$$P_i |\psi\rangle = \sum_{j=1}^{m} \alpha_j |\phi_j\rangle$$

Nótese que al ser el operador $P$ una proyección, verificará $P^2 = P$. Además, se trata de un *operador autoadjunto*, es decir, $P = P^*$.

Si volvemos al caso anterior, donde teníamos el observable $\mathcal{O} = \{E_1, E_2, \ldots, E_k\}$, la probabilidad de obtener el subespacio $E_i$ será igual a

$$\|P_i |\psi\rangle\|^2 = \langle\psi|P_i|\psi\rangle$$

y el estado colapsará en

$$\frac{P_i |\psi\rangle}{\|P_i |\psi\rangle\|}$$

En ocasiones, al tratar con autómatas, cada uno de los subespacios que formen un *observable* vendrá dados como el subespacio generado a partir de un conjunto de vectores. Por ejemplo, el subespacio $E_i$ anterior es el subespacio generado por los vectores $|\phi_1\rangle, \ldots, |\phi_m\rangle$ y para denotarlo se hace uso del operador span. Así, $E_i = \text{span}(|\phi_1\rangle, \ldots, |\phi_m\rangle)$.

### 3.3.2 Medición de un registro de varios qubits

Si tenemos, por ejemplo, un registro de dos qubits en un estado

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{10}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

puede que estemos interesados en medir un sólo qubit. De ese modo, proyectaríamos sobre el subespacio generado por $\{|00\rangle, |01\rangle\}$ y obtendríamos

$$\frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\|\alpha_{00}|00\rangle + \alpha_{01}|01\rangle\|} = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

con probabilidad $|\alpha_{00}|^2 + |\alpha_{01}|^2$.

Consideremos ahora el estado de Bell mencionado anteriormente. Si medimos el primer qubit, obtendremos el valor 0 con probabilidad $\frac{1}{2}$. En ese caso, el estado colapsará en $|00\rangle$. Si ahora queremos volver a medir, en esta ocasión respecto del segundo qubit, obtendremos el valor 0 con probabilidad 1. Al medir el primer qubit, el valor 1 también se obtiene con probabilidad $\frac{1}{2}$. En ese caso, el estado observado es $|11\rangle$. Al medir por segunda vez, sobre el segundo qubit, obtendríamos el valor 1 con probabilidad 1. Claramente, el resultado de la segunda observación viene determinado por el resultado obtenido en la primera. Esta es una propiedad particular de los estados entrelazados muy importante en muchos ámbitos de la computación cuántica.

# Capítulo 4

# Autómatas finitos cuánticos unidireccionales

Como se ha comentado en la introducción de esta memoria, los dos modelos más relevantes de autómatas finitos cuánticos son el de Moore y Crutchfield y el de Kondacs y Watrous. De estos modelos pueden considerarse dos versiones: la unidireccional y la bidireccional. En este capítulo nos centraremos exclusivamente en presentar las definiciones de los dos autómatas mencionados en su versión unidireccional, acompañadas de algunos ejemplos que ayuden a entenderlas. En el Trabajo de Fin de Grado realizado en la Facultad de Matemáticas se muestran en detalle los resultados más importantes relacionados con estos autómatas que tienen que ver, en particular, con la capacidad de reconocimiento de lenguaje que poseen. En dicho documento también se presentan las versiones bidireccionales. Aunque dichas versiones son más potentes que las unidireccionales, también son más complejas y resultan menos interesantes para plantear un modelo similar al utilizado en la computación clásica donde se distinga entre inputs y outputs.

## 4.1 MOQFA

En primer lugar nos centramos en la definición de Moore y Crutchfield donde se presentan los autómatas finitos cuánticos *measure once* [13]. Empezamos por ella ya que se trata de la versión más sencilla (y menos potente) de autómata finito cuántico.

**Definición 4.1.1.** Un *autómata finito cuántico measure once* (MOQFA) es una 5-tupla

$$(\Sigma, H, \{U_\sigma \mid \sigma \in \Sigma\}, |s_{init}\rangle, H_{acc})$$

donde:

- $\Sigma$ es el alfabeto de entrada.

- $H$ es un espacio de Hilbert de dimensión finita. Este espacio suele venir dado por un conjunto de estados clásicos $Q = \{|q_0\rangle, |q_1\rangle, \ldots, |q_n\rangle\}$, es decir, $H = \ell_2(Q)$.

- $U_\sigma$ es la matriz unitaria que determina la transición correspondiente al símbolo $\sigma \in \Sigma$.

- $|s_{init}\rangle \in H$ es el estado inicial (que ha de cumplir $\||s_{init}\rangle\|^2 = 1$).

- $H_{acc}$ es un subespacio del espacio de Hilbert $H$ llamado subespacio de aceptación y $P_{acc}$ el operador que proyecta un vector sobre él.

El autómata lee una palabra $w \in \Sigma^*$ símbolo a símbolo, aplicando la transformación $U_\sigma$ correspondiente en cada caso sobre el estado $|\psi\rangle$ del autómata, que empieza siendo $|s_{init}\rangle$. Tras leer entera la palabra, se observa y se acepta si se obtiene el subespacio $H_{acc}$. La probabilidad de que al medir se obtenga el subespacio de aceptación $H_{acc}$ será igual a

$$\left\| P_{acc} U_{w_{|w|}} \cdots U_{w_1} |s_{init}\rangle \right\|^2$$

## 4.2 MMQFA

Una vez tenemos la definición de autómata *measure once*, podemos introducir la versión presentada por Kondacs y Watrous: los autómatas finitos cuánticos *measure many* [11]. Esta noción se diferencia de la anterior en que en lugar de hacer una medición tras leer la palabra completa, se hace una medición tras la lectura de cada símbolo de la palabra. Esta versión es más potente que la representada por los autómatas MOQFA, hasta el punto de que para todo autómata MOQFA existe un autómata *measure many* equivalente.

**Definición 4.2.1.** Un *autómata finito cuántico measure many* (MMQFA) es una 7-tupla

$$(\Sigma, H, \{U_\sigma \mid \sigma \in \tilde{\Sigma}\}, |q_0\rangle, H_{acc}, H_{rej}, H_{non})$$

que presenta las siguientes diferencias respecto a la noción de autómata finito dada en la definición 4.1.1:

- Se tiene en cuenta el alfabeto $\tilde{\Sigma} = \Sigma \cup \{\#, \$\}$, donde # y \$ son dos símbolos que marcan el inicio y el final de la palabra, respectivamente.

- El estado inicial $|s_{init}\rangle$ es un estado clásico, por lo que se suele expresar como $|q_0\rangle \in Q$.

- La regla de aceptación es diferente a la de los MOQFA. Se divide $H$ en tres subespacios $H_{acc}$, $H_{rej}$ y $H_{non}$ de modo que $H = H_{acc} \oplus H_{rej} \oplus H_{non}$. Tras la

lectura de un símbolo $\sigma \in \Sigma$, se realiza una observación. Si se obtiene $H_{acc}$, la ejecución termina y se acepta la cadena. Si es $H_{rej}$, la ejecución también termina pero no se acepta la cadena. Por último, si obtenemos $H_{non}$, la ejecución continúa salvo si se ha leído ya la palabra completa (incluído \$), en cuyo caso se rechaza. Además, si tras medir la ejecución continúa, el siguiente estado en que nos encontraremos será $\frac{P_{non}U_\sigma |\psi\rangle}{\|P_{non}U_\sigma |\psi\rangle\|}$.

