

# ESTUDIO DE PROPIEDADES METAMÓRFICAS PARA TESTING DE PROGRAMAS CUÁNTICOS

SINHUÉ GARCÍA GIL

GRADO EN MATEMÁTICAS, FACULTAD DE CIENCIAS MATEMÁTICAS  
UNIVERSIDAD COMPLUTENSE DE MADRID

---



Trabajo Fin Grado en Matemática Computacional

Curso 2022-2023

Director:

Luis Fernando Llana Díaz

# Resumen

El desarrollo de la computación cuántica está en el punto de mira debido a los avances que está teniendo estos últimos años. IBM presentó a finales del año pasado un ordenador con 433 qubits y trabajando para presentar el siguiente sistema cuántico con 1121 qubits a finales de este mismo año<sup>1</sup>. Hay que destacar que el anterior sistema solo tenían 127 qubits, ¡pero este se presentó solo hace 2 años, en 2021!

Con toda esta evolución que está ocurriendo a nivel del número de qubits y las mejoras que se van realizando para alcanzar mayor fiabilidad, empieza a abrir la puerta al uso real de los algoritmos y programas cuánticos. Aunque estos no llegarán hasta conseguir ordenadores con mejores características. Pero, ¿cómo vamos a comprobar la corrección de estos?

Una de las posibilidades que vamos a plantear en este trabajo es como la unión entre la computación cuántica y el *testing* metamórfico puede ayudarnos a contestar esta pregunta y a intentar buscar esa corrección o falta de errores. Donde el *testing* metamórfico es uno de los métodos usados para la búsqueda de errores desde que se presentó en 1998, basado en el estudio de las propiedades lógicas que se pueden obtener de los algoritmos.

**Palabras clave:** Computación cuántica, qiskit, propiedades metamórficas

---

<sup>1</sup><https://www.ibm.com/quantum/roadmap>

# Abstract

The quick development that quantum computing is having in the latest years is opening new chances within the field. IBM presented in 2022 his newest developed processor, *Osprey*, with a 433 qubit structure. They have already set up on their roadmap the possibility to offer his new processor with 1121 qubits by the end of this year. This number will open the research in some prototype software applications within IBM<sup>2</sup>.

Looking at the progress made in the recent years towards qubit capacity and reliability, is making us get closer to unlock the real use of quantum algorithms, although we may need a few thousand more. But as soon as we think of using this algorithms, how are we going to probe their correctness?

We are going to present through this paper one of the alternatives to probe correctness or the lack of faults. We are going to show the possibility of bringing together quantum computing and metamorphic testing, based in logic properties directly from the algorithm. This option has been used as fault finder since 1998 when it was presented.

**Keywords:** Quantum computing, qiskit, metamorphic properties.

---

<sup>2</sup><https://www.ibm.com/quantum/roadmap>

# Índice general

<b>Índice</b>	<b>1</b>
<b>1. Introducción</b>	<b>2</b>
1.1. Metodología . . . . .	3
1.2. Objetivos . . . . .	4
<b>2. Antecedentes</b>	<b>5</b>
2.1. Introducción matemática . . . . .	5
2.2. Introducción cuántica . . . . .	8
2.3. Programación cuántica, Qiskit . . . . .	9
2.3.1. Puertas y circuitos cuánticos . . . . .	12
2.3.2. Simulaciones y ruido . . . . .	17
2.4. Propiedades Metamórficas / Testing metamórfico . . . . .	19
<b>3. Algoritmos cuánticos</b>	<b>22</b>
3.1. Suma . . . . .	22
3.2. Deutsch . . . . .	23
3.3. Deutsch-Jozsa . . . . .	27
3.4. Bernstein-Vazirani . . . . .	31
3.5. Simon . . . . .	31
3.6. QFT . . . . .	31
3.7. QEP . . . . .	31
3.8. Grover . . . . .	31
<b>4. Propiedades metamórficas</b>	<b>32</b>
4.1. DJ . . . . .	32
4.2. BV . . . . .	32
4.3. Simon . . . . .	32
<b>5. Conclusiones y trabajo futuro</b>	<b>33</b>
5.1. Conclusiones . . . . .	33
5.2. Trabajo futuro . . . . .	33
<b>Bibliografía</b>	<b>33</b>

# Capítulo 1

## Introducción

La memoria y el trabajo presentado a continuación, va a ser el recorrido que nos va a mostrar, desde los principio básicos de la mecánica cuántica y como directamente definen la computación cuántica, hasta una de las posibilidades que tenemos para hacer testing sobre estos programas. Antes de proceder con la forma en la que se ha trabajado y como se ha ido evolucionando y aprendiendo, veamos donde comenzó todo.

La mecánica cuántica se empezó a desarrollar en los años 20, pero no sería hasta los años 80 cuando se empezó a plantear la posibilidad de la aplicación que podía tener esta teoría sobre la computación. (Quantum computing, WikiEN) Paul Benioff presentó en 1980 la *máquina de Turing cuántica* que utilizaba la teoría cuántica para describir un ordenador simplificado. Ya en 1984, se utilizó dicha teoría sobre protocolos criptográficos de la mano de Charlos Bennett y Gilles Brassard.

A partir de entonces ya empezaron a surgir nuevos algoritmos, principalmente para resolver el problema de oráculo. Entre ellos, los algoritmos de Deutsch, Deutsch-Jozsa, Bernstein-Vazirani y Simon. Aquí ya se puede observar la mejora en eficiencia respecto al ordenador clásico, como conjeturó Richard Feynman en 1982. Se podría decir, que lo que hizo despertar al resto de la comunidad sobre la importancia que podía tener la computación cuántica, llegó de la mano de Peter Shor en 1994 con sus algoritmos que podrían permitir romper las claves de encriptación RSA y Diffie-Hellman, que aún se siguen utilizando. Desde entonces la inversión en el estudio de este campo ha ido creciendo, siendo el primero ordenador de dos qubits creado en 1998 y haciendo factible esta posibilidad. En la actualidad, IBM tiene el ordenador con mayor número de qubits, 433, el `ibm_seattle` presentado a finales de 2022. Aunque no todo es sobre el número de qubits que disponga el ordenador, si no de la calidad de estos qubits para evitar ruido y modificaciones no deseadas de los estados de nuestro qubit.

## 1.1. Metodología

La metodología seguida para este trabajo digamos que se puede distinguir en 3 fases que van a ser apreciables en esta memoria. De hecho, van a corresponder una a una con los capítulos siguientes. El material principal utilizado como base de estudio han sido los libros *Quantum Computation and Quantum Information*, de Michael A. Nielsen y Isaac L. Chuang, y *Quantum Computing for Computer Scientists*, de Noson S. Yanofsky y Mirco A. Mannucci. (Crear la referencia desde aquí) Vamos a introducir brevemente dichas fases:

- **Antecedentes:** Esta primera fase de estudio se basa en poder avanzar desde los conocimiento que tenemos del grado, a la base útil para poder entender los algoritmos cuánticos, la obtención de las propiedades metamórficas y su utilización en el testing. Además de los textos mencionados anteriormente, de los que he ido obteniendo los resultados más teóricos, en este capítulo se han fijado los conceptos de la parte de testing y propiedades metamórficas con el artículo *Metamorphic Testing: A Review of Challenges and Opportunities*.(ref)
- **Programación cuántica y algoritmos:** Una vez realizada y adquirida la base para empezar a entender los algoritmos y la programación, se realizaron los primeros pasos de programación, empezando por un algoritmo tan sencillo como sería la suma y como sería mi implementación de la misma utilizando *Qiskit*. Posteriormente, fui avanzando algoritmo a algoritmo desde los más simples hasta llegar a la transformada cuántica de Fourier (QFT) y la estimación de fase (QEP). El estudio de estos algoritmos se presentará en el capítulo 3 (ref) y todos los algoritmos programados y pruebas realizadas, para no sobrecargar este documento, se encuentran en el repositorio personal de GitHub, <https://github.com/sinugarc/TFG.git>
- **Propiedades y testing metamórfico:** Ahora que ya hemos conseguido entender e incluso programar nuestros algoritmos cuánticos, vamos a por el último eslabón de la cadena. Este es el estudio de las propiedades metamórficas sobre los algoritmos de Deutsch-Jozsa, Bernstein-Vazirani y Simon. Como apoyo para la comprensión y estudio de este fase, nos hemos apoyado en el artículo *Metamorphic Testing of Oracle Quantum Programs*. Además, esta parte del trabajo se ha realizado de forma conjunta con Rodrigo de la Nuez Moraleda, y todo lo que hemos programado y preparado para el testing de estos algoritmos se puede encontrar en el repositorio de GitHub que tenemos en común, <https://github.com/rodelanu/TFG>

## 1.2. Objetivos

Una vez analizada la metodología de trabajo seguida y los materiales utilizados, siguiendo el mismo esquema, veamos cuales son los objetivos y competencias que se pretenden adquirir en cada fase.

