

Introducción a la computación cuántica

Salvador E. Venegas Andraca

Tecnológico de Monterrey Campus Estado de México

<http://www.cem.itesm.mx/dia/escuelaverano>

svenegas@itesm.mx, sva@mindsofmexico.org y salvador.venegas@gmail.com

Julio de 2007

1 Introducción

La mecánica cuántica es la rama de la física que describe el comportamiento de la naturaleza a escalas muy pequeñas (por ejemplo, el comportamiento de los átomos). La teoría de la computación se encarga de estudiar si un problema es susceptible de ser resuelto utilizando una computadora, así como la cantidad de recursos (tiempo, energía) que se debe invertir en caso de existir solución. En consecuencia, la computación cuántica hace uso de la mecánica cuántica con el objetivo de incrementar nuestra capacidad computacional para el procesamiento de información y solución de problemas. Por otra parte, la teoría de la información cuántica estudia los métodos, capacidades y límites que las leyes de la física imponen en la transmisión y recuperación de información.

El estudio formal de la computación cuántica comenzó con las preguntas que Richard Feynman planteó sobre dos temas: 1) la posibilidad de simular sistemas cuánticos, y 2) las leyes de la física que caracterizan al proceso de calcular [92, 93]. A partir de ese trabajo, la computación cuántica ha avanzado a paso firme; por ejemplo, se ha definido formalmente la estructura de una computadora cuántica [3], se han encontrado resultados espectaculares como el algoritmo de Shor [4] (capaz de factorizar un número entero muy largo en tiempo razonable utilizando una computadora cuántica [4, 7]) y el algoritmo de Grover [5] (este algoritmo encuentra elementos en conjuntos desordenados de forma más eficiente que cualquier algoritmo posible ejecutado en computadoras convencionales [5, 7]), y se ha diseñado una teoría y práctica de la criptografía usando las propiedades de la física cuántica [6]. En el futuro mediato, la computación cuántica tendrá gran impacto en la industria de la computación y el desarrollo de protocolos de criptografía y seguridad computacional [12–14].

La computación y la información cuánticas representan un reto teórico y experimental por la cantidad y complejidad de problemas a resolver. A pesar de dichos retos, los avances realizados hasta ahora permiten ya pensar en aplicaciones de esta disciplina en áreas del conocimiento tales como la Inteligencia Artificial, el Reconocimiento de Patrones [16–23, 91] y la Bioinformática [24–26]. De hecho, la computación cuántica ha dado ya sus primeros frutos en la industria pues desde 2003 hay un sistema de criptografía cuántica disponible en el mercado [27, 30], y los experimentos realizados en diversos laboratorios (por ejemplo [28, 29]) ofrecen resultados muy prometedores.

Universidades de prestigio internacional como Oxford [31], Cambridge [32], Harvard [33], Viena [34], Caltech [35] y MIT [36] llevan a cabo investigación en estas disciplinas con financiamiento público (Estados Unidos, Reino Unido, Alemania, Francia, Australia y Brasil, entre otros gobiernos) y privado (Lucent Technologies [37], Hewlett Packard [38], IBM [39] y Fujitsu [40], entre otras empresas). La computación cuántica es importante para los gobiernos debido a que la creación de computadoras cuánticas implicará la existencia de escenarios de alta vulnerabilidad (las computadoras cuánticas serán capaces de descifrar, en horas o minutos, los códigos de encriptación que hoy tomaría mucho tiempo resolver. En consecuencia, la confidencialidad de mensajes electrónicos se verá afectada). Por su parte, la industria del cómputo ha invertido recursos sustanciales en esta especialidad debido a las ventajas en rapidez y seguridad de datos que la tecnología cuántica traerá consigo, además de que la miniaturización de componentes electrónicos implica la aparición de efectos cuánticos.

De lo anterior se deduce que las oportunidades de desarrollo profesional para científicos e ingenieros en estas disciplinas son amplias y prometedoras. Además, es importante subrayar que, dado el alto impacto que la computación e información cuánticas tendrán en el quehacer científico y en la

vida diaria del ser humano, el trabajo desarrollado por algunos científicos en estas disciplinas ha sido ya merecedor de premios internacionales como el Príncipe de Asturias 2006 [41, 42].

En el ámbito nacional, la computación y la información cuánticas ha despertado el interés de varios miembros de la vida científica e intelectual mexicana. Tanto en publicaciones académicas [43, 44] como en medios de comunicación masivos [45–54], estos campos del conocimiento y su posible impacto en nuestra sociedad han sido tratados desde distintas ópticas.

A pesar de contar con recursos humanos calificados en distintas universidades y centros de investigación (UNAM, INAOE y CINVESTAV, entre otras), y de que algunas instituciones han trabajado en la formación de recursos humanos (por ejemplo, la primer escuela mexicana de verano en computación cuántica, llevada a cabo en la UADY y el CINVESTAV-Mérida, organizada por los doctores Luis Alberto Muñoz Ubando, Salvador Venegas Andraca, Romeo de Coss y Juan Luis Díaz de León [55, 56]), México no tiene grupos sólidos de investigación en computación e información cuánticas.

Como parte de la estrategia para integrar a México en este campo del conocimiento, el Tecnológico de Monterrey Campus Estado de México y la universidad de Leeds (Reino Unido) han unido esfuerzos para organizar la *Segunda Escuela Mexicana de Verano en Computación e Información Cuánticas*. El presente documento es un texto diseñado para impartir un curso introductorio a la computación cuántica durante la primera semana de nuestra escuela.

2 Álgebra lineal

El álgebra lineal es ampliamente utilizada en física. En particular, este campo de las matemáticas se usa en la formulación de la mecánica cuántica y, en consecuencia, en la computación y la información cuánticas.

En esta sección estudiaremos varios elementos del álgebra lineal necesarios para una introducción seria a la computación cuántica. Para profundizar en los conceptos aquí revisados se recomienda consultar las obras [7], [8], [9], [10] y [11].

2.1 Campo, espacio vectorial, producto interno y norma

Definición 2.1. Campo. Un campo es un conjunto \mathbf{F} con dos operaciones, llamadas multiplicación y adición, que satisface los axiomas siguientes:

- 1) $\forall x, y \in \mathbf{F} \Rightarrow x + y \in \mathbf{F}$
- 2) $\forall x, y \in \mathbf{F} \Rightarrow x + y = y + x$
- 3) $\forall x, y, z \in \mathbf{F} \Rightarrow x + (y + z) = (x + y) + z$
- 4) $\exists! 0 \in \mathbf{F}$ tal que $\forall x \in \mathbf{F} \Rightarrow x + 0 = 0 + x = x$
- 5) Para cada $x \in \mathbf{F} \exists! -x \in \mathbf{F}$ tal que $x + (-x) = -x + x = 0$
- 6) $\forall x, y \in \mathbf{F} \Rightarrow xy \in \mathbf{F}$
- 7) $\forall x, y \in \mathbf{F} \Rightarrow xy = yx$
- 8) $\forall x, y, z \in \mathbf{F} \Rightarrow x(yz) = (xy)z$
- 9) $\exists! 1 \in \mathbf{F}$ tal que $\forall x \in \mathbf{F} \Rightarrow x1 = 1x = x$
- 10) Para cada $x \in \mathbf{F} - \{0\} \exists! x^{-1} \in \mathbf{F}$ tal que $xx^{-1} = x^{-1}x = 1$
- 11) $\forall x, y, z \in \mathbf{F} \Rightarrow x(y + z) = xy + xz$

Ejercicio 2.1. Sea $\mathbb{C} = \{a + bi \mid a, b \in \mathbb{R} \text{ e } i = \sqrt{-1}\}$ el conjunto de los números complejos.

Sean $z_1 = a + bi$ y $z_2 = c + di$ números complejos. Definimos entonces:

1) $z_1 + z_2 = (a + bi) + (c + di) = (a + c) + (b + d)i$

2) $z_1 z_2 = (a + bi)(c + di) = (ac - bd) + (bc + ad)i$

3) $z_1^* = a - bi$

Use 1) y 2) para demostrar que \mathbb{C} es un campo.

Definición 2.2. Espacio vectorial. Sea \mathbb{V} un conjunto cualquiera, asociado con un campo \mathbf{F} (los elementos de \mathbf{F} son llamados *escalares*). En las siguientes líneas daremos las condiciones que debe cumplir \mathbb{V} para recibir el nombre de *espacio vectorial*, pero antes haremos del conocimiento del lector algunas aclaraciones:

a) Denotaremos a los elementos de \mathbb{V} con la nomenclatura $|\cdot\rangle$, donde el punto \cdot es sustituido por una letra del alfabeto griego o castellano. Además, denotaremos a los elementos de \mathbf{F} con simples letras del alfabeto castellano.

b) La notación generalmente utilizada para describir elementos de \mathbb{V} es \vec{x} . Sin embargo, en computación cuántica se acostumbra escribir $|x\rangle$ en lugar de \vec{x} por motivos que expondremos a lo largo de las siguientes páginas. Baste decir por el momento que el símbolo $|x\rangle$ forma parte de la notación de Dirac.

Entremos entonces a la definición de un espacio vectorial. Definimos sobre \mathbb{V} dos operaciones, la *adición* y la *multiplicación escalar*.

- Por *adición* se entiende una regla que asocia a cada par de elementos $|u\rangle, |v\rangle \in \mathbb{V}$ con un objeto $|u\rangle + |v\rangle \in \mathbb{V}$, denominado suma de $|u\rangle$ y $|v\rangle$.

- Por *multiplicación escalar* se entiende una regla que asocia a cada $k \in \mathbf{F}$ y cada objeto $|u\rangle \in \mathbb{V}$ con un objeto $k|u\rangle \in \mathbb{V}$, denominado múltiplo escalar de $|u\rangle$ por k .

Si $\forall |u\rangle, |v\rangle, |w\rangle \in \mathbb{V}$ y $\forall k, l \in \mathbf{F}$ se satisfacen los axiomas 1 al 10 que se presentan enseguida $\Rightarrow \mathbb{V}$ se llama *espacio vectorial* y sus elementos *vectores*. \mathbb{V} también puede recibir el nombre de *espacio vectorial lineal*.

1) $|u\rangle, |v\rangle \in \mathbb{V} \Rightarrow |u\rangle + |v\rangle \in \mathbb{V}$

2) $|u\rangle + |v\rangle = |v\rangle + |u\rangle$

3) $|u\rangle + (|v\rangle + |w\rangle) = (|u\rangle + |v\rangle) + |w\rangle$

4) $\exists \mathbf{0} \in \mathbb{V}$, llamado *vector cero* o *elemento neutro aditivo*, tal que $|u\rangle + \mathbf{0} = \mathbf{0} + |u\rangle = |u\rangle$ (observe la notación atípica).

5) Para cada $|u\rangle \in \mathbb{V} \exists! -|u\rangle \in \mathbb{V}$ tal que $|u\rangle + (-|u\rangle) = -|u\rangle + |u\rangle = \mathbf{0}$

6) $\forall |u\rangle \in \mathbb{V}, k \in \mathbf{F} \Rightarrow k|u\rangle \in \mathbb{V}$

7) $(k + l)|u\rangle = k|u\rangle + l|u\rangle$

8) $k(l|u\rangle) = (kl)|u\rangle$

9) $k(|u\rangle + |v\rangle) = k|u\rangle + k|v\rangle$

10) $1|u\rangle = |u\rangle$, siendo 1 el elemento neutro multiplicativo de \mathbf{F} .

Nota. Si el campo en cuestión es \mathbb{C} (\mathbb{R}) entonces \mathbb{V} recibe el nombre de espacio vectorial complejo (real).

Ejercicio 2.2. Demuestre que:

- a) $\mathbb{C}^n = \{|u\rangle = (u_1, u_2, \dots, u_n)^t \mid u_1, u_2, \dots, u_n \in \mathbb{C}\}$ es un espacio vectorial complejo.
 b) $\mathbb{M}_n(\mathbb{C})$, el conjunto de las matrices de orden n y con entradas complejas, es un espacio vectorial complejo.

Definición 2.3. Espacio vectorial complejo con producto interno. En la geometría euclídea, las propiedades que cuentan con la posibilidad de medir longitudes de segmentos rectilíneos y ángulos formados por rectas se llaman propiedades *métricas*. Además, en el estudio elemental de \mathbb{R}^n se utiliza el concepto de *producto escalar* para calcular longitudes y ángulos. Ahora extenderemos estas ideas a espacios vectoriales generales a través de la definición del *producto interno*.