De este modo, tras la lectura de un símbolo $\sigma$, el autómata aceptará con probabilidad $\|P_{acc}U_\sigma |\psi\rangle\|^2$, rechazará con probabilidad $\|P_{rej}U_\sigma |\psi\rangle\|^2$ y continuará con probabilidad $\|P_{non}U_\sigma |\psi\rangle\|^2$.

Para esta definición hemos hecho uso de los símbolos # y \$. En realidad, el símbolo # es prescindible, ya que para todo autómata MMQFA que satisfaga la definición 4.2.1 puede construirse uno equivalente que no haga uso del símbolo # [6]. Tampoco es necesario que el estado inicial sea un estado clásico, bastaría con que tenga norma 1, como comprobamos en el trabajo realizado en la Facultad de Matemáticas.

### 4.2.1 Función de evolución

Una forma de expresar la evolución de un autómata $M$ tras la lectura de un símbolo $\sigma \in \Sigma$ es mediante el operador

$$T_\sigma : H \times \mathbb{C} \times \mathbb{C} \to H \times \mathbb{C} \times \mathbb{C}$$
$$(|\psi\rangle, p_{acc}, p_{rej}) \mapsto (P_{non}U_\sigma |\psi\rangle, p_{acc} + \|P_{acc}U_\sigma |\psi\rangle\|^2, p_{rej} + \|P_{rej}U_\sigma |\psi\rangle\|^2)$$

Una configuración $(|\psi\rangle, p_{acc}, p_{rej})$ representa la situación del autómata hasta ese momento: habrá aceptado con probabilidad $p_{acc}$, rechazado con probabilidad $p_{rej}$ y no parado con probabilidad $\||\psi\rangle\|^2$. Como configuración inicial tomamos $(|q_0\rangle, 0, 0)$. Además, dada una palabra $w \in \Sigma^*$, escribimos $T_w = T_{w_{|w|}} \cdots T_{w_1}$. Así, tras la lectura de $w$, la configuración del autómata será $T_w(|s_{init}\rangle, 0, 0) = (|\psi\rangle, p_{acc}, p_{rej})$. Es importante ver que, salvo en la configuración inicial, $|\psi\rangle$ no representa el estado del autómata en ese momento (la superposición de estados clásicos) sino su proyección sobre el espacio $H_{non}$.

## 4.3 Aceptación de un lenguaje

En las dos definiciones de autómatas finitos cuánticos que acabamos de ver, un autómata reconoce una cadena con una determinada probabilidad. Algo así no sucedía, por ejemplo, en el caso de los autómatas finitos clásicos, donde una cadena era aceptada o rechazada siempre por el autómata. Esto hace que existan diferentes formas de entender la aceptación de un lenguaje. De ellas, las dos más relevantes

son el *reconocimiento de un lenguaje con error no acotado* y el *reconocimiento con error acotado*. En ambos casos se parte de la función

$$\begin{aligned} f\colon \Sigma^* &\to [0,1] \\ w &\mapsto f(w) \end{aligned}$$

que asigna a cada cadena la probabilidad de aceptación que tiene en el autómata.

1. Para definir el reconocimiento con error no acotado se hace uso de la idea de reconocimiento con punto de corte (estricto o no estricto). Se llama lenguaje reconocido con *punto de corte estricto* $\lambda \in \mathbb{R}$ al conjunto de palabras

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

   A su vez, se llama lenguaje reconocido con *punto de corte no estricto* $\lambda \in \mathbb{R}$ al conjunto de palabras

   $$L = \{w \in \Sigma^* \mid f(w) \geq \lambda\}$$

   Se dice entonces que un lenguaje $L$ es reconocido con *error no acotado* si existe un punto de corte $\lambda$ tal que $L$ es reconocido con punto de corte (estricto o no) $\lambda$.

2. Un lenguaje $L$ es reconocido con *error acotado* (o *cota de error* $\epsilon \in \left[0, \frac{1}{2}\right)$) si:

   - $f(w) \geq 1 - \epsilon$ cuando $w \in L$
   - $f(w) \leq \epsilon$ cuando $w \notin L$

## 4.4 Ejemplos

En esta sección veremos dos ejemplos de autómatas finitos cuánticos, uno de un MOQFA y otro de un MMQFA.

### 4.4.1 MOQFA

Consideremos el lenguaje $NEQ = \{w \in \{a,b\}^* \mid |w|_a \neq |w|_b\}$ y sea el autómata MOQFA dado por: $Q = \{|q_0\rangle, |q_1\rangle\}$, $H = \ell(Q)$, $\Sigma = \{a,b\}$, $H_{acc} = \text{span}(|q_1\rangle)$ y transformaciones unitarias dadas por las matrices

$$U_a = \begin{pmatrix} \cos\sqrt{2}\pi & -\operatorname{sen}\sqrt{2}\pi \\ \operatorname{sen}\sqrt{2}\pi & \cos\sqrt{2}\pi \end{pmatrix},\ U_b = \begin{pmatrix} \cos(-\sqrt{2}\pi) & -\operatorname{sen}(-\sqrt{2}\pi) \\ \operatorname{sen}(-\sqrt{2}\pi) & \cos(-\sqrt{2}\pi) \end{pmatrix}$$

Este autómata aplica una rotación de ángulo $\sqrt{2}\pi$ sobre el estado del autómata tras leer una $a$ y otra rotación de ángulo $-\sqrt{2}\pi$ tras leer una $b$. Claramente, tras leer una

cadena que verifique $|w|_a = |w|_b$, el autómata habrá hecho las mismas rotaciones de ángulo $\sqrt{2}\pi$ que de ángulo $-\sqrt{2}\pi$ y estará en el estado $|q_0\rangle$, por lo que aceptará con probabilidad 0. Por contra, tras leer una cadena $w \in NEQ$, el autómata no podrá estar nunca en el estado $|q_0\rangle$ (o $-|q_0\rangle$) ya que la rotación es de un múltiplo irracional de $\pi$. Por tanto,

- $f(w) > 0$ si $w \in NEQ$.
- $f(w) = 0$ si $w \notin NEQ$.

Tenemos entonces que el autómata reconoce el lenguaje $NEQ$ con error no acotado.