- **Antecedentes:** Queremos ser capaces de entender las definiciones más básicas, partiendo de los conocimientos matemáticos, principalmente del álgebra lineal. Algunos ya conocidos como que representa una matriz unitaria o hermitiana y la relación que existe entre ellas y sus operadores, o introducir nuevos conceptos, como el espacio de Hilbert y el producto tensorial, que nos servirá para generar sistemas más complejos. Además estudiaremos la parte física de computación cuántica, con sus postulados y su relación con la programación cuántica e introduciremos los elementos básicos de dicha programación y el sentido general de las simulaciones y ejecuciones de los programas cuánticos.
- **Programación cuántica y algoritmos:** El objetivo principal de este capítulo es claro, queremos ser capaces de crear y analizar programas. Así como entender como se han construido los diversos algoritmos y la utilidad que tienen.
- **Propiedades y testing metamórfico:** Como indica el título del trabajo, este es el fin de todo el recorrido que hemos realizado, no será hasta el capítulo 4 cuando profundicemos en el testing metamórfico. Lo que queremos es ser capaces de obtener propiedades metamórfica útiles para los diversos algoritmos y programarlas para comprobar la corrección del algoritmo programado.

# Capítulo 2

## Antecedentes

El principal objetivo de este apartado es exponer brevemente al lector las bases, tanto matemáticas como físicas, para poder entender y trabajar con propiedades metamórficas y en particular, su aplicación a la computación cuántica. Para ello, vamos a hacer un breve repaso a conceptos básicos de álgebra lineal en matemáticas, los postulados de la mecánica cuántica y una iniciación a la computación cuántica y el testing metamórfico.

Esta sección, que podría ser una simple continuación de la introducción, va a ser más extensa de lo que se podría esperar. Ya que para trabajar de forma cómoda, sobre el tema a tratar, necesitamos un salto en conocimientos que se han tenido que adquirir.

### 2.1. Introducción matemática

Para poder desarrollar y entender la mecánica cuántica, que presentaré a continuación, la programación cuántica y en particular, sus algoritmos, vamos a necesitar cierta base matemática y de notación. Quizás, las definiciones que siguen este párrafo pueden parecer aleatorias, aunque todo cobrará sentido conforme vayamos profundizando en la mecánica y programación cuántica.

Definimos un **espacio de Hilbert**,  $\mathcal{H}$ , como un  $\mathbb{C}$ -espacio vectorial dotado de un producto interno que es completo. En particular, como vamos a tratar solo  $\mathbb{C}$ -espacios vectoriales con producto interno finito, este será completo.

Por el **Teorema de representación de Riesz**, tenemos que  $\mathcal{H}$  es anti-isomorfo a  $\mathcal{H}^*$ , por ser  $\mathbb{C}$  nuestro cuerpo base.

Denotaremos como ket,  $|v\rangle$ , a un vector  $v$  de  $\mathcal{H}$ . Análogamente, a toda transformación  $w$  de  $\mathcal{H}^*$ , la denotaremos como bra,  $\langle w|$ . Esta notación es conocida como **notación de Dirac** y será utilizada en mecánica cuántica.



Veamos ahora distintas definiciones para operadores:

- Sea  $\mathcal{A} : \mathcal{H} \rightarrow \mathcal{H}$  continuo, se denomina **adjunto del operador lineal**  $\mathcal{A}$  al único operador  $\mathcal{A}^* : \mathcal{H} \rightarrow \mathcal{H}$  talque  $\langle \mathcal{A}v, w \rangle = \langle v, \mathcal{A}^*w \rangle$ .
- Se dice que  $\mathcal{A} : \mathcal{H} \rightarrow \mathcal{H}$  es un **operador autoadjunto** si  $\mathcal{A} = \mathcal{A}^*$ . Por lo cual los autovalores de  $\mathcal{A}$  son reales.
- Llamaremos matriz **hermitiana** a la matriz  $A$  que determina el operador autoadjunto  $\mathcal{A}$ . De aquí obtenemos que  $A^*$  es la traspuesta conjugada de  $A$ .
- Se dice que  $\mathcal{U} : \mathcal{H} \rightarrow \mathcal{H}$  continuo, es **unitario** si  $\langle v, w \rangle = \langle \mathcal{U}v, \mathcal{U}w \rangle$ , que en particular es invertible.

Para finalizar con esta sección vamos a ver una manera de combinar dos espacios vectoriales, en particular, dos espacios de Hilbert que nos permitirá unir dos sistemas cuánticos, esta operación es el producto tensorial. (Nielsen, WikiES)

Sean  $\mathcal{H}$  y  $\mathcal{H}'$  dos espacios de Hilbert,

$$F(\mathcal{H} \times \mathcal{H}') = \left\{ \sum_{i=1}^n z_i(h_i, h'_i) \mid n \in \mathbb{N}, z_i \in \mathbb{C}, (h_i, h'_i) \in \mathcal{B}(\mathcal{H} \times \mathcal{H}') \right\}$$

$$\begin{aligned} \mathcal{R} &= (h_1 + h_2, h') \sim (h_1, h') + (h_2, h') \\ & (h, h'_1 + h'_2) \sim (h, h'_1) + (h, h'_2) \\ & z(h, h') \sim (zh, h') \sim (h, zh') \end{aligned}$$

Definimos el **producto tensorial** de  $\mathcal{H}$  y  $\mathcal{H}'$ ,  $\mathcal{H} \otimes \mathcal{H}'$ , como el espacio cociente entre  $F(\mathcal{H} \times \mathcal{H}')$  y  $\mathcal{R}$ , es decir,  $\mathcal{H} \otimes \mathcal{H}' = F(\mathcal{H} \times \mathcal{H}')/\mathcal{R}$ . Partiendo directamente desde la definición de la relación obtenemos:

- **Linealidad:** Sea  $z \in \mathbb{C}$ ,  $|h\rangle \in \mathcal{H}$  y  $|h'\rangle \in \mathcal{H}'$ . Entonces:

$$z(|h\rangle \otimes |h'\rangle) = (z|h\rangle) \otimes |h'\rangle = |h\rangle \otimes (z|h'\rangle)$$

■ **Asociatividad:**

- Sea  $|h_1\rangle \in \mathcal{H}$ ,  $|h_2\rangle \in \mathcal{H}$  y  $|h'\rangle \in \mathcal{H}'$ . Entonces:

$$(|h_1\rangle + |h_2\rangle) \otimes |h'\rangle = |h_1\rangle \otimes |h'\rangle + |h_2\rangle \otimes |h'\rangle$$

- Sea  $|h\rangle \in \mathcal{H}$ ,  $|h'_1\rangle \in \mathcal{H}'$  y  $|h'_2\rangle \in \mathcal{H}'$ . Entonces:

$$|h\rangle \otimes (|h'_1\rangle + |h'_2\rangle) = |h\rangle \otimes |h'_1\rangle + |h\rangle \otimes |h'_2\rangle$$

Además alcanzamos los siguiente resultados sobre el producto tensorial:

- **Base:**  $|i\rangle \otimes |j\rangle$  es una base de  $\mathcal{H} \otimes \mathcal{H}'$  donde  $|i\rangle$  y  $|j\rangle$  pertenecen a una base ortonormal de  $\mathcal{H}$  y  $\mathcal{H}'$  respectivamente.
- **Producto interno:** los productores internos de  $\mathcal{H}$  y  $\mathcal{H}'$  inducen naturalmente un producto interno en  $\mathcal{H} \otimes \mathcal{H}'$  por lo que hereda la estructura y con ella, las nociones de adjunta, unitaria, normalidad y hermiticidad.
- **Producto de Kronecker,** se corresponde con el producto tensorial de aplicaciones lineales. Nos será de gran utilidad a lo largo de este trabajo y nos ayuda a visualizar el producto tensorial. Veamos un ejemplo, si  $A$  y  $B$  son 2 matrices, entonces:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}$$

- **Dimensión:**  $\dim(\mathcal{H} \otimes \mathcal{H}') = \dim(\mathcal{H}) \cdot \dim(\mathcal{H}')$

Estos dos últimos apartados nos muestran la complejidad que vamos a tener a la hora de realizar cálculos, ya que la dimensión de las matrices va a crecer de manera exponencial.

## 2.2. Introducción cuántica

La base principal para el comienzo de la computación cuántica fue el desarrollo de la física cuántica. Para poder entender mejor que variaciones e implicaciones tiene, vamos a ver los postulados de la mecánica cuántica y sus diferencias con la mecánica Newtoniana. Aquí encontrará sentido la base matemática presentada en 2.1. (referencia?)

Para empezar, en mecánica clásica, un sistema de  $N$  partículas queda definido por un vector en un espacio  $\mathbb{R}^{3N} \times \mathbb{R}^{3N}$  donde las primeras coordenadas definen la posición y las últimas la velocidad. La evolución de este sistema se rige por la segunda Ley de Newton que relaciona la fuerza con la aceleración y la masa.

Por otra parte, en física cuántica, el estado y evolución de un sistema viene determinado por sus postulados que veremos a continuación (referencia a martín, wikipediaEN, Nielsen), así como una de las posibles interpretaciones que cada uno podría tener. Estos postulados nos ayudaran posteriormente a fijar la base de nuestros programas cuánticos.