Sea \mathbb{V} un espacio vectorial complejo. El espacio \mathbb{V} se conoce como espacio vectorial complejo con producto interno si es posible definir la función *producto interno* $(\cdot, \cdot) : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{C}$, la cual obedece los axiomas plasmados en las siguientes líneas:

- $\forall |u\rangle, |v\rangle, |w\rangle, |x\rangle \in \mathbb{V}, \alpha, \beta \in \mathbb{C} \Rightarrow$
 1) $(|u\rangle, |u\rangle) \geq 0$ y $(|u\rangle, |u\rangle) = 0 \Leftrightarrow |u\rangle = \mathbf{0}$
 2) $(|u\rangle, |v\rangle) = (|v\rangle, |u\rangle)^*$
 3) $(|u\rangle, \alpha|v\rangle + \beta|w\rangle) = \alpha(|u\rangle, |v\rangle) + \beta(|u\rangle, |w\rangle)$ ¹

Ejercicio 2.3. Sean $|x\rangle = (x_1, x_2, \dots, x_n)^t, |y\rangle = (y_1, y_2, \dots, y_n)^t \in \mathbb{C}^n \Rightarrow$ definimos el producto interior en \mathbb{C}^n de la siguiente manera:

$(|x\rangle, |y\rangle) = (x_1^*y_1 + x_2^*y_2 + \dots + x_n^*y_n)^{1/2} = (\sum_{i=1}^n x_i^*y_i)^{1/2}$, donde x_i^* es el conjugado de x_i . Demuestre que, con esta definición de producto interno, \mathbb{C}^n es un espacio vectorial complejo con producto interno. El espacio vectorial \mathbb{C}^n , con el producto interno definido en este ejercicio, recibe el nombre de *espacio vectorial n-dimensional complejo con producto interno*.

Definición 2.4. Espacio vectorial normado. Sea \mathbb{V} un espacio vectorial complejo con producto interno. Definimos la **norma** $\| |x\rangle \|$ de un elemento $|x\rangle \in \mathbb{V}$ como $\| |x\rangle \| = \sqrt{(|x\rangle, |x\rangle)}$, la cual debe satisfacer las siguientes propiedades:

- 1) $\forall |x\rangle \in \mathbb{V} \Rightarrow \| |x\rangle \| \geq 0$, y $\| |x\rangle \| = 0 \Leftrightarrow |x\rangle = \mathbf{0}$
 2) $\forall |x\rangle \in \mathbb{V}$ y $\forall \alpha \in \mathbb{C} \Rightarrow \| \alpha|x\rangle \| = |\alpha| \| |x\rangle \|$
 3) $\| |x\rangle + |y\rangle \| \leq \| |x\rangle \| + \| |y\rangle \|$

Si para cada $|x\rangle \in \mathbb{V}$ es posible calcular la norma correspondiente $\| |x\rangle \| \Rightarrow \mathbb{V}$ recibe el nombre de *espacio vectorial normado*.

Ejercicio 2.4. Demuestre que \mathbb{C}^n es un espacio vectorial normado.

Definición 2.5. Normalidad, ortogonalidad y ortonormalidad. Sea \mathbb{V} un espacio vectorial \Rightarrow

- Dos elementos $|x\rangle, |y\rangle \in \mathbb{V}$ se llaman *ortogonales* si su producto interior es cero.
- Un subconjunto $S \subset \mathbb{V}$ es un *conjunto ortogonal* $\Leftrightarrow \forall |x\rangle, |y\rangle \in S, |x\rangle \neq |y\rangle \Rightarrow (|x\rangle, |y\rangle) = 0$.
- Un conjunto ortogonal se llama *ortonormal* si cada uno de sus elementos tiene norma igual a 1.

Observación. El elemento cero $\mathbf{0} \in \mathbb{V}$ es ortogonal a todo elemento en \mathbb{V} .

¹En textos de matemática pura es común que esta tercera regla se escriba así :

$(|u\rangle, \alpha|v\rangle + \beta|w\rangle) = \alpha^*(|u\rangle, |v\rangle) + \beta^*(|u\rangle, |w\rangle)$, donde α^* y β^* son los conjugados de α y β respectivamente. En computación cuántica se utiliza la definición 2.3.

2.2 Subespacios, combinación lineal, generación de espacios, independencia lineal y bases

Dado el carácter elemental de los temas revisados en esta sección, se invita al lector interesado en estudiar las demostraciones correspondientes, a consultar las obras [8], [9], [10] y [11].

Notación. En las siguientes líneas, el símbolo \mathbb{V} denotará a un espacio vectorial sobre un campo F , i.e. $\mathbb{V} = \mathbb{V}(F)$.

Definición 2.6. Subespacio. Sea \mathbb{W} un subconjunto de un espacio vectorial \mathbb{V} . \mathbb{W} se denomina **subespacio** de \mathbb{V} si \mathbb{W} es un espacio vectorial bajo la suma vectorial y multiplicación escalar definidas sobre \mathbb{V} .

Teorema 1. Sea \mathbb{W} un conjunto formado por uno o más vectores de un espacio vectorial \mathbb{V} . \mathbb{W} es un subespacio de $\mathbb{V} \Leftrightarrow$ se cumplen las siguientes condiciones:

- a) Si $|u\rangle, |v\rangle \in \mathbb{W} \Rightarrow |u\rangle + |v\rangle \in \mathbb{W}$
- b) Si $k \in F$ y $|u\rangle \in \mathbb{W} \Rightarrow k|u\rangle \in \mathbb{W}$

Ejercicio 2.5. Sean el espacio vectorial $\mathbb{M}_n(\mathbb{C})$ y $S = \{A \in \mathbb{M}_n(\mathbb{C}) | A = A^t, \text{ i.e. } A \text{ es igual a su transpuesta}\}$. Demuestre que S , el conjunto de las matrices simétricas, es un subespacio de $\mathbb{M}_n(\mathbb{C})$.

Definición 2.7. Combinación lineal. Sean \mathbb{V} un espacio vectorial, $S = \{|v_i\rangle \in \mathbb{V}, i \in \{1, 2, \dots, n\}\}$ un subconjunto de \mathbb{V} y $E = \{c_i, i \in \{1, 2, \dots, n\}\}$ un subconjunto del campo F . Entonces, todo $|x\rangle$ que se exprese en la forma

$$|x\rangle = \sum_{i=1}^n c_i |v_i\rangle$$

se denominará **combinación lineal** de elementos de S .

Teorema 2. Sea \mathbb{V} un espacio vectorial. Definimos a los conjuntos $S = \{|v_i\rangle \in \mathbb{V} | i \in \{1, 2, \dots, n\}\}$ y $T = \{|x\rangle \in \mathbb{V} | |x\rangle = \sum_{i=1}^n c_i |v_i\rangle, c_i \in F, i \in \{1, 2, \dots, n\}\}$. Entonces, T es un subespacio de \mathbb{V} .

Definición 2.8. Generación de espacios. Sean \mathbb{V} un espacio vectorial y los conjuntos $S = \{|v_i\rangle \in \mathbb{V} | i \in \{1, 2, \dots, n\}\}$ y $T = \{|x\rangle \in \mathbb{V} | |x\rangle = \sum_{i=1}^n c_i |v_i\rangle, c_i \in F, i \in \{1, 2, \dots, n\}\}$. Entonces, T es el **conjunto generado** por S .

Nota: Se acostumbra escribir a T de la siguiente forma: $T = \text{gen}\{|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle\}$.

Ejercicio 2.6. Los vectores $\hat{i} = (1, 0, 0)^t$, $\hat{j} = (0, 1, 0)^t$ y $\hat{k} = (0, 0, 1)^t$, generan a \mathbb{R}^3 y a \mathbb{C}^3 .

Definición 2.9. Generación de un espacio vectorial. Sean \mathbb{V} un espacio vectorial y $S = \{|v_i\rangle \in \mathbb{V} | i \in \{1, 2, \dots, n\}\}$. Se dice que S **genera** a \mathbb{V} si y sólo si $\forall |x\rangle \in \mathbb{V} \Rightarrow |x\rangle \in \text{gen}\{S\}$.

Ejercicio 2.7. Espacios generadores.

1) $S = \{1, x_1, x_2, \dots, x_n\}$ genera a $\mathbb{P}_n(x)$.

2) $\left\{ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\}$ genera a $\mathbb{M}_2(F)$.

3) Encuentre un conjunto generador para $\mathbb{M}_n(F)$.

Definición 2.10. Independencia y dependencia lineal. Sean \mathbb{V} un espacio vectorial, S un conjunto definido por $S = \{|v_i\rangle \in \mathbb{V} | i \in \{1, 2, \dots, n\}\}$ y $c_1, c_2, \dots, c_n \in F$. La ecuación vectorial

$$\sum_{i=1}^n k_i |v_i\rangle = k_1 |v_1\rangle + k_2 |v_2\rangle + \dots + k_n |v_n\rangle = \mathbf{0} \quad (1)$$

tiene por lo menos una solución, a saber:

$$k_1 = k_2 = \dots = k_n = 0 \quad (2)$$

Si la ecuación (2) es la única solución de la ecuación (1) $\Rightarrow S$ es un conjunto **linealmente independiente**.

Si, además de la ecuación (1), existen otras soluciones para la ecuación (2) $\Rightarrow S$ es un conjunto **linealmente dependiente**.

Ejercicio 2.8. Demuestre que los vectores $\hat{i} = (1, 0, 0)^t$, $\hat{j} = (0, 1, 0)^t$ y $\hat{k} = (0, 0, 1)^t$ son linealmente independientes.

Definición 2.11. Base. Sean \mathbb{V} un espacio vectorial y $S = \{|v_i\rangle \in \mathbb{V}, i \in \{1, 2, \dots, n\}\}$. El conjunto S forma una **base** de $\mathbb{V} \Leftrightarrow$

a) $\{|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle\}$ es un conjunto linealmente independiente, y

b) $\{|v_1\rangle, |v_2\rangle, \dots, |v_n\rangle\}$ genera a \mathbb{V} .

Ejercicio 2.9. Bases vectoriales.

1) $\{\hat{i}, \hat{j}, \hat{k}\}$ es una base de \mathbb{R}^3 .

2) Sean \mathbb{R}^n con campo \mathbb{R} y $B = \{|e_1\rangle, \dots, |e_n\rangle \mid |e_i\rangle = (0, \dots, 0, 1, 0, \dots, 0)$, con 1 en la i -ésima entrada y 0 en las restantes, $\forall i \in \{1, 2, \dots, n\}$ Entonces, B es una base de \mathbb{R}^n . La misma base puede ser utilizada en \mathbb{C}^n .

Teorema 3. Todo conjunto de n vectores en \mathbb{R}^n linealmente independientes es una base de \mathbb{R}^n .

Teorema 4. Sea $\{|v_1\rangle, \dots, |v_n\rangle\}$ una base de \mathbb{V} y $|v\rangle \in \mathbb{V} \Rightarrow \exists!$ conjunto $\{c_1, \dots, c_n\} \subset F$ tal que $|v\rangle = \sum_{i=1}^n c_i |v_i\rangle$

Teorema 5. Si $\{|u_1\rangle, |u_2\rangle, \dots, |u_n\rangle\}$ y $\{|v_1\rangle, |v_2\rangle, \dots, |v_m\rangle\}$ son bases de $\mathbb{V} \Rightarrow m = n$, i.e. cualesquiera dos bases en un espacio vectorial \mathbb{V} poseen el mismo número de vectores.

Teorema 6. Sea $B_1 = \{|u_1\rangle, \dots, |u_n\rangle\}$ una base de \mathbb{V} . Supongamos además la existencia de n vectores $|x_1\rangle, \dots, |x_n\rangle \in \mathbb{V}$ expresados en la base B_1 :

$$|x_1\rangle_{B_1} = (a_{11}, \dots, a_{n1})^t, |x_2\rangle_{B_1} = (a_{12}, \dots, a_{n2})^t, \dots, |x_n\rangle_{B_1} = (a_{1n}, \dots, a_{nn})^t$$

A partir de estos vectores $|x_i\rangle_{B_1}$ construyamos una matriz A :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Entonces, $|x_1\rangle, |x_2\rangle, \dots, |x_n\rangle$ son linealmente independientes $\Leftrightarrow \det(A) \neq 0$.

Observación. Deseamos subrayar que la existencia de los elementos de un espacio vectorial $|x\rangle \in \mathbb{V}$ es *independiente* de cualquier base que uno escoja para representar al vector $|x\rangle$. Lo único que de $|x\rangle$ depende respecto de la base B que uno escogiese son las *componentes* del vector, y no el vector en sí mismo (o el número de componentes que deba tener en cualquier representación).