### 4.4.2 MMQFA

Consideramos el autómata MMQFA sin símbolo # dado por: $\Sigma = \{a\}$, $Q = \{q_0, q_1, q_2, q_3\}$, $H = \ell(Q)$, $|s_{init}\rangle = |q_0\rangle$, $H_{acc} = \text{span}(|q_2\rangle)$, $H_{non} = \text{span}(|q_0\rangle, |q_1\rangle)$, $H_{rej} = \text{span}(|q_3\rangle)$ y

$$U_a = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}, \; U_\$ = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Si tenemos una cadena $w \in \Sigma^*$ distinta de $\epsilon$, la primera transición será

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Tras la lectura de este primer símbolo, se medirá y se rechazará con probabilidad $\frac{1}{2}$ y no se parará con probabilidad $\frac{1}{2}$. En este segundo caso, el autómata colapsará en el estado $\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle$ normalizado, es decir, $\frac{1}{\sqrt{2}}|q_0\rangle + \frac{1}{\sqrt{2}}|q_1\rangle$. Haciendo uso de la función de evolución y partiendo de la configuración inicial $(|q_0\rangle, 0, 0)$, tras la lectura de esa primera aparición del símbolo $a$ tenemos la configuración

$$T_a(|q_0\rangle, 0, 0) = (\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle, 0, \frac{1}{2})$$

A partir de aquí, cualquier otra $a$ que se lea tras este estado deja inalterado el estado $\frac{1}{\sqrt{2}}|q_0\rangle + \frac{1}{\sqrt{2}}|q_1\rangle$, pues

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix}$$

Este estado es un estado de no parada, por lo que ni se aceptará ni se rechazará y la configuración del autómata se mantendrá como estaba: $(\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle, 0, \frac{1}{2})$. Es claro que el autómata permanecerá en ese estado hasta llegar al símbolo \$, cuya transición nos deja el estado

$$U_{\$} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

en el que se acepta con probabilidad $\frac{1}{2}$ y se rechaza con probabilidad $\frac{1}{2}$. Para determinar la probabilidad de aceptación de la cadena, podemos tener en cuenta que la probabilidad de no haber parado hasta ese punto era de $\frac{1}{2}$ o hacer uso de la función de evolución. Vemos que tras leer \$, la configuración del autómata pasará a ser $T_{\$}(\frac{1}{2}|q_0\rangle + \frac{1}{2}|q_1\rangle, 0, \frac{1}{2}) = (0|q_0\rangle + 0|q_1\rangle, \frac{1}{4}, \frac{1}{2} + \frac{1}{4})$. Tenemos, entonces, que el autómata aceptará toda cadena no vacía con probabilidad $\frac{1}{4}$ y la rechazará con probabilidad $\frac{3}{4}$.

En el caso de la cadena vacía, como $U_{\$}|q_0\rangle = |q_3\rangle$ y $|q_3\rangle$ es un estado de $H_{rej}$, se rechaza con probabilidad 1.

# Capítulo 5

# Autómatas finitos cuánticos con inputs y outputs

En este capítulo presentamos nuestro marco de autómatas donde distinguimos entre inputs y outputs y definimos relaciones de conformidad entre dichos autómatas. En primer lugar, realizaremos una breve introducción al campo del testing basado en modelos. A continuación, a partir de los autómatas MOQFA, plantearemos el modelo de autómata con el que trabajar indicando qué será considerado un input o un output. Después, aportaremos una primera relación de implementación entre autómatas. Las limitaciones prácticas de esta definición nos llevarán a plantear una segunda relación, menos precisa pero más apropiada. Por último, daremos una idea de cómo podría plantearse un marco similar para autómatas MMQFA.

## 5.1 Testing basado en modelos

El *testing* es una disciplina cuyo principal objetivo es el de encontrar errores en un sistema informático para su posterior corrección. Existen diversos tipos de *testing*, pero en este trabajo nos centraremos exclusivamente en el *testing* basado en modelos [10, 9].

El *testing* basado en modelos es una técnica de *testing* en la que se dispone de un sistema que quiere ser analizado y una especificación que describe el comportamiento que ha de tener el sistema. La especificación se presupone correcta y viene dada en algún tipo de lenguaje formal. Para comprobar que la implementación se comporta de manera acorde a la especificación es necesario definir una relación entre ambas. Así, si la implementación y la especificación están relacionadas, podremos decir que la implementación es conforme a la especificación. Esta relación recibe el nombre de *relación de implementación* o *de conformidad*.

A lo largo de este capítulo presentaremos un marco para trabajar con autómatas finitos cuánticos donde no se considera un único alfabeto sino que utilizamos los

conceptos de input y output. En primer lugar, definiremos el modelo con el que vamos a trabajar. Este modelo será el de los autómatas finitos cuánticos *measure once*, para los que determinaremos unos conjuntos de inputs y de outputs específicos. Tanto la implementación como la especificación vendrán dadas como uno de estos autómatas cuánticos. A partir de este modelo se definirán dos relaciones de implementación que relacionen una implementación con una especificación. La primera de ellas, más sencilla, es menos útil desde un punto de vista práctico. Es por ello que se presentará la segunda definición. Esta definición parte de un conjunto de ejecuciones concretas realizadas sobre la implementación y la relaciona con una especificación de acuerdo a un cierto grado de confianza.

## 5.2 MOQFAIO

Como acabamos de mencionar, el modelo para la especificación consiste en un autómata finito cuántico. En particular, será un MOQFA. Sobre este autómata definimos dos tipos de interacciones: inputs y outputs. Identificaremos los símbolos del alfabeto como inputs y las mediciones realizadas al final como outputs, es decir:

- Un símbolo $\sigma \in \Sigma$ es un input y este input determina una acción, una transformación unitaria.
- La medición realizada tras leer una secuencia de inputs es considerada un output del sistema. Esta medición puede ofrecer un estado del subespacio de aceptación, de modo que el autómata reconocería la secuencia, o un estado del subespacio complementario, no reconociéndola. El primer caso lo asociaremos con el output 1 y el segundo, con el 0. El valor esperado será entonces igual a la probabilidad de aceptación.

Este es el modelo que emplearemos para una especificación $s$ y que se formaliza en la siguiente definición.

**Definición 5.2.1.** Un *autómata finito cuántico measure once con inputs y outputs* (MOQFAIO) es una 6-tupla de la forma $(H, |s_{init}\rangle, I, \{U_i | i \in I\}, H_{acc}, \{X_w | w \in I^*\})$, donde:

- $(I, H, \{U_i \mid i \in I\}, |s_{init}\rangle, H_{acc})$ es un autómata MOQFA.
- $I$ se corresponde con el alfabeto de entrada del MOQFA y es considerado el conjunto de inputs del sistema. Cada input $i \in I$ lleva asociada una acción: una transformación unitaria dada por la matriz $U_i$.
- $X_w$ es una variable aleatoria que representa el resultado de la medición realizada tras el procesamiento de una secuencia de inputs $w = i_1 i_2 \cdots i_n \in I^*$ y es

lo que consideramos un output. Esta medición devolverá el valor 1 (aceptación) o el valor 0 (no aceptación) con probabilidades $\|P_{acc}U_w\,|s_{init}\rangle\|^2$ y $1-\|P_{acc}U_w\,|s_{init}\rangle\|^2$, respectivamente.

Denotamos por $\mathcal{M}$ al conjunto de todos los MOQFAIO.

Claramente, podemos considerar cada $X_w$ como una variable aleatoria con distribución Bernoulli de parámetro $\|P_{acc}U_w\,|s_{init}\rangle\|^2$. Este parámetro, que se corresponde con la probabilidad de que $X_w$ sea igual a 1, es el que nos va a interesar más adelante y con el que planteamos la siguiente definición.