### Postulados cinemáticos o de representación:

- **Primer postulado:** El estado en un sistema aislado, en un instante  $t$ , se corresponde con  $|\varphi(t)\rangle$ , en un espacio de Hilbert,  $\mathcal{H}$ .
- **Segundo Postulado:** El espacio que representa un sistema compuesto es el producto tensorial del los espacios de cada componente del sistema. Es decir, si tuviéramos  $n$  componentes, el espacio total sería  $|\varphi_1\rangle \otimes |\varphi_2\rangle \otimes \dots \otimes |\varphi_n\rangle = |\varphi_1\varphi_2\dots\varphi_n\rangle$ , de forma notacional.

### Postulados dinámicos:

- **Tercer postulado:** Evolución determinista.
  - **Primer apartado:** La evolución de un vector  $|\varphi(t)\rangle$  está determinado por la ecuación de Schrödinger,  $i\hbar \frac{d|\varphi\rangle}{dt} = H|\varphi\rangle$ . Donde  $H$  es el Hamiltoniano del sistema, que es un operador hermitiano. Además se entiende  $H(t)$  como un observable asociado a la energía total del sistema.
  - **Segundo apartado:** La evolución de un sistema aislado se describe por una transformación unitaria del estado inicial.  $|\varphi(t)\rangle = U(t; t_0)|\varphi(t_0)\rangle$ .

- **Cuarto postulado:** Evolución probabilística, tenemos observador.
- **Primer apartado:** Cada medida  $\mathcal{A}$  esta descrita por un operador hermitiano  $A$  que actúa sobre  $\mathcal{H}$ , decimos que este operador es observable, debido a que sus autovectores forman una base de  $\mathcal{H}$ . El resultado de medir una cantidad  $\mathcal{A}$  debe ser uno de los autovalores correspondientes al observable  $A$ .
- **Segundo apartado:**  $Prob(\lambda_i) = \frac{\|P_{v_i}|\varphi(t)\rangle\|^2}{\|\varphi(t)\rangle\|^2} = \frac{|\langle v_i|\varphi(t)\rangle|^2}{\|\varphi(t)\rangle\|^2}$
- **Tercer apartado:** Si tras realizar una medición  $\mathcal{A}$  del estado  $|\varphi(t)\rangle$  da como resultado  $a_n$ , entonces el estado del sistema colapsa a la proyección normal de  $|\varphi(t)\rangle$  en el subespacio de autovectores asociado a  $a_n$ ,  $P_{v_n}|\varphi(t)\rangle$ .

Todos estos postulados van a ser clave en los distintos aspectos de la computación cuántica, desde la definición del sistema más simple como el *qubit* hasta las probabilidades en las simulaciones (observaciones).

## 2.3. Programación cuántica, Qiskit

La programación cuántica se basa en la creación de un circuito o algoritmo cuántico, normalmente representado geoméricamente, donde se realizan operaciones (operadores unitarios) sobre los distintos qubits, así como sus mediciones. (quantum for inf)

Para realizar nuestros programas cuánticos y simulaciones nos vamos a apoyar en *Qiskit*, es un paquete de desarrollo libre creado por IBM para crear y manipular programas cuánticos así como realizar simulaciones(qiskit). Ya sean teóricas o conectando nuestros programas con los ordenadores cuánticos de IBM. Esto nos dará unos resultados más realistas donde podremos apreciar el ruido que hay actualmente en estos ordenadores. (ref apartado ruido)

La programación en Qiskit es una libreria de Python, en el que se permite definir y utilizar los circuitos. Además, para una mejor visualización de lo que representa el código utilizaré jupyter notebook. Existiría otra opción, que sería generar los circuitos directamente en la página de IBM de forma geométrica.(qiskit)

Además, existirían otras opciones como *Azure Quantum* en *C#* de Microsoft, *Cirq* en *Python* de Google o *Pyket* también en *Python* entre otros.

Como mencionamos anteriormente los postulados cuánticos, ordenados de acorde a la utilización en este texto, nos van a permitir sentar las bases de la computación cuántica, el primer ejemplo es el qubit.

Si recordamos el postulado 1 de la mecánica cuántica (link), vamos a definir **qubit** como el sistema cuántico más simple, que va a ser nuestra base en la programación cuántica. Un **qubit** es un espacio bidimensional, donde vamos a suponer que tomamos la base ortonormal  $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$  y  $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ . De aquí podemos obtener la combinación lineal de cualquier vector de estado del qubit, aunque los vectores de estado deben de cumplir la condición de normalización, es decir, si  $|\varphi\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$  con  $a, b \in \mathbb{C}$ , entonces  $|a|^2 + |b|^2 = 1$ . Esta condición se impone debido a que el cuadrado de los coeficientes determinan la probabilidad de aparición y por lo tanto la suma de todas la probabilidad de todas las posibilidades debe ser 1. Además, llamaremos estado de **superposición** a los estados del qubit que se encuentran en el continuo entre  $|0\rangle$  y  $|1\rangle$ .

Estos estados de un qubit se entienden muy bien utilizando la **esfera de Bloch**, que observaremos en la figura 2.1 <sup>1</sup>. Debido a que si  $|\Psi\rangle = a|0\rangle + b|1\rangle$ ,  $|a|^2 + |b|^2 = 1$ . Podemos representar  $|\Psi\rangle = e^{i\gamma} \left( \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \right)$ , donde  $\theta, \varphi, \gamma \in \mathbb{R}$ . Si bien es cierto, podemos ignorar el factor  $e^{i\gamma}$ , ya que no tiene efectos observables. Hay que tener en cuenta que esta representación está limitada a un qubit, porque no hay manera simple de generalizarla.

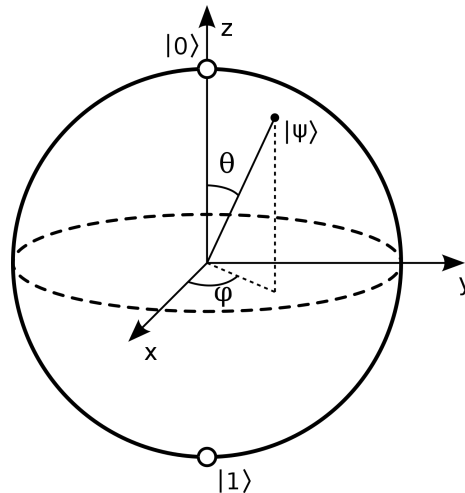


Figura 2.1: Esfera de Bloch

<sup>1</sup>[https://en.wikipedia.org/wiki/Bloch\\_sphere/media/File:Bloch\\_sphere.svg](https://en.wikipedia.org/wiki/Bloch_sphere/media/File:Bloch_sphere.svg)

Veamos que es lo que ocurre ahora cuando en vez de un único qubit, tenemos varios y la relación que hay entre ellos. Para simplificarlo, tomemos el más simple, el sistema con 2 qubits. Si recordamos el postulado 2 nos determinaba como interaccionaban estos dos qubits, como el sistema compuesto por ambos era su producto tensorial. Por ello, podemos definir el estado de un sistema de 2 qubits como:

$$|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle \text{ donde } a_{00}, a_{01}, a_{10}, a_{11} \in \mathbb{C}$$

A estos coeficiente complejos se les denomina **amplitud** de cada estado de la base. Al igual que ocurre en el qubit,  $a_{00} + a_{01} + a_{10} + a_{11} = 1$ . Este resultado se puede obtener del producto de Kronecker, y además tendría sentido con la medición de las probabilidades. En este ejemplo se puede ver claramente que la dimensión de la base va a ser exponencial respecto al número de qubits.

Por último y antes de pasar a las puertas cuánticas y como operamos en el sistema, vamos a introducir un último concepto, el enredo cuántico o *entanglement*. Esta es una propiedad de la mecánica cuántica que aparece en un sistema, se puede decir que un sistema de 2 qubits están en un estado de *entanglement* cuando no es posible separar el estado como producto tensorial de los dos qubits (CforInf), diremos además que existe una correlación entre ellos. Normalmente se observa a la hora de hacer mediciones en dicho sistema, también conocido como *entangled measurements*. Si bien es cierto, aún se sigue investigando esta propiedad, así como completar su teoría. (Nielsen) Veamos un ejemplo simple que nos va a permitir obtener una idea.

Definimos *Bell state* como  $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ , veamos primero que no se puede expresar como producto tensorial.

$$(Bell\ state)\sqrt{2} = 1|00\rangle + 0|01\rangle + 0|10\rangle + 1|11\rangle$$

Suponemos que  $(Bell\ state)\sqrt{2} = (a_0|0\rangle + a_1|1\rangle) \otimes (a'_0|0\rangle + a'_1|1\rangle)$ , entonces:

$$\begin{aligned} (a_0|0\rangle + a_1|1\rangle) \otimes (a'_0|0\rangle + a'_1|1\rangle) &= a_0a'_0|00\rangle + a_0a'_1|01\rangle + a_1a'_0|10\rangle + a_1a'_1|11\rangle \Rightarrow \\ &\Rightarrow a_0a'_0 = a_1a'_1 = 1 \text{ y } a_0a'_1 = a_1a'_0 = 0 \end{aligned}$$

Pero este sistema de ecuaciones no tiene solución, por lo que  $(Bell\ state)\sqrt{2}$  no se puede expresar como producto tensorial y en particular, *Bell state* tampoco.