Definición 2.12. Dimensión de un espacio vectorial. Sea \mathbb{V} un espacio vectorial y B una base de $\mathbb{V} \Rightarrow \dim \mathbb{V} = \#(B)$, i.e. la dimensión de \mathbb{V} es la cardinalidad de B .

2.3 Espacios de Hilbert

Definición 2.13. Estructura algebraica, homomorfismo e isomorfismo.

- *Estructura algebraica.* Conjunto cerrado sobre una o más operaciones, el cual satisface ciertos axiomas.

- *Homomorfismo.* Un homomorfismo $H : A \rightarrow B$ es una función entre dos estructuras algebraicas del mismo tipo (por ejemplo, dos espacios vectoriales) tal que H conserva/preserva la estructura de A en B , por ejemplo elementos identidad, elementos inversos y operaciones binarias.

- *Isomorfismo.* La función $f : A \rightarrow B$ es un isomorfismo si tanto f como su inversa f^{-1} son homomorfismos, i.e. mapeos que conservan la estructura de los conjuntos A y B .

La idea central en esta definición es la siguiente: si es posible contruir un isomorfismo entre dos conjuntos A, B entonces dichos conjuntos son estructuralmente idénticos.

Definición 2.14. Espacio vectorial completo con producto interno. Sea \mathbb{V} un espacio vectorial con producto interno. \mathbb{V} es *completo* si para cualquier sucesión $\{|a_i\rangle\}_{i=1}^{\infty} \in \mathbb{V}$ que cumpla con $\lim_{i,j \rightarrow \infty} \|| |a_i\rangle - |a_j\rangle \|| = 0$ también se cumple que $\exists! |b\rangle \in \mathbb{V}$ tal que $\lim_{j \rightarrow \infty} \|| |a_j\rangle - |b\rangle \|| = 0$.

Definición 2.15. Espacio de Hilbert (versión computación cuántica).

- Un espacio de Hilbert \mathcal{H} es cualquier espacio vectorial completo con producto interno.

- Dos espacios de Hilbert $\mathcal{H}_1, \mathcal{H}_2$ son isomórficos si los espacios vectoriales asociados son isomórficos a su vez y dicho isomorfismo preserva el producto interno.

- En particular, cuando solamente se trata con espacios vectoriales complejos de dimensión finita, un espacio de Hilbert se define como un espacio vectorial con producto interno (el requerimiento de completitud es eliminado). Este es el tipo de espacios de Hilbert con los que generalmente se trabaja en computación cuántica.

Observación. De acuerdo a lo establecido en la Def. (2.15), el espacio de Hilbert con el que trabajaremos en computación cuántica es $\mathbb{C}^n(\mathbb{C})$, donde n es generalmente un múltiplo de 2.

Definición 2.16. Funcional. Sea $\mathbb{V}(\mathbf{F})$ un espacio vectorial. Un *funcional lineal* (también llamado simplemente *funcional*) es una función lineal $f : \mathbb{V} \rightarrow \mathbf{F}$.

Lema 1. Sea \mathcal{H} un espacio de Hilbert. Definimos, para cada $|a\rangle \in \mathcal{H}$, la función $f_{|a\rangle} : \mathcal{H} \rightarrow \mathbb{C}$ con la operación $f_{|a\rangle}(|b\rangle) = (|a\rangle, |b\rangle)$. Entonces, la función $f_{|a\rangle}$ es una función lineal y, en consecuencia, es un funcional.

Demostración. La definición del producto interno hace de $f_{|a\rangle}$ una función. Sean ahora $|b\rangle, |c\rangle \in \mathcal{H}$ y $\alpha \in \mathbb{C} \Rightarrow$
 $f_{|a\rangle}(|b\rangle + |c\rangle) = (|a\rangle, |b\rangle + |c\rangle) = (|a\rangle, |b\rangle) + (|a\rangle, |c\rangle) = f_{|a\rangle}(|b\rangle) + f_{|a\rangle}(|c\rangle)$. Además,
 $f_{|a\rangle}(\alpha|b\rangle) = (|a\rangle, \alpha|b\rangle) = \alpha(|a\rangle, |b\rangle) = \alpha f_{|a\rangle}(|b\rangle)$, ambas deducciones de acuerdo a Def. (2.3).

Q.E.D.

Es posible demostrar que el conjunto de todos los funcionales de un espacio de Hilbert \mathcal{H} forma, a su vez, otro espacio de Hilbert, llamado *espacio dual de Hilbert* \mathcal{H}^* sobre \mathbb{C} [8]. Más aún, es también posible demostrar que existe una biyección entre \mathcal{H} y \mathcal{H}^* ([8]) y, en consecuencia, \mathcal{H} y \mathcal{H}^* son isomórficos. Este isomorfismo es la base de la muy famosa **notación de Dirac**.

Definición 2.17. Notación de Dirac. Sea \mathcal{H} un espacio de Hilbert \Rightarrow

- Un vector $\vec{\psi} \in \mathcal{H}$ se denota con el símbolo $|\psi\rangle$ y recibe el nombre de **ket**.
- El funcional correspondiente $f_{|\psi\rangle}$ recibe el nombre de **bra** y se denota con el símbolo $\langle\psi|$.
- $\langle\cdot|$ puede ser visto como una función (operador) que mapea un estado arbitrario $|\psi\rangle$ en el funcional $\langle\psi|$ tal que $f_{|\psi\rangle}(|\phi\rangle) = (|\psi\rangle, |\phi\rangle) = \langle\psi|\phi\rangle$.
- Luego, la notación $\langle\cdot|$ será utilizada, en el resto de este documento, para referirse al producto interno de dos vectores en un espacio de Hilbert. Más aún, dicha notación es un *standard* en la formulación y escritura de la mecánica cuántica.
- Por último, definimos $|\psi\rangle^\dagger \equiv \langle\psi|$. El símbolo \dagger es ampliamente utilizado en computación cuántica y su definición formal será estudiada en las siguientes páginas.

Definición 2.18. Representación de kets y bras en vectores columna y renglón. Sea \mathcal{H} un espacio de Hilbert, $\dim\mathcal{H} = n$ y B una base de $\mathcal{H} \Rightarrow$

- Cualquier $|\psi\rangle \in \mathcal{H}$ se puede representar, usando la base B , como un vector *columna* con n componentes, i.e. $|\psi\rangle = (\psi_1, \psi_2, \dots, \psi_n)^t$, donde $\psi_i \in \mathbb{C}$, $i \in \{1, 2, \dots, n\}$.
- Más aún, el bra $\langle\psi| \in \mathcal{H}^*$ se puede representar como un vector *renglón* también de n componentes, i.e. $\langle\psi| = (\psi_1^*, \psi_2^*, \dots, \psi_n^*)$, donde los ψ^* son los conjugados de ψ_i , $i \in \{1, 2, \dots, n\}$
- En consecuencia, si deseamos calcular el producto interno $\langle\psi|\phi\rangle$, donde $|\psi\rangle, |\phi\rangle \in \mathcal{H}$, y para ello queremos usar las representaciones $\langle\psi| = (\psi_1^*, \psi_2^*, \dots, \psi_n^*)$ y $|\phi\rangle = (\phi_1, \phi_2, \dots, \phi_n)^t$ obtenidas mediante el uso de una base B de $\mathcal{H} \Rightarrow$

$$\langle\psi|\phi\rangle = (\psi_1^*\phi_1 + \psi_2^*\phi_2 + \dots + \psi_n^*\phi_n)^{1/2} = \left(\sum_{i=1}^n \psi_i^*\phi_i\right)^{1/2}$$

2.4 Operadores lineales

Uno de los objetivos del Análisis Matemático es estudiar ampliamente funciones cuyos dominios y contradominios son subconjuntos de espacios vectoriales. Estas funciones reciben el nombre de transformaciones u operadores. En particular, los resultados de la teoría de operadores lineales son de mucho interés en la física cuántica y, por lo tanto, son un tema de estudio en la computación cuántica.

Definición 2.19. Operador lineal. Sean \mathbb{V} y \mathbb{W} espacios vectoriales. Un **operador lineal**

$$\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$$

es una función que asigna a cada vector $|\psi\rangle \in \mathbb{V}$ un único vector $\hat{T}|\psi\rangle \in \mathbb{W}$ tal que $\forall |\psi\rangle, |\phi\rangle \in \mathbb{V}$ y $\forall \alpha \in F$ (el campo usado en la definición de \mathbb{V} y \mathbb{W}), se cumple que:

$$\begin{aligned}\hat{T}(|\psi\rangle + |\phi\rangle) &= \hat{T}|\psi\rangle + \hat{T}|\phi\rangle \\ \hat{T}(\alpha|\psi\rangle) &= \alpha\hat{T}|\psi\rangle\end{aligned}$$

Para cada $|\psi\rangle \in \mathbb{V}$, $\hat{T}|\psi\rangle$ se llama *imagen* de $|\psi\rangle$. La imagen del dominio \mathbb{V} , $\hat{T}(\mathbb{V})$, es el *recorrido* de \hat{T} .

Ejercicio 2.10. Ejemplos de operadores lineales.

1) Sea $\hat{T} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ definida por $\hat{T}(x, y) = (x + y, x - y, 3y)$. Demuestre que \hat{T} es un operador lineal.

2) Sea $A \in \mathbb{M}_{m \times n}(\mathbb{R})$. Definamos a $\hat{T} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ por $\hat{T}|\psi\rangle = A|\psi\rangle$. Demuestre que \hat{T} es un operador lineal.

Observación 1. Si $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ es un operador lineal y \mathbb{V} es un espacio vectorial complejo de dimensión $n \Rightarrow \hat{T}$ tiene un conjunto infinito de representaciones matriciales, i.e. la acción de \hat{T} sobre los elementos $|\psi\rangle \in \mathbb{V}$ se puede llevar a cabo de la siguiente manera: escoja una representación de $|\psi\rangle$ en una base dada, para después crear una matriz de orden n que permita calcular $T|\psi\rangle$ en la base vectorial correspondiente.

Este resultado no se extiende a espacios vectoriales infinito-dimensionales (por ejemplo, en estos casos se requerirán representaciones de operadores integrales o diferenciales).

Observación 2. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal. Deseamos subrayar que la acción de \hat{T} sobre \mathbb{V} y \mathbb{W} es *independiente* de las bases que uno escoja para representar a los elementos de \mathbb{V} y \mathbb{W} . Un operador lineal puede ser pensado como un procedimiento para transformar una entidad geométrica en otra.

Teorema 7. El conjunto $\hat{T}(\mathbb{V})$, el recorrido del \hat{T} , es un subespacio de \mathbb{W} .

Demostración

Sean $|\psi\rangle, |\phi\rangle \in \mathbb{V}$ tales que $\hat{T}|\psi\rangle, \hat{T}|\phi\rangle \in \hat{T}(\mathbb{V}) \subset \mathbb{W}$. Como \mathbb{W} es un espacio vectorial $\Rightarrow \hat{T}|\psi\rangle + \hat{T}|\phi\rangle \in \mathbb{W}$. Dado que \hat{T} es un operador lineal $\Rightarrow \hat{T}|\psi\rangle + \hat{T}|\phi\rangle = \hat{T}(|\psi\rangle + |\phi\rangle)$, i.e. $\hat{T}(|\psi\rangle + |\phi\rangle) = \hat{T}|\psi\rangle + \hat{T}|\phi\rangle$ es la imagen de $|\psi\rangle + |\phi\rangle$. Por lo tanto, $\hat{T}|\psi\rangle + \hat{T}|\phi\rangle \in \hat{T}(\mathbb{V})$.

Ahora, sean $|\psi\rangle \in \mathbb{V}, \alpha$ un escalar y $\hat{T}|\psi\rangle \in \hat{T}(\mathbb{V}) \subset \mathbb{W}$. Como \mathbb{W} es un espacio vectorial entonces $\alpha\hat{T}|\psi\rangle \in \mathbb{W}$ y, dado que \hat{T} es un operador lineal, $\alpha\hat{T}|\psi\rangle = \hat{T}(\alpha|\psi\rangle)$, esto es, existe un elemento $\hat{T}(\alpha|\psi\rangle) \in \mathbb{W}$ que es, por definición, la imagen de $\alpha|\psi\rangle$, i.e. $\hat{T}(\alpha|\psi\rangle) \in \hat{T}(\mathbb{V})$.