**Definición 5.2.2.** Dado un MOQFAIO $(H, |s_{init}\rangle, I, \{U_i | i \in I\}, H_{acc}, \{X_w | w \in I^*\})$, definimos la función $out : I^* \to [0,1]$ tal que a cada $w \in I^*$ le asigna la probabilidad de obtener el output 1:

$$out(w) = P(X_w = 1) = \|P_{acc}U_w\,|s_{init}\rangle\|^2$$

### 5.2.1 Ejemplo

Consideramos el autómata del ejemplo 4.4.1 que reconocía el lenguaje $NEQ$, donde el conjunto de inputs es $I = \Sigma = \{a, b\}$. En el ejemplo vimos que el autómata aplicaba rotaciones de ángulo $\sqrt{2}\pi$ tras leer una $a$ y de ángulo $-\sqrt{2}\pi$ al leer una $b$. La composición de dichas rotaciones es una rotación cuyo ángulo es la suma de las rotaciones aplicadas. Entonces, tras una secuencia de inputs $w \in I^*$, el estado final será el resultado de aplicar una rotación de ángulo $\theta = |w|_a\sqrt{2}\pi - |w|_b\sqrt{2}\pi = \sqrt{2}\pi(|w|_a - |w|_b)$ sobre $|s_{init}\rangle = |q_0\rangle$, es decir, $\cos\theta\,|q_0\rangle + \operatorname{sen}\theta\,|q_0\rangle$. Considerando además que $H_{acc} = \operatorname{span}(|q_1\rangle)$, podemos definir $out(w)$ de la siguiente manera:

$$\begin{aligned} out : I &\to [0,1] \\ w &\mapsto out(w) = \operatorname{sen}^2\theta = \operatorname{sen}^2(\sqrt{2}\pi(|w|_a - |w|_b)) \end{aligned}$$

Así, por ejemplo, para la secuencia de inputs $w = aab$ se tiene $out(w) = \operatorname{sen}^2\sqrt{2}\pi$

## 5.3 Relación de implementación para autómatas finitos cuánticos

A lo largo de esta sección se busca presentar una relación entre una implementación y una especificación. Consideraremos en este caso que tanto la implementación como la especificación pertenecen al conjunto de todos los MOQFAIO, es decir, a $\mathcal{M}$.

Siguiendo el patrón marcado por la relación ioco [19], la idea de la que partimos para definir esta relación es que si tenemos una implementación $i$ tal que para cualquier secuencia de inputs $w$ obtiene el output 1 con la misma probabilidad

que una especificación $s$, entonces ambas están relacionadas. En ese caso, también se obtendrá el output 0 con la misma probabilidad. Así, llegamos a la siguiente definición.

**Definición 5.3.1.** Sean $i, s \in \mathcal{M}$ y sea $I$ el conjunto de inputs de $s$. Decimos que $i$ es conforme a $s$, y lo denotamos por $i \sim s$, si se cumple la siguiente condición:

$$\forall w \in I^* : out_i(w) = out_s(w)$$

Nótese que en esta definición nos restringimos a comprobar que la implementación se comporta igual que la especificación solo para las secuencias de inputs que están *especificadas* en dicha especificación.

### 5.3.1 Ejemplo

En esta sección se muestran tres ejemplos de MOQFAIO sencillos. Dos de ellos nos servirán como modelo de dos especificaciones $s_1$ y $s_2$ y otro como modelo de una implementación $i$. Veremos si la relación $\sim$ de la definición 5.3.1 se cumple entre $i$ y $s_1$ y entre $i$ y $s_2$.

1. $i$ viene dada por el siguiente autómata MOQFAIO: $Q = \{|q_0\rangle, |q_1\rangle, |q_2\rangle\}$, $H = \ell_2(Q)$, $|s_{init}\rangle = \frac{1}{\sqrt{2}}|q_0\rangle + \frac{1}{\sqrt{2}}|q_1\rangle$, $H_{acc} = \text{span}(|q_0\rangle, |q_1\rangle)$, $I = \{a\}$ y

$$U_a = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

2. $s_1$ viene dada por el MOQFAIO: $Q = \{|q_0\rangle, |q_1\rangle\}$, $H = \ell_2(Q)$, $|s_{init}\rangle = |q_0\rangle$, $H_{acc} = \text{span}(|q_0\rangle)$, $I = \{a\}$ y

$$U_a = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

3. $s_2$ viene dada por el MOQFAIO: $Q = \{|q_0\rangle, |q_1\rangle\}$, $H = \ell_2(Q)$, $|s_{init}\rangle = |q_0\rangle$, $H_{acc} = \text{span}(|q_0\rangle)$, $I = \{a\}$ y

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

En el caso de $s_1$, tenemos que $U_a|q_0\rangle = |q_1\rangle$ y $U_a|q_1\rangle = |q_0\rangle$. Como $H_{acc} = \text{span}(|q_0\rangle)$, para una secuencia de inputs $w \in I^*$ el autómata ofrece el output 1 con probabilidades:

- $out_{s_1}(w) = 1$ si la secuencia de inputs $w$ tiene longitud par.

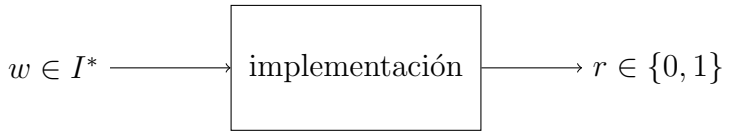


Figura 5.1: Representación gráfica de una implementación vista como caja negra

- $out_{s_1}(w) = 0$ si la secuencia de inputs $w$ tiene longitud impar.

Atendiendo al caso de $i$, podemos ver por una parte que

$$U_a \left| s_{init} \right\rangle = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \left| q_2 \right\rangle$$

mientras que por otra parte, $U_a \left| q_2 \right\rangle = \left| s_{init} \right\rangle$. Así, si la longitud de la secuencia de inputs $w$ es impar, $out_i(w) = 0$, ya que el estado final del autómata será $\left| q_2 \right\rangle$, que no es de aceptación. Si la longitud de $w$ es par, entonces $out_i(w) = 1$ dado que el estado final del autómata será $\frac{1}{\sqrt{2}} \left| q_0 \right\rangle + \frac{1}{\sqrt{2}} \left| q_1 \right\rangle$, que está en el subespacio de aceptación. Por tanto, $i \sim s_1$.

Si atendemos a $s_2$, el autómata cumple $U_a \left| s_{init} \right\rangle = \frac{1}{\sqrt{2}} \left| q_0 \right\rangle + \frac{1}{\sqrt{2}} \left| q_1 \right\rangle$ y

$$U_a \left( \frac{1}{\sqrt{2}} \left| q_0 \right\rangle + \frac{1}{\sqrt{2}} \left| q_1 \right\rangle \right) = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Tenemos entonces que toda secuencia de inputs $w$ de longitud par llegará al estado $\left| q_0 \right\rangle$, que es de aceptación, y por tanto $out_{s_2}(w) = 1$. Sin embargo, tras una secuencia de longitud impar, se encontrará en el estado $\frac{1}{\sqrt{2}} \left| q_0 \right\rangle + \frac{1}{\sqrt{2}} \left| q_1 \right\rangle$, por lo que $out_{s_2}(w) = \frac{1}{2}$, distinta a la probabilidad que teníamos en $i$ para secuencias de longitud impar. Por tanto, $i \not\sim s_2$.