Ahora queremos realizar una medición con base  $\{|0\rangle, |1\rangle\}$ , pero sabemos que sólo podemos obtener los resultados  $|00\rangle$  o  $|11\rangle$ . Es decir, cuando se mida el primer qubit ya tendremos el valor del segundo sin haberlo medido <sup>2</sup>. Podemos encontrar otros ejemplos y preguntas muy interesantes sobre *entanglement* en <https://quantum-computing.ibm.com/composer/docs/ibmqx/guide/entanglement>, donde estudia distintas posibilidades con la puerta CNOT, que presentaremos a continuación, y estados interesantes como *Bell state*, *GHZ states* and *W states*.

### 2.3.1. Puertas y circuitos cuánticos

Partimos ahora del tercer postulado, en particular el apartado 2 (link). Se podría ver que existe una correspondencia 1 a 1 entre el Hamiltoniano  $H$ , por ser hermitiano, y un operador unitario  $U$ . Estos operadores unitarios serán nuestras **puertas cuánticas** que vamos a usar junto a los qubits. Este postulado viene de la evolución **determinista** desde un puesto de vista dinámico, donde no se realiza ninguna observación sobre el sistema. Uno de los puntos importantes de que estos operadores sean unitarios es que conservan la norma, por lo cual no rompen la condición de normalización de la definición de qubit.

En programación cuántica estos operadores unitarios pueden crearse directamente con una matriz, que cumpla las condiciones necesarias, aunque habitualmente utilizaremos puertas cuánticas (operadores) específicas que son las utilizadas en los ordenadores cuánticos reales (Martín). Veamos cuales son las puertas cuánticas mas útiles para un qubit: (WikipediaEN, Nielsen)

- **Puertas de Pauli:** Estas puertas son la más básicas y nos van a permitir, a excepción de la identidad, realizar rotaciones de  $\pi$  radianes dentro de la esfera de Bloch, cada una sobre el eje que indica su propio nombre. Las matrices que determinan estas puertas generan una base del espacio de matrices hermitianas  $2 \times 2$ .

- **Puerta identidad:**  $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ , con puerta geométrica  $\text{---} \boxed{I} \text{---}$
- **Y:** La puerta  $Y$ ,  $\text{---} \boxed{Y} \text{---}$ , viene determinada por la matriz  $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
- **Z:** La puerta  $Z$ ,  $\text{---} \boxed{Z} \text{---}$ , viene determinada por la matriz  $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

---

<sup>2</sup><https://jcc.dcc.fceia.unr.edu.ar/2006/slides/2006-diazcaro-samborskiforlese.pdf>

- **X**: La puerta  $X$  es un operador que viene determinado por la matriz  $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Esta puerta sería la análoga cuántica a la puerta NOT clásica y nos permite

$$|0\rangle \rightarrow |1\rangle, |1\rangle \rightarrow |0\rangle, \text{ por lo que dado } |\varphi\rangle = a|0\rangle + b|1\rangle \Rightarrow X|\varphi\rangle = b|0\rangle + a|1\rangle$$

Su puerta geométrica al realizar los circuitos será:  $\text{---}\boxed{X}\text{---}$

- **Puerta de Hadamard,  $H$** : Este operador viene determinado por  $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Probablemente la puerta más interesante de todas, que nos permite poner el qubit en un estado especial, es más, dichas transformaciones sobre la base tiene su propia notación:

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

A su vez, al igual que las matrices de Pauli, la matriz de Hadamard realiza una rotación de  $\pi$  radianes, pero esta vez sobre el eje  $(\hat{x} + \hat{z})/\sqrt{2}$ . Puerta:  $\text{---}\boxed{H}\text{---}$

- **Puerta de cambio de fase,  $P(\theta)$** : Esta puerta nos va a permitir realizar rotaciones de un ángulo  $\theta$  sobre el eje  $\hat{z}$ . Y está determinada por  $P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$

Casos particulares de puertas importantes:

- $P(\pi) = Z$ .
- $P(\pi/2) = S$ , también conocida como puerta de fase,  $\text{---}\boxed{S}\text{---}$
- $P(\pi/4) = T$ ,  $\text{---}\boxed{T}\text{---}$

Análogamente, se puede definir las matrices de rotación para los distintos ejes. Y en general, vamos a definir la puerta de rotación sobre un eje cualquiera  $\hat{n}$ , esta puerta queda determinada por  $R_{\hat{n}}(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) (n_x X + n_y Y + n_z Z)$

La idea de introducir al lector con esta puerta de rotación, además de su utilidad, se debe a que nos va a permitir presentar el siguiente teorema. Como se mencionó anteriormente, cualquier matriz unitaria  $2 \times 2$ , puede definir un operador sobre un qubit, veamos que relación hay con las rotaciones.



**Teorema 2.1: Descomposición Z-Y para un único qubit**(Nielsen,175): Sea  $U$  un operador unitario sobre un qubit. Entonces, existen números reales  $\alpha, \beta, \gamma$  y  $\delta$  tal que:

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

Existe un resultado análogo para X-Y. Este teorema va a permitir descomponer cualquier operador unitario en estas rotaciones y así, los ordenadores cuánticos actuales, pueden procesar cualquier circuito independientemente de las diferencias entre puertas que componen el circuito cuántico y las puertas que dispone el ordenador. (Martín)

Veamos ahora las puertas más importantes para más de un qubit, en particular nos vamos a fijar en 2 y 3 qubits, para 2 utilizaremos la base  $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$  y la análoga para 3 qubits.

- **CNOT**, también conocida como *puerta de control Pauli-X*. Es un caso particular de las puertas de control sobre cualquier operador. En este caso, sobre la puerta X. Este operador viene determinado por la matriz,

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

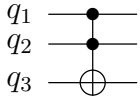
La interpretación de esta puerta es que si el primer qubit es  $|1\rangle$ , realiza la operación  $X$  sobre el segundo qubit. En general, podríamos definir la puerta controlada para un operador  $U$ , de forma análoga intercambiando  $X$  por  $U$ .

La representación de CNOT en un circuito es:  $\begin{array}{c} q_1 \text{ --- } \bullet \\ | \\ q_2 \text{ --- } \oplus \end{array}$ . En general,  $\begin{array}{c} q_1 \text{ --- } \bullet \\ | \\ q_2 \text{ --- } \boxed{U} \end{array}$

- **SWAP**, o puerta de intercambio de qubits. Con notación  $\begin{array}{c} \times \\ \text{---} \times \end{array}$ . La determinada,

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


- **Toffoli, CCNOT.** Puerta de control sobre 2 qubits  $q_1, q_2$  aplicando la puerta X a  $q_3$ .

Usaremos como notación  . Queda determinada por,

$$CCNOT = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & X \end{bmatrix}$$

Esta puerta, al igual que CNOT, podría generalizarse para un operador unitario U.

Ahora bien, todavía no hemos observado el sistema, solo hemos ido realizando operaciones unitarias, es decir, seguimos en un sistema determinista pero sin conocer realmente lo que está ocurriendo. Si recordamos otro postulado de la física cuántica, en particular el cuarto, se refería a la evolución dinámica del sistema cuando era observado. Aquí dejaremos la evolución determinista y pasaremos a la probabilística. Para poder entender que ha ocurrido en nuestro sistema y obtener resultados, vamos a necesitar medir.

Esta será la última operación que presentar, , la **medición** de un qubit. Si bien es cierto que podemos medir sobre distintas base, vamos a tomar como referencia  $\{|0\rangle, |1\rangle\}$ . Pero, ¿qué va a ser realmente la medición de un qubit?

Esta medición viene totalmente determinada por el postulado 4, obtendrá uno de los elementos donde proyectamos con cierta probabilidad. Que interpretando la fórmula vista anteriormente, esta probabilidad va a ser el cuadrado de su amplitud. Es decir, si  $|\varphi\rangle = a|0\rangle + b|1\rangle$  es el estado de un qubit antes de la medición, la probabilidad de que el resultado obtenido sea  $|0\rangle$  es  $|a|^2$  y para  $|1\rangle$  será  $|b|^2$ . Lo que hay que tener en cuenta, es que una vez realizada una medición, apartado 3, hemos modificado el qubit y a partir de entonces  $|\varphi\rangle = |0\rangle$  o  $|\varphi\rangle = |1\rangle$ .

Para almacenar esta información utilizaremos bits clásicos, por eso podremos observar como la medición se representa con la puerta anterior y la caída hacia el bit clásico donde se almacene.

Con todas estas operaciones, nos ayuda a poner los qubits en los distintos estados queridos y permitir realizar nuestros programas obteniendo una medición final. Al fin y al cabo, un programa cuántico es una sucesión de operaciones (o puertas) aplicadas sobre los qubits del sistema.