Luego, de acuerdo al teorema (1), $\hat{T}(\mathbb{V})$ es un subespacio de \mathbb{W} .

Q.E.D.

Teorema 8. Sean \mathbb{V} y \mathbb{W} espacios vectoriales definidos sobre \mathbb{R} y $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal. Entonces, $\forall |\psi\rangle, |\phi\rangle, |\psi_i\rangle \in \mathbb{V}$ y $\forall \alpha_i \in \mathbb{R}$ se encuentra que:

$$1) \hat{T}(\mathbf{0}_{\mathbb{V}}) = \mathbf{0}_{\mathbb{W}}$$

$$2) \hat{T}(|\psi\rangle - |\phi\rangle) = \hat{T}|\psi\rangle - \hat{T}|\phi\rangle$$

$$3) \hat{T}\left(\sum_{i=1}^n \alpha_i |\psi_i\rangle\right) = \sum_{i=1}^n \alpha_i \hat{T}|\psi_i\rangle$$

Demostración

1) Sea $\hat{T}(\mathbf{0}_{\mathbb{V}}) = \hat{T}(\mathbf{0}_{\mathbb{V}} + \mathbf{0}_{\mathbb{V}}) = \hat{T}(\mathbf{0}_{\mathbb{V}}) + \hat{T}(\mathbf{0}_{\mathbb{V}})$, de acuerdo a la Def. (2.19). Tenemos entonces que $\hat{T}(\mathbf{0}_{\mathbb{V}}) = \hat{T}(\mathbf{0}_{\mathbb{V}}) + \hat{T}(\mathbf{0}_{\mathbb{V}})$, siendo $\hat{T}(\mathbf{0}_{\mathbb{V}})$ un elemento de \mathbb{W} .

Esta suma nos obliga a concluir que $\hat{T}(\mathbf{0}_{\mathbb{V}}) = \mathbf{0}_{\mathbb{W}}$, pues $\mathbf{0}_{\mathbb{W}}$ es el único elemento de \mathbb{W} que, al ser sumado al elemento $\hat{T}(\mathbf{0}_{\mathbb{V}})$, puede producir como resultado el mismo elemento $\hat{T}(\mathbf{0}_{\mathbb{V}})$.

2) Sean $|\psi\rangle, |\phi\rangle \in \mathbb{V}$. Sabemos que $(-1) \times |\phi\rangle = -|\phi\rangle \in \mathbb{V}$, así que $\hat{T}(|\psi\rangle - |\phi\rangle) = \hat{T}|\psi\rangle + \hat{T}(-|\phi\rangle)$, de acuerdo a la Def. (2.19).

Más aún, usando otra vez la Def. (2.19), encontramos que $\hat{T}(-|\phi\rangle) = \hat{T}((-1) \times |\phi\rangle) = (-1)\hat{T}|\phi\rangle = -\hat{T}|\phi\rangle$. Luego, $\hat{T}(|\psi\rangle - |\phi\rangle) = \hat{T}|\psi\rangle - \hat{T}|\phi\rangle$.

3) La demostración se hace por inducción sobre el número de vectores y se deja al lector como ejercicio (consejo: use la Def. (2.19) para formular el caso base).

Q.E.D.

Definición 2.20. Núcleo de un operador lineal. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal. Definimos al *núcleo* de \hat{T} , designado por $N(\hat{T})$, de la siguiente manera:

$$N(\hat{T}) = \{|\psi\rangle \in \mathbb{V} \mid \hat{T}|\psi\rangle = \mathbf{0}\}$$

Teorema 9. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal \Rightarrow el núcleo de \hat{T} es un subespacio de \mathbb{V} .

Demostración

1) Sean $|\psi\rangle, |\phi\rangle \in N(\hat{T}) \Rightarrow \hat{T}|\psi\rangle = \mathbf{0}$ y $\hat{T}|\phi\rangle = \mathbf{0}$. Por una parte, tenemos que $\hat{T}|\psi\rangle + \hat{T}|\phi\rangle = \mathbf{0} + \mathbf{0} = \mathbf{0}$. Además, dado que \hat{T} es un operador lineal, encontramos que $\hat{T}|\psi\rangle + \hat{T}|\phi\rangle = \hat{T}(|\psi\rangle + |\phi\rangle)$, lo que nos lleva a concluir que $\hat{T}(|\psi\rangle + |\phi\rangle) = \mathbf{0}$, i.e. $(|\psi\rangle + |\phi\rangle) \in N(\hat{T})$.

2) Sean $|\psi\rangle \in N(\hat{T})$ y α un escala $\Rightarrow \hat{T}|\psi\rangle = \mathbf{0} \Rightarrow \alpha\hat{T} = \alpha\mathbf{0} = \mathbf{0}$. Dado que \hat{T} es lineal entonces $\alpha\hat{T}|\psi\rangle = \hat{T}(\alpha|\psi\rangle)$. En consecuencia, $\hat{T}(\alpha|\psi\rangle) = \mathbf{0}$, i.e. $\alpha|\psi\rangle \in N(\hat{T})$.

Luego, de acuerdo al teorema (1), $N(\hat{T})$ es un subespacio de \mathbb{V} .

Q.E.D.

2.4.1 Acción de un operador lineal sobre una base

Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal, con \mathbb{V} y \mathbb{W} espacios vectoriales y $B = \{|e_1\rangle, |e_2\rangle, \dots, |e_n\rangle\}$ una base de \mathbb{V} . Por definición, para cada $|\psi\rangle \in \mathbb{V} \exists! |w\rangle \in \mathbb{W}$ tal que $\hat{T}|\psi\rangle = |w\rangle$. En particular, vemos que para cada $|e_i\rangle \in B$ existe un único $|u_i\rangle \in \mathbb{W}$ tal que $\hat{T}|e_i\rangle = |u_i\rangle$.

Ahora bien, sea $|x\rangle$ un elemento arbitrario de $\mathbb{V} \Rightarrow$ podemos expresar a $|x\rangle$ como la sumatoria $|x\rangle = \sum_{i=1}^n \alpha_i |e_i\rangle$ (recuerde que el conjunto de escalares $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ es único).

Luego, si conocemos los elementos $|u_i\rangle \in \mathbb{W}$ tales que $\hat{T}|e_i\rangle = |u_i\rangle$ y usamos el Teorema (8), encontramos que $\hat{T}|x\rangle = \hat{T}(\sum_{i=1}^n \alpha_i |e_i\rangle) = \sum_{i=1}^n \alpha_i \hat{T}|e_i\rangle = \sum_{i=1}^n \alpha_i |u_i\rangle$.

En otras palabras, si suponemos la existencia de un operador lineal $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$, una base $B = \{|e_1\rangle, \dots, |e_n\rangle\}$ de \mathbb{V} y el conjunto $C = \{\hat{T}|e_1\rangle, \dots, \hat{T}|e_n\rangle\}$, i.e. la acción de \hat{T} sobre $B \Rightarrow$ **siempre se puede calcular** la acción de \hat{T} sobre un vector arbitrario $|v\rangle \in \mathbb{V}$ para el cual conocemos sus componentes respecto de la base B , i.e. $|v\rangle = \sum_{i=1}^n \alpha_i |e_i\rangle$. Este razonamiento es un método eficaz para calcular el efecto de un operador lineal sobre cualquier elemento de su dominio.

Veremos ahora un teorema que vincula la acción de un operador lineal inyectivo sobre las bases de su dominio y contradominio. La demostración de este teorema es directa y se deja como ejercicio al lector (en caso de necesitar guía, se recomienda consultar [9]).

Teorema 10. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal con $\dim \mathbb{V} = n$ entonces las siguientes proposiciones son equivalentes:

- 1) \hat{T} es invertible y su inversa $\hat{T}^{-1} : \hat{T}(\mathbb{V}) \rightarrow \mathbb{V}$ es también un operador lineal.
- 2) si $|e_1\rangle, |e_2\rangle, \dots, |e_n\rangle \in \mathbb{V}$ son vectores linealmente independientes $\Rightarrow \hat{T}|e_1\rangle, \hat{T}|e_2\rangle, \dots, \hat{T}|e_n\rangle \in \hat{T}(\mathbb{V})$ son también vectores linealmente independientes.
- 3) $\dim \hat{T}(\mathbb{V}) = n$
- 4) Si $|e_1\rangle, |e_2\rangle, \dots, |e_n\rangle$ es una base de $\mathbb{V} \Rightarrow \hat{T}|e_1\rangle, \hat{T}|e_2\rangle, \dots, \hat{T}|e_n\rangle$ es una base de $\hat{T}(\mathbb{V})$.

2.4.2 Representaciones matriciales de operadores lineales

Mencionamos en el comienzo de este capítulo que un operador lineal cuyo dominio es un espacio vectorial complejo n -dimensional tiene un conjunto infinito de representaciones matriciales. En esta sección formalizaremos dicha noción.

Sea el operador lineal $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ con $\dim \mathbb{V} = n$ y $\dim \mathbb{W} = m$. De acuerdo a lo estudiado en la sección anterior, el efecto de un operador lineal sobre su dominio está determinado por su acción sobre una base $\{|e_i\rangle\}$ de \mathbb{V} . Puesto que los vectores $\hat{T}|e_i\rangle \in \mathbb{W}$ entonces podemos tomar una base $\{|w_i\rangle\}$ de \mathbb{W} y construir n ecuaciones de la forma

$$\hat{T}|e_1\rangle = \sum_{k=1}^m \alpha_{k1} |w_k\rangle; \hat{T}|e_2\rangle = \sum_{k=1}^m \alpha_{k2} |w_k\rangle; \dots; \hat{T}|e_n\rangle = \sum_{k=1}^m \alpha_{kn} |w_k\rangle$$

O, en su forma más general,

$$\hat{T}|e_i\rangle = \sum_{k=1}^m \alpha_{ki}|w_k\rangle$$

Escribamos ahora una matriz cuyas *columnas* sean los coeficientes de los vectores $\hat{T}|e_i\rangle$ en términos de la base $\{|w_i\rangle\}$, i.e.

$$\left(\hat{T}|e_1\rangle_{\{|w_i\rangle\}} \quad \hat{T}|e_2\rangle_{\{|w_i\rangle\}} \quad \dots \quad \hat{T}|e_n\rangle_{\{|w_i\rangle\}} \right) = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ & & \vdots & \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mn} \end{pmatrix}$$

Los coeficientes α_{ij} *dependen* de la base $\{|w_i\rangle\}$ que escojamos para \mathbb{W} . Puesto que para un espacio vectorial de dimensión m existe un conjunto infinito de distintas bases, entonces **hay un conjunto infinito de representaciones matriciales distintas** para un mismo operador lineal \hat{T} .

Formalicemos ahora cómo una representación matricial de un operador lineal \hat{T} actúa sobre los elementos del dominio de dicho operador.

Teorema 11. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ un operador lineal con $\dim\mathbb{V} = n$ y $\dim\mathbb{W} = m$. Además, suponga la existencia de $\{|e_i\rangle\}$ y $\{|w_i\rangle\}$, bases de \mathbb{V} y \mathbb{W} respectivamente, a partir de las cuales se calcula la matriz $T = (\alpha_{ij})$ cuyos elementos están determinados por las ecuaciones

$$\hat{T}|e_i\rangle = \sum_{k=1}^m \alpha_{ki}|w_k\rangle \quad (3)$$

Tomemos ahora un elemento cualquiera

$$|\psi\rangle = \sum_{p=1}^n \beta_p|e_p\rangle \quad (4)$$

y apliquemos el operador

$$\hat{T}|\psi\rangle = \sum_{p=1}^n \beta_p \hat{T}|e_p\rangle = \sum_{q=1}^m \gamma_q|w_q\rangle \quad (5)$$

Entonces los coeficientes γ_i tienen la forma $\gamma_i = \sum_{k=1}^n \beta_k \alpha_{ik}$.

Demostración.