Aunque la definición anterior es correcta desde el punto de vista formal, y mimetiza el comportamiento de la ya mencionada ioco, presenta una importante limitación si la consideramos en el entorno de una caja negra. Para que se cumpla la relación que hemos planteado es necesario que toda secuencia de inputs que se realice sobre la implementación, y que puede ser generada por la especificación, tenga la misma probabilidad de aceptación que en la especificación. Sin embargo, desde la perspectiva del testing de caja negra no es posible comprobar que ambas probabilidades sean iguales. Al trabajar con una caja negra, sólo es posible observar un output concreto dada una secuencia de inputs. Es decir, si ejecutamos una secuencia de inputs sobre una implementación, observaremos o el valor 1 o el valor 0 (véase una representación gráfica en la figura 5.1). Esto hace necesaria otra definición algo menos restrictiva y que haga uso de una muestra concreta de ejecuciones. Esta relación más práctica es la que presentamos en la siguiente sección.

## 5.4 Relación de implementación basada en una muestra

La relación presentada en la sección anterior consideraba que era posible conocer las probabilidades $out_i(w)$ de la implementación. Como esto solo es posible desde un punto de vista teórico, presentamos a continuación una nueva relación que tenga en cuenta un conjunto finito de ejecuciones realizadas sobre la implementación.

Para la nueva definición utilizaremos la *prueba binomial*, útil para realizar contrastes de hipótesis sobre experimentos concretos que tienen dos posibles resultados. En este contraste de hipótesis se busca dilucidar si, dada una muestra, se puede afirmar con un cierto nivel de significación que una determinada variable aleatoria se ajusta a una distribución presupuesta. Pasamos a continuación a describir su funcionamiento.

Supongamos que tenemos una muestra $X$ correspondiente a $n$ ensayos de Bernoulli independientes que representa el número de éxitos obtenidos. Si consideramos que un experimento concreto tiene una probabilidad de éxito $p$, el valor esperado para $X$ será $n \cdot p$ y su función de probabilidad tendrá la forma

$$P(X = k) = \binom{n}{k} p^k \cdot (1-p)^{n-k}$$

Consideramos ahora dos opciones distintas: si el número $k$ de éxitos obtenidos verifica $k < n \cdot p$, entonces $P(X \leq k)$ determina el nivel de significación. Si $k > n \cdot p$, entonces viene determinado por $P(X \geq k)$.

A partir del test binomial definimos la siguiente función que será útil para la nueva definición de relación de conformidad.

$$F_n(p, X) = \begin{cases} P(X \leq k) & \text{si } k < n \cdot p \\ P(X \geq k) & \text{si } k > n \cdot p \\ 1 & \text{si } k = n \cdot p \end{cases}$$

Presentaremos la nueva relación haciendo uso de la prueba binomial. Sin embargo, para muestras grandes puede ser mejor hacer uso de otras pruebas, como la prueba $Z$, pues la prueba binomial es más costosa de computar en esos casos.

Una vez hemos definido la función $F$, podemos presentar la nueva definición. Esta definición estará asociada a una muestra concreta. En nuestro caso, el valor de $p$ vendrá dado por $out_s(w)$ y diremos que una implementación está relacionada con una especificación, dados una muestra y un valor $\alpha$ entre 0 y 1, si el nivel de significación obtenido en la prueba binomial para esa muestra es mayor que $\alpha$.

**Definición 5.4.1.** Sea $s \in \mathcal{M}$ una especificación, siendo $I$ su conjunto de inputs, $i \in \mathcal{M}$ una implementación, $M$ un multiconjunto de pares que pertenecen a $I^* \times \{0, 1\}$ y

$0 \leq \alpha \leq 1$. Decimos que $i$ es conforme a $s$ con respecto al conjunto de ejecuciones $M$ y con significación $\alpha$, y lo denotamos por $i \sim_{(\alpha,M)} s$, si se cumple la siguiente condición:

$$\forall w \in I^* : (\exists r \in \{0,1\} : (w,r) \in M) \Longrightarrow F_{M(w)}(out_s(w), M_1(w)) > \alpha$$

donde $M(w)$ denota el número de veces que aparece $w$ en $M$ y $M_1(w)$ representa el número de veces que se ha obtenido el output 1 para la secuencia de inputs $w$ en ese multiconjunto de ejecuciones.

## 5.5 MMQFAIO

Del mismo modo que hemos planteado un modelo de autómatas *measure once* que distinguía entre inputs y outputs, podríamos plantearnos hacer lo mismo para autómatas *measure many*. En el resto de esta sección esbozaremos las líneas que deberían seguirse para definir formalmente, y de forma análoga a la que acabamos de definir, una relación para este tipo de autómatas.

Podría partirse de la misma idea y considerar el alfabeto de entrada como conjunto de inputs y el conjunto de mediciones como outputs. Como mencionamos en el capítulo 4, los autómatas MMQFA realizan mediciones tras la lectura de cada símbolo y, en caso de observar un estado de aceptación o de rechazo, paran. Si no, continúan. Tendríamos entonces un mayor número de outputs. Sin embargo, sería posible considerar que dos autómatas están relacionados si las probabilidades de aceptación para cualquier secuencia de inputs son iguales para todas las secuencias de inputs posibles.

Después, si tratásemos el caso práctico, habría que considerar que tras dos ejecuciones concretas sobre el autómata, el número de mediciones que se realizarían para la misma cadena podría variar. De todas formas, para una secuencia de inputs concreto observaríamos como mucho el output aceptación una sola vez, ya que tras observarlo el autómata pararía. Por tanto, podríamos hacer uso de esta variable para desarrollar una relación similar a la presentada en la sección 5.4.

# Capítulo 6

# Simulador

Este capítulo está dedicado al simulador de autómatas finitos cuánticos desarrollado en el ámbito de este Trabajo de Fin de Grado. El programa está escrito en *Python*. Se trata de un simulador sencillo que permite introducir las opciones necesarias para definir un autómata y después simularlo. También se permite la introducción de un segundo autómata para poder comparar ambos siguiendo la definición 5.3.1. De este modo, el simulador posee tres vistas: una primera para la creación de autómatas, la segunda para la simulación de un autómata y la última para la comparación de dos autómatas. El programa ha sido desarrollado siguiendo el patrón de diseño MVC y su código completo puede encontrarse en `https://github.com/javiegal/Simulador-QFA`. En este repositorio también se puede encontrar tanto la memoria de este Trabajo de Fin de Grado como el correspondiente a la Facultad de Matemáticas.