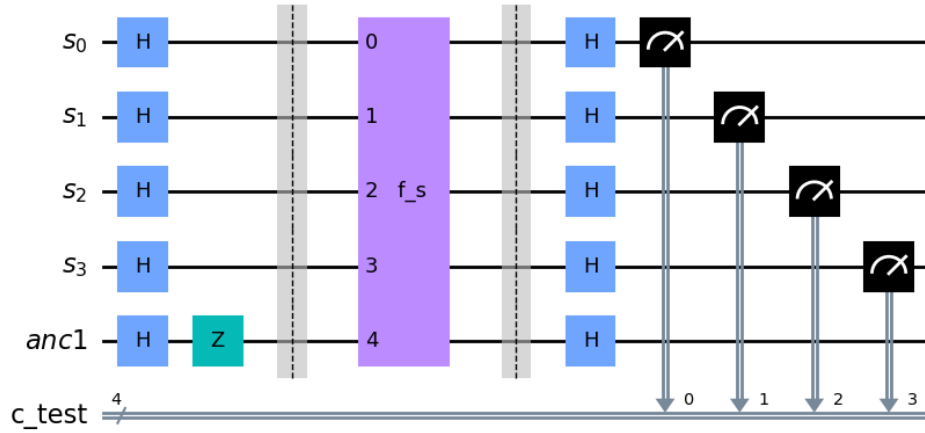


Figura 2.2: Circuito, algoritmo BV para s de longitud 4

Así que permítanme mostrarles un ejemplo que será utilizado y explicado en el capítulo siguiente. Este ejemplo, Figura 2.2, es la implementación del algoritmo de Bernstein-Vazirani para una cadena s de longitud 4. Lo que debe hacer el algoritmo es descubrir esa cadena.

Detallemos brevemente los elementos principales de este circuito:

- **Qubits:** Se representan en las primeras líneas, cada uno en una línea distinta. El circuito de la figura 2.2 se compone de 5 qubits. Entre ellos vemos un qubit llamado anc1, que es un qubit ancilla con un estado inicial conocido. Algunas veces este qubit auxiliar nos servirá como apoyo para realizar operaciones o almacenará el resultado, ya que nunca deberíamos modificar los datos originales. Se debería seguir el siguiente esquema:

$$\begin{array}{ccc} |x\rangle & \text{---} & |x\rangle \\ |y\rangle & \text{---} \boxed{U_f} \text{---} & |y \oplus f(x)\rangle \end{array}$$

- **Bits clásicos:** Se representan en la última fila, todos juntos. Podemos ver el número, en este caso 4, que nos indica el número de bits clásicos que tenemos.
- **Puertas para un qubit:** Podemos observar puertas ya conocidas como Hadamard, Z o la medición que vemos como cae hacia el cable de bits clásicos indicando en qué bit se registra la medición.

- **Puerta  $f_s$ :** Como ya mencionamos anteriormente puede haber puertas que se apliquen a varios qubits, en este caso tenemos la puerta, o bloque, que replica la función del problema de Bernstein-Vazirani. Dentro de esta puerta tenemos puertas más pequeñas como las presentadas anteriormente en este mismo apartado. En particular,  $f_s$  tiene puertas CNOT. En diversos algoritmos, se representa esta puerta como un bloque debido a que la podemos entender como un oráculo, del que no sabemos como funciona internamente.
- **Barreras:** Nos sirven como apoyo para visualizar el algoritmo y poder separar en secciones.

### 2.3.2. Simulaciones y ruido

Una vez que ya hemos visto las puertas que podemos usar en un circuito cuántico, así como un ejemplo del mismo, veamos una introducción a las distintas opciones para la ejecución. *Qiskit* traduce a bajo nivel a *qasm*, que es lo que se va a ejecutar al final. Para ello tenemos las dos opciones siguientes: (qiskit, intro cuantica)

- **Simulación**, en nuestro caso con los simuladores de IBM. Esta posibilidad nos va a ofrecer una simulación teórica de lo que ocurre en nuestro circuito. Será ejecutado a través de un simulador de IBM y nos va a permitir obtener resultados con seguridad y sin ningún problema asociado de errores o ruido. Hay que entender que las simulaciones van a tener una limitación debido a la dimensión de las matrices con las que estamos tratando, ya que el coste es exponencial.
- **Ordenador cuántico**, utilizaremos los ordenadores de IBM. La tecnología para la creación de ordenadores cuánticos más fieles sigue avanzando. Esta intenta mitigar los errores que tiene cada operación, así como los errores relacionados con el ruido del entorno que influyen en el estado del sistema modificándolo. Cada ordenador tiene su propia estructura y sus estudios sobre los errores que se cometen en cada qubit, así como calibraciones. Se puede ver en la figura 2.3 <sup>3</sup>, los errores de cada qubit así como la estructura del ordenador. Qiskit analizará el circuito que le proponemos y lo adaptará a la arquitectura del sistema elegido para la ejecución, con el objetivo de minimizar los errores cometidos.

---

<sup>3</sup>[https://qiskit.org/documentation/qc\\_intro.html](https://qiskit.org/documentation/qc_intro.html)

Ambas opciones utilizan el mismo mecanismo, repiten el proceso tantas veces como les sea requerido y muestran la frecuencia acumulada de los resultados (mediciones) o directamente las probabilidades sobre el número total de ejecuciones.

Veamos un ejemplo para entender la diferencia de resultados de utilizar un opción u otra. Para ello vamos a usar los resultados del circuito presentado en la figura 2.1 del algoritmo de Bernstein-Vazirani que se presentará en el siguiente capítulo. Este algoritmo nos debería dar un resultado único, pero veamos lo que ocurre.

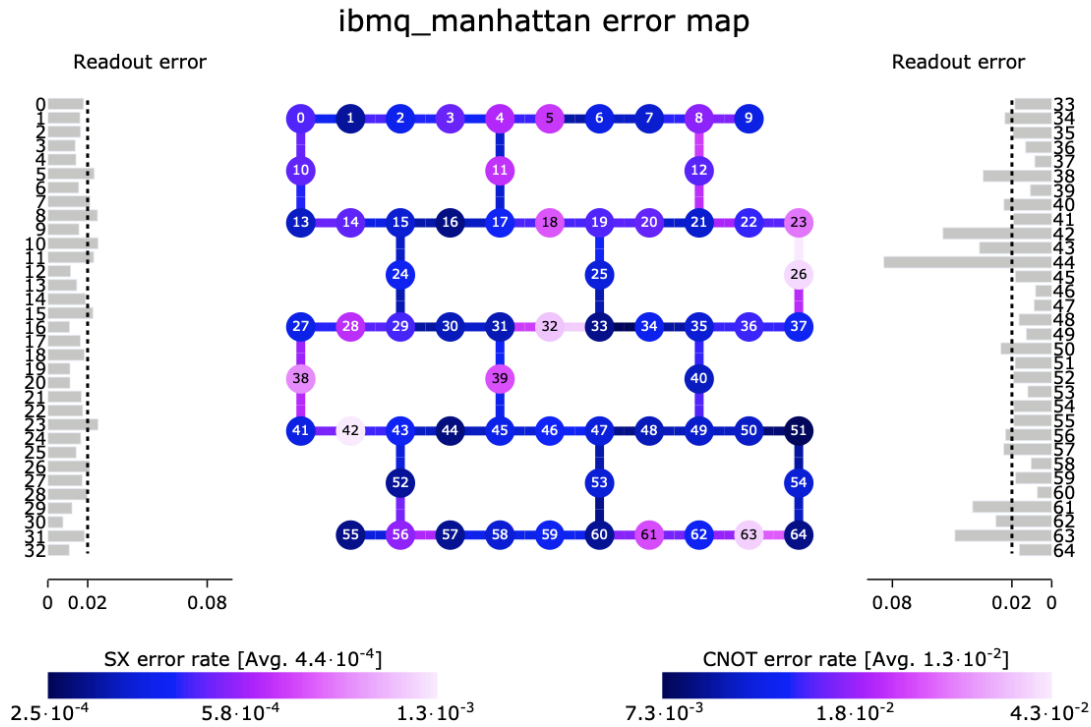


Figura 2.3: Mapa de errores del ibmq\_manhattan

Se puede observar en la figura 2.4 la diferencia entre la simulación teórica (a) y la ejecución con el sistema ibmq\_quito (b). Nuestra simulación nos ha dado únicamente el resultado querido, pero al ejecutarlo en sistema cuántico nos ha dado una variedad de resultados, aunque no todos los posibles. Si bien es cierto que el más probable, con suficiente diferencia, es idéntico al obtenido con la simulación. Cabe destacar de la figura 2.4.b que los de mayor probabilidad, una vez eliminada la solución, son aquellos en los cuales solo varía 1 dígito y por el contrario, el complementario binario no ha aparecido como resultado en ninguna de las ejecuciones realizadas.

Para acabar con esta sección, como curiosidad y por completitud, hemos mencionado antes que *Qiskit* traduce a *qasm* a la hora de ejecutar. La función utilizada en *Qiskit* para ensamblar el programa es *assemble* que devuelve un objeto de clase *QasmQobj* <sup>4</sup>. El código que utilizaría *Qiskit* para la ejecución del circuito presentado anteriormente lo podemos encontrar en *GitHub*, archivo (link y nombre).