Combinando las Ecs. (5 y 3) encontramos que

$$\begin{aligned} \hat{T}|\psi\rangle &= \sum_{p=1}^n \beta_p \hat{T}|e_p\rangle = \beta_1 \hat{T}|e_1\rangle + \beta_2 \hat{T}|e_2\rangle + \dots + \beta_n \hat{T}|e_n\rangle = \\ &= \beta_1 \sum_{k=1}^m \alpha_{k1}|w_k\rangle + \beta_2 \sum_{k=1}^m \alpha_{k2}|w_k\rangle + \dots + \beta_n \sum_{k=1}^m \alpha_{kn}|w_k\rangle = \\ &= \beta_1(\alpha_{11}|w_1\rangle + \alpha_{21}|w_2\rangle + \dots + \alpha_{m1}|w_m\rangle) + \\ &= \beta_2(\alpha_{12}|w_1\rangle + \alpha_{22}|w_2\rangle + \dots + \alpha_{m2}|w_m\rangle) + \dots + \\ &= \beta_n(\alpha_{1n}|w_1\rangle + \alpha_{2n}|w_2\rangle + \dots + \alpha_{mn}|w_m\rangle). \end{aligned}$$

Agrupando los sumandos en términos de los coeficientes de $|w_i\rangle$, observamos lo siguiente

$$\begin{aligned}\hat{T}|\psi\rangle &= \sum_{p=1}^n \beta_p \hat{T}|e_p\rangle \\ &= (\beta_1 \alpha_{11} + \beta_2 \alpha_{12} + \dots + \beta_n \alpha_{1n})|w_1\rangle + \\ &= (\beta_1 \alpha_{21} + \beta_2 \alpha_{22} + \dots + \beta_n \alpha_{2n})|w_2\rangle + \dots + \\ &= (\beta_1 \alpha_{m1} + \beta_2 \alpha_{m2} + \dots + \beta_n \alpha_{mn})|w_m\rangle = \\ &= \sum_{k=1}^n \beta_k \alpha_{1k}|w_1\rangle + \sum_{k=1}^n \beta_k \alpha_{2k}|w_2\rangle + \dots + \sum_{k=1}^n \beta_k \alpha_{mk}|w_m\rangle\end{aligned}$$

Esto es,

$$\begin{aligned}\hat{T}|\psi\rangle &= \sum_{p=1}^n \beta_p \hat{T}|e_p\rangle = \sum_{q=1}^m \gamma_q |w_q\rangle = \\ &= \sum_{k=1}^n \beta_k \alpha_{1k}|w_1\rangle + \sum_{k=1}^n \beta_k \alpha_{2k}|w_2\rangle + \dots + \sum_{k=1}^n \beta_k \alpha_{mk}|w_m\rangle.\end{aligned}$$

Del desarrollo anterior se observa que γ_i , el coeficiente de $|w_i\rangle$, es igual a $\sum_{k=1}^n \beta_k \alpha_{ik}$, con lo que queda demostrado el teorema.

Q.E.D.

Dado que hay muchas representaciones matriciales para un operador lineal, es de interés del investigador encontrar y caracterizar representaciones simples, esto es, matrices cuya manipulación sea fácil de realizar.

Entre las representaciones matriciales que se usan comúnmente en computación cuántica, encontramos las *matrices diagonales*. El siguiente teorema demuestra la existencia de este tipo de matrices, y su demostración puede encontrarse en [9].

Teorema 12. Sean \mathbb{V} y \mathbb{W} espacios vectoriales de dimensión finita, con $\dim\mathbb{V} = n$ y $\dim\mathbb{W} = m$. Supongamos que $\hat{T} : \mathbb{V} \rightarrow \mathbb{W}$ y que $r = \dim\hat{T}(\mathbb{V})$ representa el rango de \hat{T} . Existe entonces una base $\{|e_1\rangle, |e_2\rangle, \dots, |e_n\rangle\}$ de \mathbb{V} y otra base $\{|w_1\rangle, |w_2\rangle, \dots, |w_m\rangle\}$ de \mathbb{W} tales que

$$\begin{aligned}\hat{T}(|e_i\rangle) &= |w_i\rangle, i \in \{1, \dots, r\} \\ \hat{T}(|e_i\rangle) &= \mathbf{0}, i \in \{r+1, \dots, n\}\end{aligned}$$

Además, la matriz (α_{ij}) relativa a estas bases tiene todos sus elementos iguales a cero excepto los r elementos de la diagonal, cuyo valor es $t_{11} = t_{22} = \dots = t_{rr} = 1$.

2.4.3 Operadores lineales usando notación de Dirac

Definición 2.21. Producto externo: representación de un operador lineal en notación de Dirac.

Sean $|\psi\rangle, |a\rangle \in \mathcal{H}_1$ y $|\phi\rangle \in \mathcal{H}_2$. Definimos entonces al **producto externo** (también conocido como *producto exterior*) $|\phi\rangle\langle\psi|$ como el operador lineal con dominio \mathcal{H}_1 y contradominio \mathcal{H}_2 , el cual opera de la siguiente manera:

$$(|\phi\rangle\langle\psi|)(|a\rangle) \equiv |\phi\rangle\langle\psi|a\rangle \equiv \langle\psi|a\rangle|\phi\rangle$$

donde $\langle\psi|a\rangle$ es el producto interno de $|\psi\rangle$ y $|a\rangle$.

Ejemplos de operadores en notación de Dirac y en representación matricial.

Comenzamos por definir los operadores de Pauli usando notación de Dirac:

$$\hat{\sigma}_x = |0\rangle\langle 1| + |1\rangle\langle 0| \quad ; \quad \hat{\sigma}_y = i|0\rangle\langle 1| - i|1\rangle\langle 0| \quad ; \quad \hat{\sigma}_z = |0\rangle\langle 0| - |1\rangle\langle 1| \quad (6)$$

Si además definimos $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $\langle 0| = (1, 0)$ y $\langle 1| = (0, 1)$.

entonces podemos calcular representaciones matriciales de los operadores de Pauli:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad ; \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad ; \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (7)$$

El **operador de Hadamard** es otro operador lineal ampliamente usado en computación cuántica:

$$\hat{H} = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) \quad (8)$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (9)$$

Los operadores de Pauli y Hadamard son ejemplos de dos tipos de operadores utilizados frecuentemente en mecánica cuántica: operadores hermitianos y unitarios.

2.5 Operadores usados en computación cuántica

Suponga que $\hat{A} : \mathbb{V} \rightarrow \mathbb{V}$ es un operador lineal. Por una parte, sabemos que $\forall |\psi\rangle \in \mathbb{V} \Rightarrow \hat{A}|\psi\rangle \in \mathbb{V}$ y que, para cada ket $|\psi\rangle$ en \mathbb{V} , existe un único bra $\langle\psi|$ en el espacio dual \mathbb{V}^* . Por extensión, si $\hat{A}|\psi\rangle = |\psi'\rangle$ es un elemento de \mathbb{V} entonces el bra $\langle\psi'|$ es un elemento de \mathbb{V}^* .

Dados estos elementos, procedemos a definir a \hat{A}^\dagger , el operador adjunto de \hat{A} , como el operador capaz de generar al bra $\langle\psi'|$ a partir del bra $\langle\psi|$, esto es,

$$\hat{A}|\psi'\rangle = |\psi\rangle \Leftrightarrow \langle\psi'| = \langle\psi|\hat{A}^\dagger$$

Ahora bien, suponga que \hat{A} es un operador lineal y A una de sus representaciones matriciales. Entonces, la representación matricial de \hat{A}^\dagger , basada en la matriz A , es la matriz $(A^*)^t$, esto es, la matriz que representa a \hat{A}^\dagger es la transpuesta de la matriz conjugada de A .

Damos en el siguiente teorema algunas propiedades fundamentales de la operación \dagger , cuya demostración puede consultarse en varias fuentes, en particular en [57].

Teorema 13. Sean \hat{A}, \hat{B} operadores con dominio \mathbb{V} , $|u\rangle\langle v|$ un operador en notación de Dirac y $\alpha \in \mathbb{C}$. Entonces

- 1) $(\hat{A}^\dagger)^\dagger = \hat{A}$
- 2) $(\alpha\hat{A})^\dagger = \alpha^*\hat{A}^\dagger$
- 3) $(\hat{A} + \hat{B})^\dagger = \hat{A}^\dagger + \hat{B}^\dagger$
- 4) $(\hat{A}\hat{B})^\dagger = \hat{B}^\dagger\hat{A}^\dagger$
- 5) $(|u\rangle\langle v|)^\dagger = |v\rangle\langle u|$

Usando el Teorema (13), podemos demostrar fácilmente el siguiente resultado

Teorema 14. Sean $\hat{A}_i : \mathbb{V} \rightarrow \mathbb{V}, i \in \{1, \dots, n\}$ n operadores lineales y $\alpha_i \in \mathbb{C}$ n escalares \Rightarrow

$$\left(\sum_{i=1}^n \alpha_i \hat{A}_i\right)^\dagger = \sum_{i=1}^n \alpha_i^* \hat{A}_i^\dagger$$

Definimos enseguida dos tipos de operadores muy usados en mecánica cuántica.

2.5.1 Operadores hermitianos

Definición 2.22. Operador hermitiano. Sea \mathcal{H} un espacio de Hilbert de dimensión finita y $\hat{A} : \mathcal{H} \rightarrow \mathcal{H}$ un operador lineal. Si $\hat{A} = \hat{A}^\dagger$ entonces \hat{A} es un **operador hermitiano**. La definición se extiende de forma natural a cualquier representación matricial de \hat{A} .

Ejercicio 2.11. Demuestre que los operadores de Pauli son hermitianos.

Una clase de operadores hermitianos fundamentales en mecánica cuántica son los **proyectores**.

Definición 2.23. Proyectores. Sea \mathbb{V} un espacio vectorial y \mathbb{W} un subespacio de \mathbb{V} , con $\dim \mathbb{V} = n$ y $\dim \mathbb{W} = m$. Usando el procedimiento de Gram-Schmidt (ortonormalización de un conjunto de vectores, consultar [9–11]), es posible construir una base ortonormal $|u_1\rangle, |u_2\rangle, \dots, |u_n\rangle$ para \mathbb{V} tal que $|u_1\rangle, |u_2\rangle, \dots, |u_k\rangle$ sea, a su vez, una base ortonormal para \mathbb{W} . Definimos

$$\hat{P} = \sum_{i=1}^k |u_i\rangle\langle u_i| \tag{10}$$

como el operador de proyección al subespacio \mathbb{W} .

Observe que, de acuerdo a los Teoremas (13) y (14),

$$\hat{P}^\dagger = \left(\sum_{i=1}^k |u_i\rangle\langle u_i|\right)^\dagger = \sum_{i=1}^k (|u_i\rangle\langle u_i|)^\dagger = \sum_{i=1}^k |u_i\rangle\langle u_i| = \hat{P}$$

Luego, \hat{P} es hermitiano de acuerdo a la Def. (2.22).

Ejemplo 2.1. El conjunto $\{|0\rangle, |1\rangle\}$ es una base ortonormal de \mathcal{H}^2 . Luego,

$|0\rangle\langle 0|$ es el operador de proyección al subespacio generado por $|0\rangle$

$|1\rangle\langle 1|$ es el operador de proyección al subespacio generado por $|1\rangle$

Definición 2.24. Operador unitario. Sea \mathcal{H} un espacio de Hilbert y $\hat{U} : \mathcal{H} \rightarrow \mathcal{H}$ un operador lineal. \hat{U} es un **operador unitario** si $\hat{U}\hat{U}^\dagger = \hat{I}$, donde \hat{I} es el operador identidad. Como en el caso de los operadores hermitianos, la definición de matrices unitarias es una extensión natural de la definición de un operador unitario.

Los operadores unitarios son muy importantes en mecánica cuántica porque preservan el valor del producto interno: sean $|\alpha\rangle = \hat{U}|a\rangle$ y $|\beta\rangle = \hat{U}|b\rangle \Rightarrow \langle\alpha|\beta\rangle = \langle a|\hat{U}^\dagger\hat{U}|b\rangle = \langle a|\hat{I}|b\rangle = \langle a|b\rangle$.

Los operadores hermitianos y unitarios son ejemplos de *operadores normales*, los cuales se definen a continuación.

Ejercicio 2.12. Demuestre lo siguiente:

- 1) Los operadores de Pauli (Ec. (6)) y Hadamard (Ec. (8)) son unitarios.
- 2) Las matrices de Pauli (Ec. (7)) y Hadamard (Ec. (9)) son unitarias.

Definición 2.25. Operador normal. Sea \mathcal{H} un espacio de Hilbert y $\hat{A} : \mathcal{H} \rightarrow \mathcal{H}$ un operador lineal. \hat{A} es *normal* si y sólo si $\hat{A}\hat{A}^\dagger = \hat{A}^\dagger\hat{A}$.

Los operadores normales tienen una propiedad matemática, la *descomposición espectral*, que juega un papel fundamental en la formulación de la mecánica cuántica. Para poder revisar el teorema de la descomposición espectral, antes es necesario recordar un tema: eigenvectores y eigenvalores.