## 6.1 Diseño del simulador

Como acabamos de mencionar, el simulador ha sido desarrollado siguiendo el patrón de diseño *Modelo-Vista-Controlador*. Detallamos a continuación cada uno de los paquetes que contiene el sistema:

1. El modelo se encarga de la lógica del programa. Posee una clase *QFA* de la que heredan las clases *MOQFA* y *MMQFA*. Estas dos clases representan cada uno de los dos tipos de autómatas posibles y permiten la creación de un autómata, el procesamiento de una palabra y las comprobaciones necesarias sobre las transformaciones unitarias y el observable (subespacio de aceptación y su complementario para los MOQFA y los subespacios de aceptación, rechazo y no parada para los MMQFA). Por último, tenemos una clase *Modelo* que contiene dos autómatas y gestiona todas las manipulaciones que se hacen sobre los mismos.

2. La vista se encarga de presentar el modelo al usuario. Para desarrollarla se ha

hecho uso del paquete *Tkinter*. Contiene una clase *UI* con tres vistas que se corresponden con lo mencionado anteriormente: la *VistaInicio*, que contiene dos objetos de la clase *VistaCrear* y que es la encargada de manejar la creación de los dos posibles autómatas; la *VistaSimular*, encargada de mostrar el procesamiento de palabras por parte del autómata; y la *VistaComparar*, encargada de la comparación de los dos autómatas introducidos.

Además de todo esto, encontramos otras clases de las que hace uso *VistaCrear* para introducir la información sobre el autómata, como son *Aceptacion*, *Transformaciones* y *MatrizEntrada*.

3. Por último, el controlador está desarrollado en la clase *Controlador* y realiza las funciones que le corresponden según el patrón MVC.

## 6.2 Creación de un autómata

La vista inicial del simulador se corresponde con la de creación de autómatas. En ella se permite la introducción de las opciones necesarias para definir un autómata, que son:

1. Tipo de autómata: MOQFA o MMQFA.
2. Dimensión del espacio de Hilbert.
3. Estado inicial del autómata.
4. Alfabeto de entrada y transformaciones asociadas a cada símbolo del alfabeto. Además, si el autómata es MMQFA, también se introduce la transformación asociada al símbolo \$. Se ha considerado la definición que no hace uso del símbolo #, por lo que no es necesario introducirlo en el caso de seleccionar un autómata MMQFA.
5. Observable que determina la regla de aceptación. En el caso de los autómatas MOQFA sólo es necesario introducir la matriz $P_{acc}$, mientras que en el caso de los autómatas MMQFA se han de introducir las matrices $P_{acc}$, $P_{rej}$ y $P_{non}$.

Además, el sistema permite la comprobación de que las matrices introducidas son correctas. En el caso de las transformaciones unitarias, es posible comprobar que la matriz escrita es efectivamente unitaria. En el caso del observable, permite comprobar que las matrices introducidas configuran adecuadamente un observable. Si el autómata es MOQFA, bastará con comprobar que la matriz introducida es la asociada a una proyección ortogonal, es decir, es hermítica e idempotente. Si se trata de un autómata MMQFA, además de que las tres matrices sean proyecciones ortogonales es necesario que representen subespacios ortogonales que sumen el total.

También será posible seleccionar algún ejemplo preestablecido, entre los que hemos añadido todos los que han sido mencionados en las secciones anteriores.

Cabe destacar que el simulador ha sido elaborado haciendo uso de la biblioteca de cálculo simbólico *Sympy*. De este modo, es posible introducir tanto el estado inicial como las matrices haciendo uso de las expresiones que aporta esta biblioteca. Esto facilita, por ejemplo, la introducción de los ejemplos presentados en las secciones anteriores en los que se hacía uso de funciones trigonométricas o de raíces cuadradas, así como de manejar números complejos fácilmente (la unidad imaginaria es representada en *Smypy* mediante el carácter $I$).

En la figura 6.1 es posible ver un ejemplo de la vista de creación de un autómata donde se introduce el autómata presentado en la sección 4.4.1 que era capaz de reconocer el lenguaje $NEQ = \{w \in \{a, b\}^* \mid |w|_a \neq |w|_b\}$ con error no acotado.

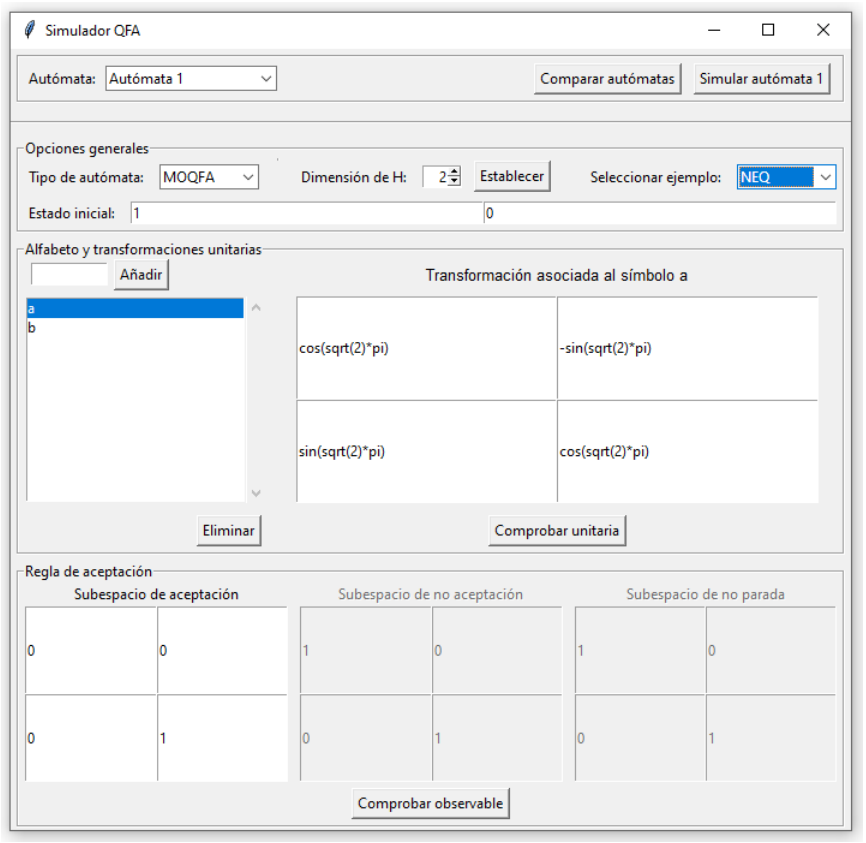


Figura 6.1: Creación del autómata que reconoce el lenguaje $NEQ$.

## 6.3 Simulación de un autómata

Una vez creado un autómata, es posible pasar a su simulación. Ello sólo será posible si el autómata introducido es correcto, pues se harán las comprobaciones anteriormente mencionadas así como la de que el estado inicial sea un vector de norma 1.

Al pasar a simular un autómata, será posible introducir cualquier cadena que contenga símbolos del alfabeto introducido anteriormente. Además, también se ofrece la posibilidad de leer todas las cadenas que se puedan generar con ese alfabeto hasta una longitud máxima introducida por el usuario.