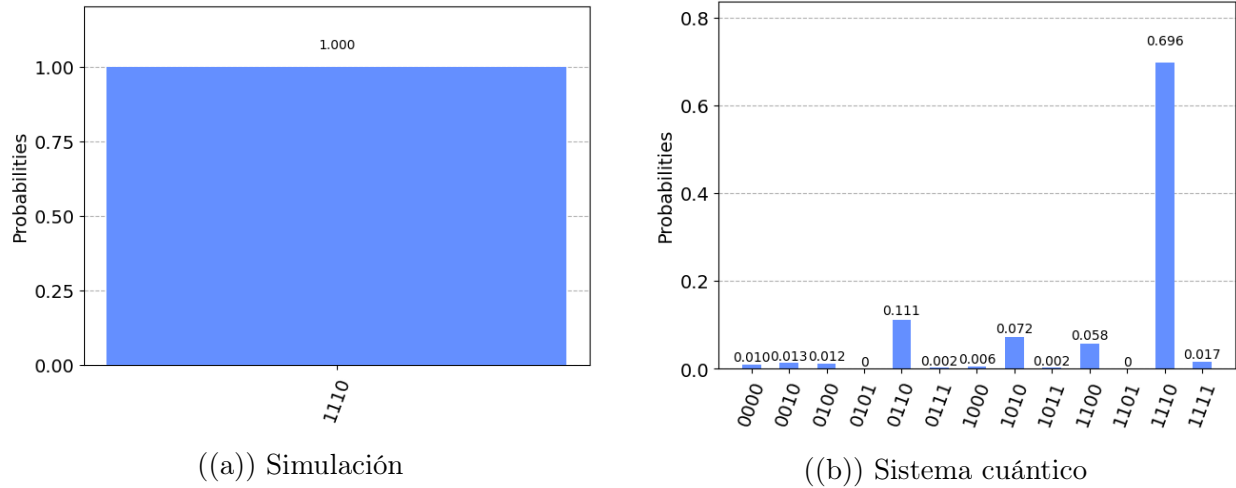


Figura 2.4: Diferencias en resultados de ejecución del algoritmo de Bernstein-Vazirani

## 2.4. Propiedades Metamórficas / Testing metamórfico

Informalmente entendemos como propiedad metamórfica aquella que podemos derivar de forma lógica de una definición o especificación. Empecemos con un ejemplo para ponernos en situación, nos vamos a fijar en la función seno,  $f(x) = \sin(x)$ .

La definición que aprendemos cuando empezamos a ver trigonometría es que el seno es la proporción entre el cateto opuesto y la hipotenusa en un triángulo rectángulo. Teniendo esta imagen la cabeza es muy fácil darse cuenta que  $\sin(x) = \sin(x + 2\pi) = \sin(\pi - x)$ . Estas son dos propiedades metamórficas de la función seno.

<sup>4</sup><https://qiskit.org/documentation/stubs/qiskit.compiler.assemble.html>

Veamos ahora que es lo que consideramos formalmente una regla o propiedad metamórfica y como llegamos a los pasos de testing metamórfico: (Metamorphic testing paper)

- **Relación metamórfica**, PM: Sea  $f : X \rightarrow Y$  una función o algoritmo. Se considera que  $\mathcal{R} \subseteq X^n \times Y^n$  es una **regla metamórfica** si es una relación entre una secuencia de entrada  $\langle x_1, x_2, \dots, x_n \rangle$  con  $n > 1$  y sus salidas correspondientes  $\langle f(x_1), f(x_2), \dots, f(x_n) \rangle$ , la cual se puede deducir de forma lógica desde el algoritmo. Es decir, es una propiedad necesaria de  $f$ .
- **Source/follow-up input**: Sea  $\mathcal{R}$  una relación metamórfica y sea  $\langle x_1, x_2, \dots, x_k \rangle$  la secuencia original con sus respectivos resultados. Denotaremos como **source input** a  $\langle x_1, x_2, \dots, x_k \rangle$  los cuales son datos definidos o caracterizados, es decir, ya conocidos. A su vez, podemos generar  $\langle x_{k+1}, x_{k+2}, \dots, x_n \rangle$ , los cuales están contruidos en base a la entrada original e incluso a la salida de esta. A esta secuencia la llamaremos **follow-up input**.
- **Grupo metamórfico de entrada**: Llamaremos **grupo metamórfico de entrada** a la secuencia definida por *source* y *follow-up input*, es decir,  $\langle x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_n \rangle$ .
- **Testing metamórfico**, TM: Sea  $f$  la función o algoritmo objetivo,  $P$  una implementación de  $f$  y  $\mathcal{R}$  una regla metamórfica de  $f$ .

Para realizar **testing metamórfico** sobre  $P$  seguiremos los siguiente pasos:

- Definimos  $\mathcal{R}'$  reemplazando  $f$  por  $P$  en  $\mathcal{R}$ .
- Dado el *source input*, generamos sus salidas según  $P$ , construimos a partir de estos el *follow-up input*  $\langle x_{k+1}, \dots, x_n \rangle$ , y obtenemos  $\langle P(x_{k+1}), \dots, P(x_n) \rangle$ .
- Estudiamos los resultados obtenidos respecto a  $\mathcal{R}'$ . Si no es satisfacible entonces diremos que  $P$  no es correcto.

La estrategia presentada anteriormente para TM, será la que se siga en la implementación de las propiedades que se obtendrán a lo largo de este documento.

Para finalizar con esta introducción vamos a revisar brevemente un par de ventajas y retos que presenta el camino que estamos tomando para probar la corrección, o más bien la no corrección, de un algoritmo.

### Ventajas:

- **Simplicidad.** El concepto e interpretación de una PM es bastante simple en comparación con otros conceptos que se utilizan dentro del campo del testing. Se ha visto en estudios, que incluso gente con poca experiencia, podrían utilizar estas técnicas en relativamente poco tiempo de forma efectiva.(ref 55,73 del paper de testing)
- **Facilidad en la implementación.** Si partimos de la ventaja anterior como es la simplicidad de la definición, podemos continuar que la implementación de este test es prácticamente seguir los pasos explicados en la definición de TM (link a la definicion).

### Retos:

- **Generación efectiva de los grupos metamórficos de entrada.** Aun se sigue estudiando que garantías tenemos en la efectividad que puede tener la elección del grupo metamórfico de entrada en la demostración de la corrección de un algoritmo y en particular, la forma en la que obtenemos nuestro *source input*, ya que *follow-up input* se genera a partir de esta. Al fin y al cabo nuestro objetivo es maximizar la identificación de errores o defectos en  $P$ .
- **Estructura del TM.** Debido a la juventud de este tipo de *testing* y la gran variedad de PM, aun no hay un acuerdo sobre una estructura definida y formal que nos proporcione seguridad en nuestras pruebas y englobe a todas las posibilidades que tenemos dentro de las PM. Aunque, si bien es cierto, ya ha ayudado a identificar diversos errores en sistemas muy estudiados con otros métodos de *testing*, como por ejemplo los compiladores GCC y LLVM de C, en los cuales encontré más de 100 fallos. (ref 50,51,78)

Los retos de TM pueden ser una buena base para todo el trabajo futuro que se puede realizar en este campo y las posibilidades que este nos puede ofrecer. Trataremos en más profundidad estas posibilidades en el apartado 5.2 de posibles trabajos futuros (link)



# Capítulo 3

## Algoritmos cuánticos

El siguiente paso en este camino hacia la unión entre la computación cuántica y el *testing* metamórfico es la creación de algoritmos o programas cuánticos que nos ayuden a resolver problemas propuestos o avanzar.

Todo el desarrollo de estos algoritmos se pueden encontrar en los libros principales (Nielson//CforComSci). Presentaremos a continuación todos los algoritmos finales que nos permiten obtener nuestros objetivos, si bien es cierto, solo vamos a presentar el camino completo de creación del algoritmo de Deutsch por su simplicidad. Esto se debe a que para alcanzar nuestro objetivo debemos ir haciendo modificaciones y cálculos sobre nuestros algoritmos hasta dar con la combinación correcta de puertas que nos permita resolverlo.

Esta sección va a seguir prácticamente la misma estructura para cada apartado, a excepción de la suma. Empezará con una introducción, seguida de la exposición del problema a resolver. Entonces nos dispondremos a presentar el algoritmo que lo resuelve, o como lo creamos, y las pruebas realizadas. Es necesario recordar que toda la programación realizada sobre estos algoritmos se encuentra en el repositorio de GitHub <https://github.com/sinugarc/TFG.git>

### 3.1. Suma

Este primer algoritmo nos va a servir como primera toma de contacto con la programación cuántica, las puertas que podemos utilizar y el uso de *Qiskit*. Veamos cual es el problema a resolver:

**Problema:** Dadas dos cadenas binarias, queremos obtener la suma binaria de ambas.

## 3.2. Deutsch

(CforCompSci) El algoritmo de Deutsch fue propuesto por David Deutsch en 1985, siendo este uno de los primeros algoritmos propuestos y en sí, el que se podría entender como uno de los problema más simples. Sea  $f : \{0, 1\} \rightarrow \{0, 1\}$ , diremos que  $f$  es **balanceada** si  $f(0) \neq f(1)$  y diremos que es **constante** si  $f(0) = f(1)$ .