2.6 Eigenvectores y eigenvalores

Definición 2.26. Eigenvectores y eigenvalores. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ un operador lineal. Un escalar λ recibe el nombre de *eigenvalor* de \hat{T} si existe un elemento *no nulo* $|\psi\rangle \in \mathbb{V}$ tal que

$$\hat{T}|\psi\rangle = \lambda|\psi\rangle \quad (11)$$

El elemento $|\psi\rangle$ recibe el nombre de *eigenvector* de \hat{T} perteneciente a λ . El escalar λ se llama *eigenvalor* de \hat{T} .

Definición 2.27. Eigenespacio. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ un operador lineal con un eigenvalor λ . Definimos a $E(\lambda)$ el *eigenespacio* de λ , de la siguiente manera:

$$E(\lambda) = \{|\psi\rangle \in \mathbb{V} \mid \hat{T}|\psi\rangle = \lambda|\psi\rangle\}$$

- 1) Observe que $E(\lambda)$ contiene al vector $\mathbf{0}$ y a todos los eigenvectores pertenecientes a λ .
- 2) Es fácil demostrar que $E(\lambda)$ es un subespacio de \mathbb{V} , pues si $|x\rangle, |y\rangle \in \mathbb{V}$, $\alpha, \beta \in \mathbb{C} \Rightarrow$

$$\hat{T}(\alpha|x\rangle + \beta|y\rangle) = \alpha\hat{T}|x\rangle + \beta\hat{T}|y\rangle = \alpha\lambda|x\rangle + \beta\lambda|y\rangle = \lambda(\alpha|x\rangle + \beta|y\rangle), \text{ con } \alpha, \beta \in \mathbb{C}.$$

2.6.1 Polinomio característico

Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ un operador lineal. Tomemos la ecuación que define eigenvalores y eigenvectores:

$$\hat{T}|\psi\rangle = \lambda|\psi\rangle \Rightarrow \lambda|\psi\rangle = \hat{T}|\psi\rangle \Rightarrow \lambda\hat{I}|\psi\rangle = \hat{T}|\psi\rangle \Rightarrow \lambda\hat{I}|\psi\rangle - \hat{T}|\psi\rangle = \mathbf{0} \quad (12)$$

Por definición, $(f_1 + f_2)(x) = f_1(x) + f_2(x)$, por ello es que

$$\lambda\hat{I}|\psi\rangle - \hat{T}|\psi\rangle = \mathbf{0} \Rightarrow (\lambda\hat{I} - \hat{T})|\psi\rangle = \mathbf{0} \quad (13)$$

Puesto que $(\lambda\hat{I} - \hat{T})$ es un operador, podemos escribir la Ec. (13) de esta manera:

$$\hat{A}|\psi\rangle = \lambda|\psi\rangle \quad (14)$$

Sea ahora A una representación matricial del operador \hat{A} . Así las cosas, la Ec. (14) se puede escribir de la siguiente manera:

$$A|\psi\rangle = \lambda|\psi\rangle, \text{ i.e.}$$

$$(\lambda I - T)|\psi\rangle = \mathbf{0} \quad (15)$$

donde T es una representación matricial de \hat{T} e I es la matriz identidad.

Sabemos de estudios elementales de álgebra lineal que la ecuación matricial $A|x\rangle = \mathbf{0}$ sólo tiene la solución trivial $\Leftrightarrow \det(A) = 0$. Por otra parte, dado que la Def. (2.26) excluye al vector nulo, i.e. $|\psi\rangle \neq \mathbf{0}$ entonces sólo nos queda concluir que, para que la Ec. (15) se cumpla, forzosamente se debe cumplir la siguiente ecuación:

$$\det(\lambda I - T) = 0 \quad (16)$$

Definición 2.28. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ un operador lineal y T una representación lineal de \hat{T} . Definimos entonces la **ecuación característica**

$$c(\lambda) = \det(\lambda I - T) = 0$$

Observación. Puesto que $\det(kA) = k^n \det(A)$, donde $A \in \mathbb{M}_n(\mathbb{C}) \Rightarrow c(\lambda) = \det(\lambda I - T) = \det(T - \lambda I) = 0$.

Definición 2.29. Polinomio característico. Al desarrollar $c(\lambda)$ se obtiene un polinomio en λ , denominado **polinomio característico** de A .

Es posible demostrar que, si la matriz T es de orden $n \Rightarrow$ su polinomio característico es de grado n . Por otra parte, el teorema fundamental del álgebra establece que todo polinomio de grado n tiene exactamente n raíces, contando duplicidades. Por lo tanto,

Teorema 15. Sea $A \in \mathbb{M}_n(\mathbb{C})$. Si se consideran mutiplicidades A tiene exactamente n eigenvalores.

Un teorema más, cuya demostración es directa y se deja como ejercicio para el lector,

Teorema 16. Sea $T \in \mathbb{M}_n(\mathbb{C})$ y λ un eigenvalor de A . El conjunto $E(\lambda)$, el eigenespacio de λ , es un subespacio de \mathbb{C}^n .

En resumen, un camino para conocer los eigenvalores de un operador lineal \hat{T} es calcular una representación matricial T de dicho operador. Acto seguido, sobre esta matriz T se calcula la función característica $c(\lambda)$ y, a partir de esto último, el polinomio característico correspondiente. Las raíces son los eigenvalores de la matriz T y, por lo tanto, los eigenvalores del operador \hat{T} .

En el razonamiento anterior pareciera que existe un paso falso. Si un operador \hat{T} tiene muchas representaciones matriciales, ¿cómo saber que los eigenvalores de una representación matricial determinada son también los del operador \hat{T} ?

La respuesta a esta interrogante la dan los siguientes teoremas, cuyas demostraciones se recomienda consultar en [9]

Definición 2.30. Dos matrices $A, B \in \mathbb{M}_n(\mathbb{C})$ se llaman **semejantes** si existe una matriz no singular $C \in \mathbb{M}_n(\mathbb{C})$, i.e. C^{-1} existe, tal que

$$B = C^{-1}AC$$

Teorema 17. Si dos matrices $A, B \in \mathbb{M}_n(\mathbb{C})$ representan el mismo operador lineal $\hat{T} \Rightarrow \exists$ una matriz no singular $C \in \mathbb{M}_n(\mathbb{C})$ tal que $B = C^{-1}AC$. En otras palabras, si A, B representan al mismo operador lineal si dichas matrices son semejantes.

El recíproco del Teorema (17) también es cierto.

Teorema 18. Sean $A, B, C \in \mathbb{M}_n(\mathbb{C})$ dos matrices relacionadas por una ecuación de la forma $B = C^{-1}AC \Rightarrow A$ y B representan el mismo operador lineal.

Los teoremas (17,18) se resumen de la siguiente manera:

Teorema 19. Dos matrices son semejantes si y sólo si representan al mismo operador lineal.

El teorema (19), unido a la siguiente propiedad de la función determinante: $\det(C^{-1}AC) = \det(C^{-1})\det(A)\det(C)$, son el fundamento del siguiente teorema, el cual establece que, si cualesquiera dos representaciones matriciales de un operador lineal tienen los mismos autovalores.

Teorema 20. Las matrices semejantes tienen el mismo polinomio característico y por tanto los mismos eigenvalores.

Para terminar, enunciamos un teorema que relaciona la diagonalización de una matriz con sus eigenvalores. Este resultado es fundamental en la estructura matemática de la mecánica cuántica.

Teorema 21. Sea $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ un operador lineal, \mathbf{F} el campo de definición del espacio vectorial \mathbb{V} y $\dim \mathbb{V} = n$. Además, supongamos que el polinomio característico de \hat{T} tiene n raíces *distintas* $\lambda_1, \lambda_2, \dots, \lambda_n \Rightarrow$

- Existe un conjunto de eigenvalores $U = \{|u_1\rangle, |u_2\rangle, \dots, |u_n\rangle\}$, que corresponden a $\lambda_1, \lambda_2, \dots, \lambda_n$ y que conforman una base para \mathbb{V} .
- la matriz T relativa a la base U es la matriz diagonal Λ que tiene a los eigenvalores $\lambda_1, \lambda_2, \dots, \lambda_n$ como elementos de la diagonal, i.e. $T = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$.
- Si A es la matriz de \hat{T} relativa a otra base $\{|e_1\rangle, |e_2\rangle, \dots, |e_n\rangle\}$ entonces la matriz diagonal Λ se calcula de acuerdo a la fórmula

$$\Lambda = C^{-1}AC,$$

donde C es la matriz no singular que relaciona las dos bases $\{|u_i\rangle\}, \{|e_i\rangle\}$ mediante la ecuación $U = EC$.

2.7 Descomposición espectral

Teorema 22. Descomposición espectral. Todo operador normal $\hat{T} : \mathbb{V} \rightarrow \mathbb{V}$ es diagonal respecto de una base ortonormal de \mathbb{V} .

Luego, una representación diagonal de \hat{T} tiene la forma $\hat{T} = \sum_i \lambda_i |i\rangle\langle i|$, donde $\{|i\rangle\}$ es un conjunto ortonormal de eigenvectores de \hat{A} que corresponden a los eigenvalores λ_i .

2.8 Producto tensorial

Los conceptos matemáticos revisados hasta el momento serán utilizados para el análisis de sistemas cuánticos *individuales*. En este apartado presentamos al lector un formalismo matemático usado para representar sistemas cuánticos multipartitas.

Definición 2.31. Producto tensorial. Sean $\mathbb{V}(\mathbf{F})$ y $\mathbb{W}(\mathbf{F})$ espacios vectoriales de dimensión m y n respectivamente. Definimos a \mathbb{X} como el producto tensorial de \mathbb{V} y \mathbb{W} , i.e. $\mathbb{X} = \mathbb{V} \otimes \mathbb{W}$.

- Los elementos de \mathbb{X} son *combinaciones lineales* de vectores $|a\rangle \otimes |b\rangle$, donde $|a\rangle \in \mathbb{V}$ and $|b\rangle \in \mathbb{W}$.
- En particular, si $\{|i\rangle\}$ y $\{|j\rangle\}$ son bases ortonormales de \mathbb{V} y \mathbb{W} entonces $\{|i\rangle \otimes |j\rangle\}$ es una base² para \mathbb{X} .

²Un ejemplo concreto: sea $\{|0\rangle, |1\rangle\}$ una base ortonormal de un espacio de Hilbert bidimensional \mathcal{H}^2 . Entonces, una base para $\mathcal{H}^4 = \mathcal{H}^2 \otimes \mathcal{H}^2$ es $\{|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle\}$.

Sean ahora \hat{A}, \hat{B} operadores lineales en \mathbb{V} y \mathbb{W} respectivamente $\Rightarrow \forall |a\rangle_1, |a\rangle_2 \in \mathbb{V}, |b\rangle_1, |b\rangle_2 \in \mathbb{W}, \alpha \in F$ se cumple que

- 1) $\alpha(|a\rangle_1 \otimes |b\rangle_1) = (\alpha|a\rangle_1) \otimes |b\rangle_1 = |a\rangle_1 \otimes (\alpha|b\rangle_1)$
- 2) $(|a\rangle_1 + |a\rangle_2) \otimes |b\rangle_1 = |a\rangle_1 \otimes |b\rangle_1 + |a\rangle_2 \otimes |b\rangle_1$
- 3) $|a\rangle_1 \otimes (|b\rangle_1 + |b\rangle_2) = |a\rangle_1 \otimes |b\rangle_1 + |a\rangle_1 \otimes |b\rangle_2$
- 4) $\hat{A} \otimes \hat{B}(|a\rangle_1 \otimes |b\rangle_1) = \hat{A}|a\rangle_1 \otimes \hat{B}|b\rangle_1$
- 5) Sean $|a\rangle_i \in \mathbb{V}, |b\rangle_i \in \mathbb{W}$ y $\alpha_i \in F \Rightarrow \hat{A} \otimes \hat{B}(\sum_i \alpha_i |a\rangle_i \otimes |b\rangle_i) = \sum_i \alpha_i \hat{A}|a\rangle_i \otimes \hat{B}|b\rangle_i$

El producto tensorial $|a\rangle \otimes |b\rangle$ también se puede escribir $|ab\rangle$ o $|a, b\rangle$. Por otra parte, el producto tensorial de $|a\rangle$ consigo mismo n veces $|a\rangle \otimes |a\rangle \otimes \dots \otimes |a\rangle$ se puede escribir como $|a\rangle^{\otimes n}$.