En la figura 6.2 se muestra un ejemplo en el que se hace uso del autómata propuesto por Ambainis y Freivalds [2] con el que probaron que un autómata MMQFA reconoce el lenguaje $a^*b^*$ con error acotado. En concreto, el autómata reconoce ese lenguaje con cota de error $1-p$ donde $p$ es la raíz del polinomio $p^3+p=1$ ($p \simeq 0.68$). En la imagen puede observarse el resultado de la lectura de todas las cadenas hasta longitud 3 que pueden escribirse con el alfabeto $\{a, b\}$.

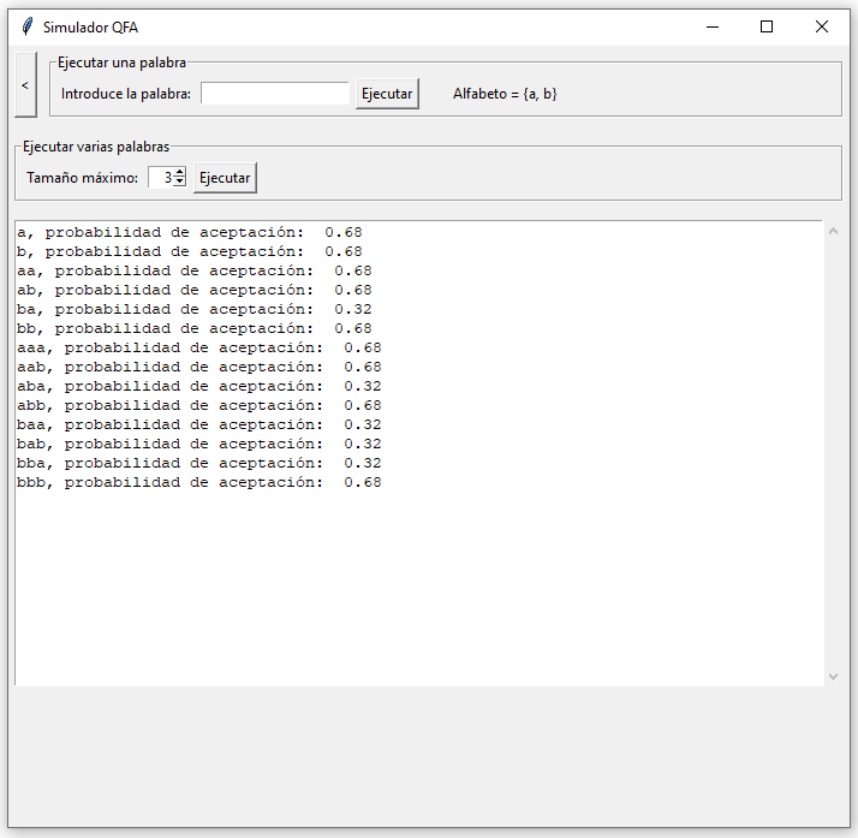


Figura 6.2: Vista de simulación correspondiente al autómata que reconoce $a^*b^*$

## 6.4 Comparación de dos autómatas

La otra funcionalidad del simulador permite comparar dos autómatas MOQFA introducidos. En la vista de creación es posible navegar de un autómata al otro para crear ambos. Una vez creados, además de realizar sobre cada autómata las comprobaciones mencionadas en la sección anterior, también será necesario comprobar

que ambos autómatas son MOQFA y tienen el mismo alfabeto de entrada [1] (el cual constituía el conjunto de inputs).

Una vez llegado a la vista de comparación de autómatas, se podrán crear todas las secuencias de inputs posibles hasta una longitud máxima establecida por el usuario. Estas secuencias de inputs serán procesadas en ambos autómatas y permitirán comprobar si, hasta esa longitud, ambos autómatas están relacionados según la relación de conformidad presentada en la definición 5.3.1.

Como las probabilidades están representadas por *floats* (representación de punto flotante), es necesario realizar las comparaciones de probabilidades añadiendo un grado de tolerancia. En este caso, se ha hecho uso del valor por defecto que presenta la función *isclose()* de la librería *math*, que es igual a $1e-9$.

En las figuras 6.3 y 6.4 se muestran los resultados obtenidos en las comparaciones presentadas en el ejemplo 5.3.1 para secuencias de inputs hasta longitud 15.

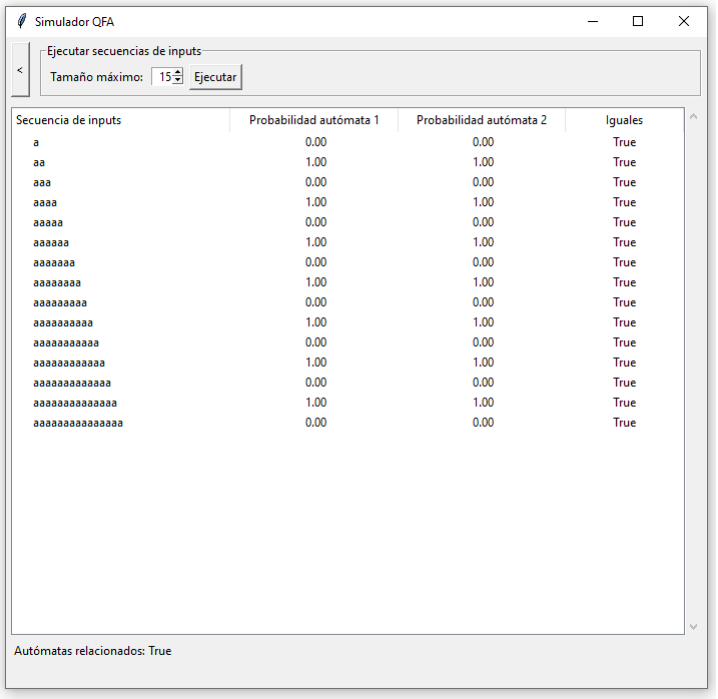


Figura 6.3: Comparación entre $i$ y $s_1$.

[1] Técnicamente, es suficiente con que los inputs de la especificación estén contenidos en los de la implementación pero a efectos prácticos, dado que solo comprobaremos las secuencias de inputs de la especificación, podemos considerar que los dos conjuntos son iguales.

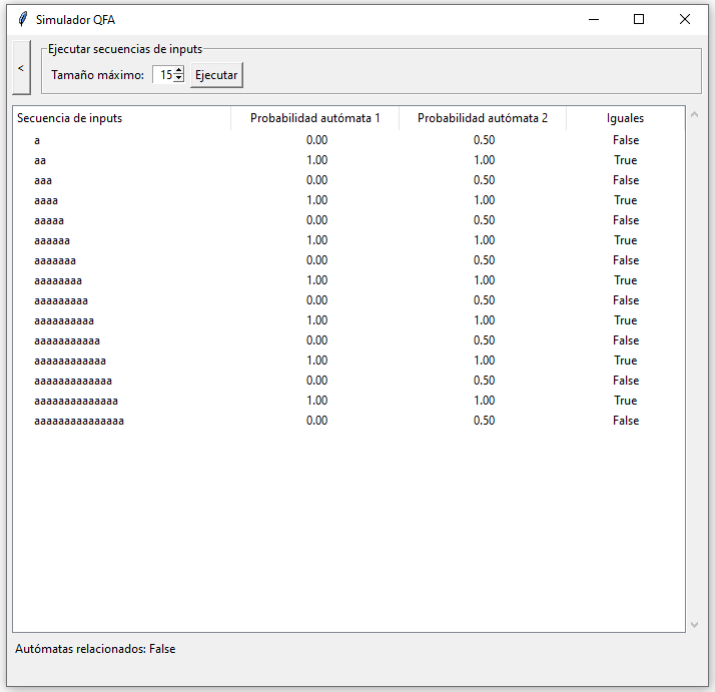


Figura 6.4: Comparación entre $i$ y $s_2$.