**Problema:** Dada una función  $f : \{0, 1\} \rightarrow \{0, 1\}$  balanceada o constante, la cual no podemos observar su definición. Queremos determinar si esta función es constante o balanceada.

**Solución clásica:** Tenemos que evaluar  $f$  en ambos valores y comparar las soluciones.

Veamos ahora que podemos hacer con un ordenador cuántico, ¿seremos capaces de evaluar  $f$  una única vez?

Primero observamos un ejemplo particular y cómo vamos a llevarlo a un circuito cuántico. Sea  $f(0) = 1$  y  $f(1) = 0$ , buscamos una matriz unitaria que nos permita representarla en un circuito cuántico, aunque tendremos que hacer alguna modificación más. Por ahora lo que obtenemos es,

$$\begin{array}{c} \mathbf{0} \quad \mathbf{1} \\ \mathbf{0} \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \mathbf{1} \end{array}$$

Donde la columna representa la entrada y la fila la salida. Como ya se mencionó al explicar los qubits del circuito (figura 2.2), es importante conservar la entrada y por lo tanto si  $U_f$  es la caja negra que representa a  $f$ , nuestro circuito será:

$$\begin{array}{ccc} |x\rangle & \text{---} & |x\rangle \\ |y\rangle & \text{---} \boxed{U_f} \text{---} & |y \oplus f(x)\rangle \end{array}$$

Es más, si aplicáramos  $U_f$  dos veces obtendríamos la entrada:

$$\begin{array}{ccccc} & & |x\rangle & & \\ |x\rangle & \text{---} & \boxed{U_f} & \text{---} & |x\rangle \\ |y\rangle & \text{---} & \boxed{U_f} & \text{---} & |y\rangle \\ & & |y \oplus f(x)\rangle & & \end{array}$$

Esto se debe a que  $f(x) \oplus f(x) = 0$ . Pero claro, ¿qué matriz realmente representa a  $U_f$ ? Veamos como quedaría respecto a la base:

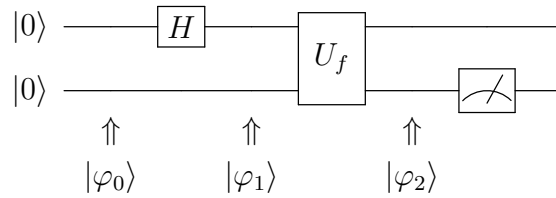
$$\begin{array}{c} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{bmatrix} & |00\rangle & |01\rangle & |10\rangle & |11\rangle \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$$

Se podría comprobar que esta matriz es su propia adjunta, por lo que es invertible y unitaria. Por lo que cumple las características necesarias para ser una puerta de un circuito cuántico.

Ahora que ya tenemos la caja negra deseada, determinada por una matriz unitaria, nos queremos poner a resolver el problema. Como comenté en la introducción de este capítulo, esta va a ser el único algoritmo al que le vamos a seguir la pista de razonamiento de principio a final, es decir, partiremos de una idea inicial y acabaremos en el algoritmo de Deutsch que resuelve este problema.

Nuestro objetivo es mejorar la complejidad de la solución del problema respecto a la solución clásica. Recordamos que necesitamos evaluar  $f$  dos veces, por lo que nuestro objetivo va a ser obtener el resultado evaluando  $f$  una única vez. Para ello nos vamos a sustentar en la superposición, para así analizar que ocurre en el  $|0\rangle$  y en el  $|1\rangle$  al mismo tiempo, es decir, vamos a tener que utilizar la puerta de Hadamard.

**Primer intento:** Tomamos nuestro primer *input* para el problema, por ejemplo  $|x\rangle = |0\rangle$  e  $|y\rangle = |0\rangle$  y creamos el primer circuito:



Donde cada  $\varphi_i$  va a representar el estado del sistema en cada momento, esto nos va a ayudar a analizar de forma determinista lo que ocurre en nuestro circuito.

Podemos resumir el circuito como  $U_f(H)|0,0\rangle$ , veamos ahora que estados tenemos en cada instante  $i$ :

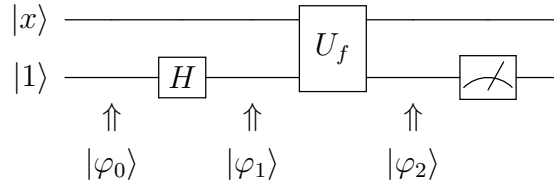
- $|\varphi_0\rangle = |0\rangle \otimes |0\rangle = |0,0\rangle$
- $|\varphi_1\rangle = (H \otimes I)|0,0\rangle = \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] |0\rangle = \frac{|0,0\rangle + |1,0\rangle}{\sqrt{2}}$
- $|\varphi_2\rangle = \frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$

Si nos fijamos en el ejemplo que expuesto al principio del algoritmo obtendríamos el siguiente estado antes de la medición:

$$|\varphi_2\rangle = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} = \frac{|0,1\rangle + |1,0\rangle}{\sqrt{2}} \quad (3.1)$$

De aquí podemos observar, que sin importar donde hagamos la medición, el resultado no va a ser concluyente, porque vamos a tener una probabilidad de 0.5 de obtener  $|0\rangle$  o  $|1\rangle$ . Es decir, este primer intento no nos sirve para nuestro objetivo.

**Segundo intento:** Veamos ahora que ocurre si en vez de poner en superposición el primer qubit, ahora ponemos el segundo, tomando  $|1\rangle$  como *input*:



Ahora iremos directamente a  $|\varphi_2\rangle$  :

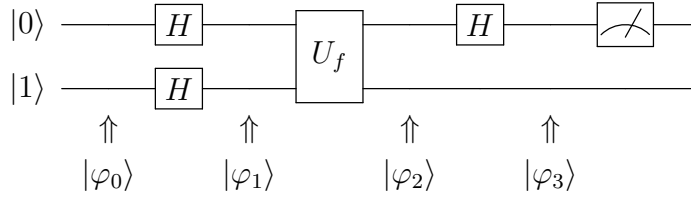
$$|\varphi_2\rangle = |x\rangle \left[ \frac{|0 \otimes f(x)\rangle - |1 \otimes f(x)\rangle}{\sqrt{2}} \right] = \begin{cases} |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{si } f(x) = 0 \\ |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{si } f(x) = 1 \end{cases} \quad (3.2)$$

Esto lo podemos resumir como:

$$|\varphi_2\rangle = (-1)^{f(x)} |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \quad (3.3)$$

Pero al igual que en intento anterior, al intentar medir en cualquiera de ambos qubits no obtendríamos ningún resultado concluyente. Esto nos va a llevar al último intento.

**Tercer intento, Algoritmo de Deutsch:** Al no ser capaces de obtener un resultado poniendo sólo un qubit en superposición, esta vez vamos a poner ambos en estado de superposición, con el *input*  $|0, 1\rangle$ . Además, vamos a medir sobre el qubit superior tras aplicar una puerta de Hadamard, que recordamos es su propia inversa. Este es el circuito final que representa al algoritmo de Deutsch:



Esto matricialmente nos queda como  $(H \otimes I) U_f (H \otimes H) |0, 1\rangle$ .

Analizamos ahora todos los estados  $|\varphi_i\rangle$ :

- $|\varphi_0\rangle = |0\rangle \otimes |1\rangle = |0, 1\rangle$
- $|\varphi_1\rangle = \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$
- $|\varphi_2\rangle = \left[ \frac{(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$

Hemos obtenido este resultado substituyendo  $|x\rangle$  en la ecuación 3.3 (link). Pero veamos que ocurre separando  $(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle$  según nuestras posibilidades:

- $f$  es constante:  $(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle = (\pm 1) \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$
- $f$  es balanceada:  $(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle = (\pm 1) \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$

- Y por último obtenemos:

$$|\varphi_3\rangle = \begin{cases} (\pm 1) |0\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{si } f(x) \text{ es constante} \\ (\pm 1) |1\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{si } f(x) \text{ es balanceada} \end{cases} \quad (3.4)$$

Por lo que hemos conseguido nuestro objetivo. Tras una única evaluación de  $f$  según la medición del primer qubit, sabremos si  $f$  es constante con medición  $|0\rangle$  o balanceada,  $|1\rangle$ . Consiguiendo así nuestro objetivo.

**Simulaciones:** En el repositorio podemos encontrar este algoritmo programado con ayuda de *Qiskit*. Una de las peculiaridades que encontramos es la forma en la que *Qiskit* nos presenta los qubits. Las cadenas se presentan al revés. Veamos un ejemplo:

**FALTA EJEMPLO DE PROGRAMACION**

### 3.3. Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa es una generalización del algoritmo de Deutsch del apartado anterior, pero ahora trabajamos con  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . El dominio se puede entender como los números escrito en binario de 0 a  $2^n - 1$ . En este caso vamos a redefinir los conceptos anteriores:

- Diremos que  $f$  es **constante** si todo el dominio va a 0 ó a 1.
- Diremos que  $f$  es **balanceada** si exactamente la mitad del dominio va a 0 y la otra mitad va a 1.