El producto de Kronecker es una representación matricial del producto tensorial. Sean $A = (a_{ij}), B = (b_{ij})$ dos matrices de orden $m \times n$ y $p \times q$ respectivamente. Entonces $A \otimes B$ se calcula de la siguiente manera:

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{pmatrix}.$$

$A \otimes B$ es de orden $mp \times nq$.

3 Probabilidad elemental para computación cuántica

3.1 Discrete Random variables and distributions

Definición 3.1. Discrete Random Variable. An experiment is a situation with a set of possible outcomes. Let us suppose we have an experiment with outcome space \mathcal{E} .

A **random variable** (rv) is a real mapping $X : \mathcal{E} \rightarrow \mathbb{R}$ that is defined for all possible outcomes in S . A **discrete random variable** (drv) takes only a finite or countable infinite number of distinct values, i.e. $X : \mathcal{E} \rightarrow A \subset \mathbb{R}$ is a drv iff $\#(A) \leq \aleph_0$. The expression $X = x_i$ is shorthand for $X(e_i) = x_i, \forall e_i \in \mathcal{E}, x_i \in A$.

Definición 3.2. Probability distribution for a drv. Let $X : \mathcal{E} \rightarrow A$ be a drv. Since the outcomes of an experiment are uncertain in general, we associate with each outcome $x_i \in A$ a probability $p(x_i)$, where $p(x_i) = \Pr(X = x_i)$. The numbers $p(x_i)$ are called a **probability distribution of X** iff *i*) $p(x_i) \geq 0$, and *ii*) $\sum_{x_i \in A} p(x_i) = 1$.

Definición 3.3. Expectation value and variance. The expectation value μ of a drv X , also known as **mean**, is defined as $E[X] = \sum_i x_i p(x_i)$. More generally, the expectation value of any function $g(X)$ of X is given by $E[g(X)] = \sum_i g(x_i) p(x_i)$. The **variance** $V[X]$ of a distribution, also written as σ^2 , is given by $V[X] = E[(X - \mu)^2] = \sum_i (x_i - \mu)^2 p(x_i)$. The square root of the variance is known as the **standard deviation** and is denoted by σ .

If X, Y are two drv and $a, b \in \mathbb{R}$, the following propositions can be proved ([58])

$$E[aX + bY] = aE[X] + bE[Y] \quad (17)$$

$$V[X] = E[X^2] - (E[X])^2 \quad (18)$$

$$\text{If } X, Y \text{ are independent drvs} \Rightarrow V[aX + bY] = a^2V[X] + b^2V[Y] \quad (19)$$

Definición 3.4. Bernoulli distribution. The Bernoulli distribution, denoted $\mathcal{B}(\theta)$, is used as a model for experiments which have only two outcomes: success with probability θ , and failure with probability $1 - \theta$. If X is $\mathcal{B}(\theta)$ then $X = 1$ if success and $X = 0$ if failure. It is a well known fact that if X is $\mathcal{B}(\theta)$ then $\mu_X = \theta$ and $\sigma^2 = \theta(1 - \theta)$.

Definición 3.5. Binomial distribution. The binomial distribution, denoted $\text{Bin}(n, p)$, describes experiments that consist of a number of independent identical trials with two possible outcomes: success with probability p and failure with probability $q = 1 - p$. So, the random variable $X = \text{'number of successes'}$ can take any value from $\{1, 2, \dots, n\}$ and its distribution is described by the binomial distribution. If X is $\text{Bin}(n, p)$ then the probability $p(r)$ of obtaining r successes from n trials is given by $p(r) = \binom{n}{r} p^r q^{n-r}$.

4 Postulados de la mecánica cuántica (versión computación cuántica)

La mecánica cuántica es una teoría sobre el comportamiento de la masa y la luz, en particular a escala atómica [57, 59]. La historia de la mecánica cuántica (1900- *circa* 1930) incluye un conjunto de resultados experimentales que pusieron en tela de juicio las ideas que sobre la Naturaleza se tuvieron hasta el principio del siglo XX ([60], [61] and [62]). Gracias al trabajo de W. Heisenberg y E. Schrödinger, el cual fue continuado por otros científicos como R. Feynman and M. Born, la mecánica cuántica es hoy una teoría científica robusta, utilizada diariamente en la labor teórica y experimental.

En mecánica cuántica existen dos formalismos usados para describir un sistema físico: vectores de estado y operadores de densidad. Ambos formalismos son equivalentes [7] y, en consecuencia, el uso de uno u otro depende los requerimientos del investigador. En este texto expresaremos los cuatro postulados de nuestro interés usando el primer formalismo, el que corresponde a vectores de estado.

En esta capítulo nos concentraremos en el estudio de los postulados de la mecánica cuántica, siguiendo la formulación que de los mismos se hace en [7]. Además, revisaremos varios resultados presentados en las obras [57], [63], [59], [8] y [7]. Entre muchas obras y artículos, se recomienda al lector interesado en profundizar en los temas aquí expuestos que consulte [64].

4.1 Espacio de estados

Este primer postulado consiste en la descripción matemática de sistemas físicos aislados, esto es, aquellos que no están en contacto con ningún otro sistema físico.

Postulado 1. A cada sistema físico aislado asociaremos un espacio de Hilbert \mathcal{H} , el cual recibe el nombre de *espacio de estados*.

La descripción total y absoluta de las características que nos interesan del sistema físico en cuestión se encuentra contenida en su *vector de estado*, el cual es un vector unitario $|\psi\rangle \in \mathcal{H}$. La dimensión del espacio de estados \mathcal{H} depende del número de grados de libertad de la propiedad física en consideración.

Este primer postulado tiene una implicación muy importante: una combinación lineal de vectores de estado es también un vector de estado [57]. Este es el **principio de superposición** y es una característica fundamental de la descripción cuántica de sistemas físicos. En particular, observe que cualquier vector de estado $|\psi\rangle$ se puede expresar como una superposición de vectores de estado pertenecientes a una base de \mathcal{H} , i.e. $|\psi\rangle = \sum_i c_i |e_i\rangle$, $c_i \in \mathbb{C}$.

4.1.1 El qubit

En la teoría de la computación clásica la unidad fundamental de almacenamiento y manipulación de información es el *bit*, cuya estructura matemática es bastante simple: basta con definir dos valores tradicionalmente etiquetados como $\{0, 1\}$ y con relacionar dichas etiquetas con dos resultados posibles generados a través de una medición clásica. Un ejemplo de este procedimiento es tomar un transistor TTL como el sistema físico a medir y hacer la siguiente asignación:

- Si la diferencia de potencial entre colector y emisor se encuentra en el conjunto $[0, 0.5]V$ entonces hemos leído un '0' lógico.

- Si la diferencia de potencial entre colector y emisor se encuentra en el conjunto $[4.5, 5]V$ entonces hemos leído un '1' lógico.

Así pues, es evidente que la descripción matemática de un bit clásico es un elemento de un espacio escalar.

En computación cuántica, la unidad básica de almacenamiento, manipulación y medición de información es el *qubit*. Un qubit es un sistema físico cuyo comportamiento se describe a través de las leyes de la mecánica cuántica y que se puede representar matemáticamente como un vector unitario en un espacio de Hilbert bidimensional, esto es

$$|\psi\rangle = \alpha|p\rangle + \beta|q\rangle \quad (20)$$

donde $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ y $\{|p\rangle, |q\rangle\}$ es una base cualquiera de \mathcal{H}^2 . Es común que la base de elección sea $\{|0\rangle, |1\rangle\}$, la llamada base computacional o canónica de \mathcal{H}^2 .

Como se puede observar de la Ec. (20), un qubit $|\psi\rangle$ es una superposición de los estados base $|p\rangle$ y $|q\rangle$, la cual se puede preparar en un número infinito de formas sólo con modificar los valores de los coeficientes $\alpha, \beta \in \mathbb{C}$, siempre sujetos a la restricción de normalización.

La Ec. (20) también se puede escribir de la siguiente manera:

$$|\psi\rangle = e^{i\gamma} \left(\cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \right) \quad (21)$$

donde $\gamma, \theta, \varphi \in \mathbb{R}$. Puesto que $e^{i\gamma}$ no tiene efectos experimentales [7], podemos prescindir de este factor. En consecuencia,

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \quad (22)$$

Los números θ y φ definen un punto en la esfera unitaria conocida como la **esfera de Bloch** (Fig. (1)).

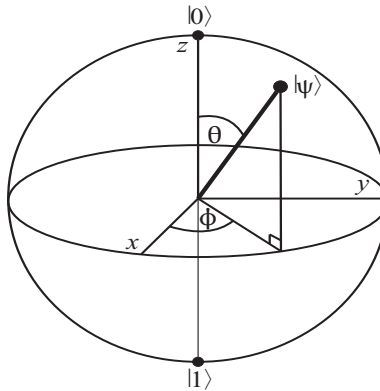


Figure 1: Un qubit como elemento de la esfera de Bloch $|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle$.

4.2 Evolución de un sistema cuántico aislado

En este apartado estudiaremos la de la evolución de un sistema cuántico, esto es, el formalismo matemático que describe el comportamiento temporal de un sistema físico aislado, de acuerdo a las leyes de la mecánica cuántica.

Postulado 2 (versión operador unitario).

Sea $|\Psi\rangle$ el vector de estado de un sistema cuántico aislado. La evolución de $|\Psi\rangle$ se describe mediante un operador unitario \hat{U} (Def. (2.24)), también conocido como operador de evolución. Luego, el estado del sistema en el tiempo t_2 dado el vector de estado del mismo sistema en el tiempo t_1 es:

$$|\Psi(t_2)\rangle = \hat{U}|\Psi(t_1)\rangle. \quad (23)$$

El postulado 2 sólo describe las características matemáticas que debe cumplir un operador de evolución cualquiera. Las propiedades particulares que debe tener \hat{U} a fin de reflejar la naturaleza de la evolución de cierto sistema físico dependen de este mismo sistema.

El postulado 2 también se puede escribir en una forma más tradicional, usando la famosa ecuación de Schrödinger.

Postulado 2 (versión operador hermitiano). La evolución de un sistema cuántico aislado está dada por la ecuación de Schrödinger:

$$i\hbar \frac{d|\psi\rangle}{dt} = \hat{H}|\psi\rangle \quad (24)$$

donde \hbar es la constante de Planck y \hat{H} es un operador hermitiano (Eq. (2.22)) conocido como el *Hamiltoniano* del sistema.

The Hamiltonian of particular physical systems must be determined and calculated for each case. In general, figuring out the Hamiltonian of a particular physical system is a difficult task.

Nota aclaratoria. Tenga en mente que, a pesar de lo similar de la notación, \hat{H} y \hat{H} representan dos cosas distintas: el primero es el hamiltoniano al que se hace referencia en el postulado 2, en tanto que el último es el operador hadamard.

Veamos el efecto del operador Hadamard (Eq. (8)), usado como operador de evolución, sobre un qubit.

$$\begin{aligned} \hat{H}|0\rangle &= \frac{1}{\sqrt{2}}[|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|]|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \hat{H}|1\rangle &= \frac{1}{\sqrt{2}}[|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|]|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

Ejercicio 4.1. Sean los siguientes operadores unitarios:

$$\hat{\sigma}_x = |0\rangle\langle 1| + |1\rangle\langle 0|, \hat{\sigma}_y = i|0\rangle\langle 1| - i|1\rangle\langle 0|, \hat{\sigma}_z = |0\rangle\langle 0| - |1\rangle\langle 1|, \text{ y } \hat{H} = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)$$

y los qubits con los siguientes estados cuánticos:

$$|\psi\rangle_1 = |0\rangle, |\psi\rangle_2 = |1\rangle, \text{ y } |\psi\rangle_3 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Aplique dos veces cada operador unitario a cada qubit.

4.3 Medición de un sistema cuántico

En la teoría de la mecánica cuántica, la medición de propiedades de sistemas físicos es un proceso alejado de la intuición debido a las siguientes razones:

1. La medición en mecánica cuántica es intrínsecamente probabilística. Esto significa que, sin importar el detalle y control que tengamos sobre un experimento, la generación de los resultados obtenidos en la medición de una propiedad física obedece una función de distribución (o función de densidad, según sea el caso).
2. Además, al momento de llevar a cabo una medición, el estado del sistema físico en cuestión se altera, de forma inevitable, debido a la interacción que dicho sistema tiene con el aparato de medición. Esto significa que, en general, el estado cuántico que describe al sistema *antes* de la medición es distinto del estado que describe a este mismo sistema *después* de ser medido.

Las siguientes líneas contienen la versión del postulado de medición más frecuentemente usada en computación cuántica.