# Capítulo 7

# Conclusiones y trabajo futuro

A lo largo de este trabajo hemos podido introducirnos en la computación cuántica haciendo uso de los autómatas finitos cuánticos. Nos hemos centrado en sus versiones más sencillas, pues eran las que presentaban mayor facilidad a la hora de definir un marco en el que trabajar con autómatas que distinguiesen entre inputs y outputs. El primer punto fue plantear qué íbamos a considerar un input y qué un output. Así, llegamos a plantear la definición de autómatas finitos cuánticos con inputs y outputs que aparece en el capítulo 5. Una vez la teníamos, quisimos ver cómo podíamos definir una relación entre este tipo de autómatas. Esto nos hizo llegar a la primera relación de conformidad planteada. Esta relación, más apropiada desde un punto de vista teórico, era peor desde el punto de vista práctico pues no podía ser comprobada en casos de testing de caja negra. Es por ello por lo que se planteó otra relación alternativa, menos precisa pero que puede ser comprobada con un cierto grado de confianza.

Además del trabajo de revisión realizado sobre autómatas finitos cuánticos y la definición del marco teórico, y ante la escasez de herramientas que permitiesen simular autómatas cuánticos, hemos implementado un simulador con una interfaz gráfica sencilla que permite introducir autómatas *measure once* y *measure many* para después simularlos.

En cuanto al trabajo futuro, una de las posibles opciones sería la de definir un marco análogo para autómatas *measure many* tal y como esbozamos en la sección 5.5. En ese caso, podría plantearse un modelo similar al que hemos definido para los autómatas *measure once* que permitiera definir una relación de implementación del mismo tipo (para la que también sería necesario plantear una aproximación práctica más realista). También sería posible pensar relaciones para otros autómatas cuánticos (como el autómata finito cuántico *Latvian* [1] o el autómata finito cuántico propuesto por Nayak [15]) para los que sería necesario definir otras relaciones diferentes. Además, también sería posible añadir nuevas funcionalidades al simulador. Por un lado, existe la posibilidad de simular y comparar otros formalismos que

representen autómatas cuánticos. Por otro lado, se pueden añadir otras funcionalidades que tengan que ver con la capacidad de aceptación de lenguajes por parte de un autómata. Específicamente, y en el marco de la segunda línea planteada, podría resultar útil implementar la relación de implementación basada en ejecuciones que se presenta en la definición 5.4.1.

# Chapter 8

# Conclusions and future work

Along this Thesis, we provided an introduction to Quantum Computing using quantum finite automata. We focused on the simplest automata models, which were easier to manipulate in order to get a framework for automata that distinguishes between inputs and outputs. The first step was to fix what we considered an input and what was an output. Thus, we were able to present the inputs and outputs quantum finite automata definition that appears in Chapter 5. Once we had it, we wanted to see how we could define a relation between these automata. This led us to the first implementation relation presented in this Thesis. This relation, very appropriate from a theoretical point of view, is not suitable in practical terms because it is not possible to check it in a black-box testing framework. Due to that, we presented an alternative relation, less precise but checkable up to some confidence degree.

In addition to the theoretical work, and given the lack of existing tools to simulate automata, we implemented an automata simulator with a simple graphic user interface. This tool allows the user to introduce measure once and measure many automata and simulate them.

Thinking about possible lines for future work, one option has to do with developing the model suggested in Section 5.5 for measure many quantum finite automata. In this case, a similar model to the measure once proposed could be considered. It could present a similar implementation relation (and the subsequent practical approach). Another option has to do with doing the same work for other models of quantum automata (as the Latvian quantum finite automata [1] or the quantum finite automata presented by Nayak [15]) for which different relations should be proposed. Also, some other new functionalities could be added to the simulator. On the one hand, there is the possibility to simulate or compare other models of quantum automata. On the other hand, some additional functionalities related to language recognition could be included. Specifically, the implementation relation based on executions presented in Definition 5.4.1 could be implemented too.

# Bibliografía


[1] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, and D. Therien. Algebraic results on quantum automata. *Theory of Computing Systems*, 39:165–188, 2006.

[2] A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *39th Annual Symposium on Foundations of Computer Science, FOCS'98*, pages 332–341, 1998.

[3] A. Ambainis, A. Kikusts, and M. Valdats. On the class of languages recognizable by 1-way quantum finite automata. In *8th Annual Symposium on Theoretical Aspects of Computer Science, STACS'01, LNCS 2010*, pages 75–86, 2001.

[4] A. Ambainis and A. Yakaryılmaz. Automata and quantum computing. https://arxiv.org/abs/1507.01988, 2018.

[5] P. Ammann and J. Offutt. *Introduction to Software Testing*. Cambridge University Press, 2nd edition, 2017.

[6] A. Brodsky and N. Pippenger. Characterizations of 1-way quantum finite automata. *SIAM Journal on Computing*, 31:1456–1478, 2002.

[7] A. C. Cem Say and A. Yakaryılmaz. Quantum finite automata: A modern introduction. In C. S. Calude, R. Freivalds, and I. Kazuo, editors, *Computing with New Resources*, pages 208–222. Springer, 2014.

[8] J. Gruska. *Quantum Computing*. McGraw-Hill, 1999.

[9] R. M. Hierons, K. Bogdanov, J.P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A.J.H Simons, S. Vilkomir, M.R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Computing Surveys*, 41(2):9:1–9:76, 2009.

[10] R. M. Hierons, J.P. Bowen, and M. Harman, editors. *Formal Methods and Testing, LNCS 4949*. Springer, 2008.

[11] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *38th Annual Symposium on Foundations of Computer Science, FOCS'97*, pages 66–75, 1997.

[12] A. Miranskyy and L. Zhang. On testing quantum programs. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER'19*, 2019.

[13] C. Moore and J. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237:275–306, 2000.

[14] G. J. Myers, C. Sandler, and T. Badgett. *The Art of Software Testing*. John Wiley & Sons, 3rd edition, 2011.

[15] A. Nayak. Optimal lower bounds for quantum automata and random access codes. In *40th Annual Symposium on Foundations of Computer Science, FOCS'99*, pages 369–376, 1999.

[16] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[17] M. Núñez and I. Rodríguez. Towards testing stochastic timed systems. In *23rd IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, FORTE'03, LNCS 2767*, pages 335–350, 2003.

[18] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, FOCS'94*, pages 124–134, 1994.

[19] J. Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing, LNCS 4949*, pages 1–38, 2008.

[20] A. Yakaryılmaz and A. C. Cem Say. Efficient probability amplification in two-way quantum finite automata. *Theoretical Computer Science*, 410:1932–1941, 2009.