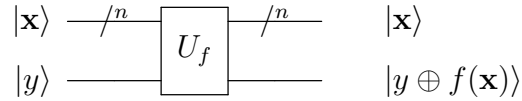
**Problema:** Dada una función  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  balanceada o constante, la cual no podemos observar su definición. Queremos determinar si esta función es constante o balanceada.

**Solución clásica:** Vamos a tener que evaluar la función  $f$  tantas veces como sea necesario. El mejor caso es tras 2 ejecuciones, ya que puede darse la situación en la que es balanceada y cada ejecución da un resultado diferente. Por otra parte, el peor caso se da con  $2^{n-1} + 1$  ejecuciones, ya que si las primeras  $2^{n-1}$  son iguales, la siguiente nos va a determinar

si es constante o balanceada. Todo esto se debe a que como hipótesis tenemos que es una de las 2 cosas.

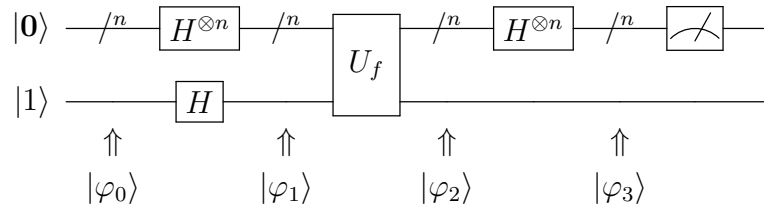
**Observación:** Hay que tener en cuenta, que si la función no es ni balanceada ni constante, este algoritmo no soluciona el problema y los resultados no son concluyentes.

Vamos a partir del circuito inicial, donde  $U_f$  determina la función  $f$ :



Hay que destacar que  $|\mathbf{x}\rangle = |x_0x_1\dots x_{n-1}\rangle$ , por eso posteriormente escribimos  $/n$ , para indicar que son  $n$  qubits.

**Algoritmo de Deutsch-Jozsa:**



Que en término de matrices es:  $(H^{\otimes n} \otimes I) U_f (H^{\otimes n} \otimes H) |0, 1\rangle$

Antes de analizar los estados en cada  $|\varphi_i\rangle$ , vamos a estudiar que ocurre cuando tenemos la puerta  $H^{\otimes n}$  y como se comporta sobre los diferentes estados, así como la notación que utilizamos, ya que esta será necesaria en el análisis de los  $|\varphi_i\rangle$ .

El estudio sobre las características de  $H^{\otimes n}$  se pueden estudiar en profundidad en (Cfor-CompSci), aquí resumiremos los resultados principales que nos van a permitir profundizar en los circuitos cuánticos, ya que la aplicación de esta puerta es muy común.

Antes de meternos en  $H^{\otimes n}$ , vamos a definir un producto interno en  $\mathbb{Z}_2$ :

$$\langle , \rangle : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$$

Si  $\mathbf{x} = x_0x_1\dots x_{n-1}$  e  $\mathbf{y} = y_0y_1\dots y_{n-1}$ , entonces:

$$\begin{aligned}\langle \mathbf{x}, \mathbf{y} \rangle &= \langle x_0x_1\dots x_{n-1}, y_0y_1\dots y_{n-1} \rangle \\ &= (x_0 \wedge y_0) \oplus (x_1 \wedge y_1) \oplus \dots \oplus (x_{n-1} \wedge y_{n-1})\end{aligned}$$

Donde  $\wedge$  es la operación lógica AND.

También definimos,  $\mathbf{x} \oplus \mathbf{y} = x_0 \oplus y_0, x_1 \oplus y_1, \dots, x_{n-1} \oplus y_{n-1}$ .

Con esto obtenemos las siguiente propiedades del producto interno:

- $\langle \mathbf{x} \oplus \mathbf{x}', \mathbf{y} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \oplus \langle \mathbf{x}', \mathbf{y} \rangle$
- $\langle \mathbf{x}, \mathbf{y} \oplus \mathbf{y}' \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \oplus \langle \mathbf{x}, \mathbf{y}' \rangle$
- $\langle 0 \cdot \mathbf{x}, \mathbf{y} \rangle = \langle 0, \mathbf{y} \rangle = 0$
- $\langle \mathbf{x}, 0 \cdot \mathbf{y} \rangle = \langle \mathbf{x}, 0 \rangle = 0$

Una vez definidas estas operaciones y propiedades vamos a poder definir la matriz que caracteriza a  $H^{\otimes n}$ . Cabe recordar que cada fila y cada columna representa un elemento de la base, que se corresponde con la representación binaria de la fila y la columna desde 0:

$$H^{\otimes n}[\mathbf{i}, \mathbf{j}] = \frac{1}{\sqrt{2^n}} (-1)^{\langle \mathbf{i}, \mathbf{j} \rangle} \quad (3.5)$$

Desde aquí podemos obtener las 2 expresiones que utilizaremos a continuación:

$$H^{\otimes n}|\mathbf{0}\rangle = H^{\otimes n}[-, \mathbf{0}] = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle \quad (3.6)$$

$$H^{\otimes n}|\mathbf{y}\rangle = H^{\otimes n}[-, \mathbf{y}] = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\langle \mathbf{x}, \mathbf{y} \rangle} |\mathbf{x}\rangle \quad (3.7)$$

Ahora ya si que vamos a observar los estados que se producen en los distintos momentos del circuito:

- $|\varphi_0\rangle = |\mathbf{0}, 1\rangle$
- $|\varphi_1\rangle = \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle}{\sqrt{2^n}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$



- $|\varphi_2\rangle = \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle}{\sqrt{2^n}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$

- Ahora tenemos que superponer la superposición, pero en este caso no es la inversa ya que se aplica sobre otros estados  $|\mathbf{z}\rangle$ :

$$\begin{aligned}
|\varphi_3\rangle &= \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{\langle \mathbf{z}, \mathbf{x} \rangle} |\mathbf{z}\rangle}{2^n} \right] \left[ \frac{|0\rangle - |1\rangle}{2^n} \right] \\
&= \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} (-1)^{\langle \mathbf{z}, \mathbf{x} \rangle} |\mathbf{z}\rangle}{2^n} \right] \left[ \frac{|0\rangle - |1\rangle}{2^n} \right] \\
&= \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} \sum_{\mathbf{z} \in \{0,1\}^n} (-1)^{f(\mathbf{x}) \oplus \langle \mathbf{z}, \mathbf{x} \rangle} |\mathbf{z}\rangle}{2^n} \right] \left[ \frac{|0\rangle - |1\rangle}{2^n} \right]
\end{aligned} \tag{3.8}$$

El último paso a analizar y que nos determinará la validez o el que esperamos de este algoritmo es al medición de los primeros  $n$ -qubits. Otra manera de estudiar la medición es directamente preguntarnos cual es la probabilidad de que  $|\varphi_3\rangle$  colapse a  $|\mathbf{0}\rangle$  al realizar la medición. Esto es análogo a  $\mathbf{z} = \mathbf{0}$ , que por las propiedades estudiadas anteriormente  $\langle \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{0}, \mathbf{x} \rangle = 0$ . Con este resultado, si volvemos a la ecuación 3.5 (link):

$$|\varphi_3\rangle = \left[ \frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{0}\rangle}{2^n} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \tag{3.9}$$

Veamos cual sería el estado del primer qubit dependiendo de de nuestras posibilidades:

- $f$  es constante:  $\frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{0}\rangle}{2^n} = \frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{0}\rangle}{2^n} = \frac{(\pm 1) 2^n |\mathbf{0}\rangle}{2^n} = (\pm 1) |\mathbf{0}\rangle$
- $f$  es balanceada:  $\frac{\sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{0}\rangle}{2^n} = \frac{0 |\mathbf{0}\rangle}{2^n} = 0 |\mathbf{0}\rangle$

Por lo que podemos concluir que este algoritmo resuelve nuestro problema con una única evaluación de  $f$ . Si el resultado es  $|\mathbf{0}\rangle$ ,  $f$  es constante y si es otro cualquiera,  $f$  es balanceada. Por esta razón la observación que realizamos anteriormente, ya que nos sería imposible identificar si la función no es ni constante ni balanceada. Y esto explica porque es una de nuestras hipótesis.

### **3.4. Bernstein-Vazirani**

Algoritmo de Bernstein- Vazirani

### **3.5. Simon**

Algoritmo de Simon

### **3.6. QFT**

Algoritmo de la transformada cuántica de Fourier

### **3.7. QEP**

Algoritmo cuántico de estimación de fase

### **3.8. Grover**

Algoritmo cuántico de búsqueda de Grover

# Capítulo 4

## Propiedades metamórficas

4.1. DJ

4.2. BV

4.3. Simon

# Capítulo 5

## Conclusiones y trabajo futuro

### 5.1. Conclusiones

Hilbert

### 5.2. Trabajo futuro

Postulados