Postulado 3. Medición proyectiva. Una medición proyectiva es descrita por un *observable* \hat{M} , el cual es un operador hermitiano definido en el espacio de estados que se desea observar. El observable \hat{M} se puede escribir, gracias al teorema de la descomposición espectral, de la siguiente manera:

$$\hat{M} = \sum_i r_i \hat{P}_{r_i}$$

donde \hat{P}_{r_i} es el proyector al eigensubespacio $E(r_i)$ definido por el eigenvalor r_i . Los resultados posibles de la medición corresponden a los eigenvalores r_i del observable.

Este postulado provee de medios para cuantificar la función de distribución que determina las frecuencias relativas correspondientes a las funciones de distribución de resultados.

Sea $|\psi\rangle$ el vector de estado de un sistema cuántico *inmediatamente antes de la medición*. Entonces, la probabilidad de obtener el resultado r_i se calcula usando la siguiente expresión:

$$p(r_i) = \langle \psi | \hat{P}_{r_i} | \psi \rangle \quad (25)$$

Y el estado de post-medición asociado al resultado r_i es:

$$|\psi\rangle_{pm} = \frac{\hat{P}_{r_i} |\psi\rangle}{\sqrt{p(r_i)}} \quad (26)$$

Veamos un ejemplo. Suponga que tiene en su poder un fotón polarizado con orientaciones de polarización vertical y horizontal. Simbolizamos a la polarización horizontal con el vector $|0\rangle$ y a la polarización vertical con $|1\rangle$. Luego, la polarización inicial de nuestro fotón se puede describir con la expresión

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

donde $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ y $\{|0\rangle, |1\rangle\}$ conforman la base computacional de \mathcal{H}^2 .

Ahora construyamos dos operadores de proyección $\hat{P}_{a_0} = |0\rangle\langle 0|$ y $\hat{P}_{a_1} = |1\rangle\langle 1|$, los cuales corresponden a los resultados a_0, a_1 . Entonces, el *observable* utilizado en este experimento es

$$\hat{M} = a_0|0\rangle\langle 0| + a_1|1\rangle\langle 1|$$

Con esta información y de acuerdo al postulado 3, podemos decir lo siguiente:

1) Hay sólo dos resultados posibles en la medición de la polarización de nuestro fotón: a_0 y a_1 .

2) La probabilidad de obtener el resultado a_0 es, de acuerdo a la Ec. (25):

$$p(a_0) = \langle \psi | \hat{P}_{a_0} | \psi \rangle = (\langle 1 | \beta^* + \langle 0 | \alpha^*) \hat{P}_{a_0} (\alpha | 0 \rangle + \beta | 1 \rangle) = |\alpha|^2$$

3) Si la medición efectivamente arroja el resultado $a_0 \Rightarrow$ el estado de postmedición sería, de acuerdo a la Ec. (26)s:

$$\frac{\hat{P}_{a_0} |\psi\rangle}{\sqrt{p(a_0)}} = \frac{|0\rangle\langle 0|(\alpha|0\rangle + \beta|1\rangle)}{\sqrt{|\alpha|^2}} = |0\rangle$$

4) De forma análoga, la probabilidad de obtener el resultado a_1 es, de acuerdo a la Ec. (25):

$$p(a_1) = \langle \psi | \hat{P}_{a_1} | \psi \rangle = (\langle 1 | \beta^* + \langle 0 | \alpha^*) \hat{P}_{a_1} (\alpha | 0 \rangle + \beta | 1 \rangle) = |\beta|^2$$

5) Si la medición efectivamente arroja el resultado $a_1 \Rightarrow$ el estado de postmedición sería dado por la Ec. (26), esto es:

$$\frac{\hat{P}_{a_1} |\psi\rangle}{\sqrt{p(a_1)}} = \frac{|1\rangle\langle 1|(\alpha|0\rangle + \beta|1\rangle)}{\sqrt{|\beta|^2}} = |1\rangle$$

Ejercicio 4.2. Sea

$$|\psi\rangle = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \in \mathcal{H}^2$$

Definimos además dos bases de \mathcal{H}^2 :

$$B_1 = \{|0\rangle, |1\rangle\}$$

y

$$B_2 = \left\{ |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right\}$$

Ahora, definimos los siguientes proyectores:

$$\begin{aligned} \hat{P}_0 &= |0\rangle\langle 0|, \hat{P}_1 = |1\rangle\langle 1| \\ \hat{P}_+ &= |+\rangle\langle +|, \hat{P}_- = |-\rangle\langle -| \end{aligned}$$

Calcule probabilidades y estados de postmedición de $|\psi\rangle$ usando \hat{P}_0, \hat{P}_1 y \hat{P}_+, \hat{P}_-

4.4 Sistemas cuánticos multipartitas

We now focus on the mathematical description of a composite quantum system, i.e. a system made up of several different physical systems.

Postulate 4. The state space of a composite quantum system is the tensor product of the component system state spaces.

- If we have n quantum systems expressed as *state vectors*, labeled $|\psi\rangle_1, |\psi\rangle_2, \dots, |\psi\rangle_n$ then the joint state of the total system is given by $|\psi\rangle_T = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \dots \otimes |\psi\rangle_n$.

- Similarly, if we have n quantum systems expressed as *density operators* $\rho_1, \rho_2, \dots, \rho_n$ then the joint state of the total system is given by $\rho_T = \rho_1 \otimes \rho_2 \otimes \dots \otimes \rho_n$ (in the absence of any knowledge of correlations).

As an advance of the operations we shall perform on the following chapters let us show the details of applying an evolution operator to a composite quantum system. Let $\hat{H}^{\otimes 2}$ be the tensor product of the Hadamard operator (Eq. (8)) with itself and let $|\psi\rangle = |00\rangle$. Then

$$\begin{aligned} \hat{H}^{\otimes 2} |\psi\rangle &= \frac{1}{2}(|00\rangle\langle 00| + |01\rangle\langle 00| + |10\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 01| - |01\rangle\langle 01| + |10\rangle\langle 01| - |11\rangle\langle 01| \\ &+ |00\rangle\langle 10| + |01\rangle\langle 10| - |10\rangle\langle 10| - |11\rangle\langle 10| + |00\rangle\langle 11| - |01\rangle\langle 11| - |10\rangle\langle 11| + |11\rangle\langle 11|)|00\rangle \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \quad (27) \end{aligned}$$

5 Theory of Computation

The purpose of this chapter is to present a concise introduction to the *Theory of Computation*, in order to provide the necessary background and to motivate our further discussion on the importance of classical random walks and quantum walks in computer science.

We begin by providing an overview of the theory of computation and deliver a succinct historical background of those ideas that led to its creation and development. We then formulate the essential concepts of a basic and yet powerful model of computation: Finite Automata. We use these concepts to introduce a formal definition of a model of computation that has played a most important role in the development of Computer Science: Turing Machines. We extend our discussion by introducing a third model of computation: Quantum Turing Machines. The previous concepts are followed by key definitions and theorems from Complexity Theory and the definitions of **P**, **NP** and **NP-complete** problem categories. This chapter is based on [65], [66], [67], [68] and [70].

5.1 What is the Theory of Computation?

The Theory of Computation is a scientific field devoted to understanding the fundamental capabilities and limitations of computers, and it is divided into three areas:

1. *Automata Theory*. The study of different models of computation.
2. *Computability Theory*. This focuses on determining which problems can be solved by computers and which cannot.
3. *Complexity Theory*. The objective in this area is to understand what makes some problems computationally hard and other easy.

The development of the theory of computation was driven in great part by several challenges posed by D. Hilbert and other mathematicians on the foundations of mathematics at the beginning of the 20th Century. A. Turing and other scientists, while working on the ideas required to formalize the idea of computation, answered some of the questions posed by Hilbert *et al.*

5.2 Hilbert's program and the Entscheidungsproblem

At the beginning of the 20th century D. Hilbert and other mathematicians were aware that the methods of reasoning in mathematics could in some cases lead to contradictions. Hilbert believed that the proper way to develop any scientific subject rigorously required an axiomatic approach. In providing an axiomatic treatment, the theory would be developed independently of any need for intuition.

Hilbert's first step towards a rigorous formulation of mathematics was undertaken in his lecture at the International Congress of Mathematicians (Paris, 1900) [71], where he stated that "it shall be possible to establish the correctness of the solution by means of a finite number of steps based upon a finite number of hypotheses which are implied in the statement of the problem and which must always be formulated . . . This conviction of solvability of every mathematical problem is a powerful incentive to the worker. We hear within us the perpetual call: There is the problem. Seek its solution. You can find it by pure reason, for in mathematics there is no *ignorabimus*³".

³Short for *ignoramus et ignorabimus*: we do not know and will not know.

The second step towards a rigorous formulation of mathematics was taken in 1920-1921, when Hilbert's program was proposed [72]. Hilbert thought that, in order to formulate mathematics on a solid and complete logical foundation, it would suffice to prove that "all of mathematics follows from a correctly-chosen finite system of axioms, as well as that some such axiom system is provably consistent." Finally, Hilbert and Ackerman posed a challenge known as the Entscheidungsproblem (**Decision Problem**) [73]. The Decision Problem is formulated in terms of first-order logic⁴ and can be stated as follows

Definition 5.1. Entscheidungsproblem. *Does there exist a procedure which can be followed for a finite number of steps in order to determine the validity of a given first-order statement?*

The idea of 'finite procedure' was deeply rooted in Hilbert's proposals. A contemporary computer scientist would immediately think of an 'algorithm' as an equivalent definition of 'finite procedure'. However, in Hilbert's time the concept of algorithm did not exist yet.

Alan Turing published a most influential paper in 1936 [69] in which he pioneered the theory of computation, introducing the famous abstract computing machines now known as *Turing Machines*. In [69], Turing explained the fundamental principle of the modern computer, the idea of controlling the machine's operation by means of a program of coded instructions stored in the computer's memory, i.e. Turing showed that it was possible to build a "Universal Turing Machine", that is, a Turing machine capable of simulating any other Turing machine (the Universal Turing Machine being the actual digital computer and the simulated Turing machine the program that has been encoded in the digital computer's memory). In addition, Turing proved that not all precisely stated mathematical problems can be solved by computing machines, in particular the Entscheidungsproblem.

In the following section we deliver a brief summary of ideas and main results from [69].

5.3 On Computability

When Turing wrote [69], **a computer was not a machine at all, but a human being**. A computer was a mathematical assistant who calculated by rote, in accordance with a systematic method. The method was supplied by an overseer prior to the calculation. It is in that sense that Turing uses the word 'computer' in [69] and a Turing machine is an idealized version of this human computer. What Turing did in [69] was to propose a mathematical formalism for the idealization of a human computer as well as to study the calculation capabilities and limitations of that mathematical model.

Turing meant by a systematic method (sometimes called an *effective* method and a *mechanical* method) any mathematical method for which all the following are true:

1. The method can, in practice or in principle, be carried out by a human computer working with paper and pencil.
2. The method can be given to a human computer in the form of a *finite* number of instructions.
3. The method demands neither insight nor ingenuity on the part of the human being carrying it out.
4. The method will definitely work if carried out without error.

⁴First-order logic or First Order Predicate Calculus (FOPC) is a formalisation of deductive logical reasoning. A concise tutorial on FOPC can be found in [65].

Turing’s definition of a systematic method is the definition of an **algorithm**. Also, Turing proved that it was possible to build a particularly powerful machine called **Universal Turing Machine (UTM)** that could simulate any other Turing machine in reasonable time. Furthermore, Turing stated a conjecture now known as the **Church-Turing Thesis**, in which he established an equivalence correspondence between the existence of Turing machines and that of systematic methods. If the Church-Turing thesis is correct, then the existence or non-existence of systematic methods can be replaced throughout mathematics by the existence or non-existence of Turing machines. For instance, one could establish that there is no systematic method for doing a certain task by proving that no Turing machine can do the task in question.

5.1. The Church-Turing Thesis. *Three ways to express the thesis are:*

1. *The UTM can perform any calculation that any human computer can carry out.*
2. *Any systematic method can be carried out by the UTM.*
3. *Every function which would be naturally regarded as computable can be computed by the Universal Turing Machine.*

Turing proved that it was not possible to build a Turing machine capable of solving problem from Def. (5.1), i.e. the Entscheidungsproblem is undecidable. If the Church-Turing thesis is true that means that the problem posed in Def. (5.1) cannot be solved by any algorithm, i.e. any finite method, as required by Hilbert. In this thesis we shall focus only on decidable problems, that is, problems for which it is possible to build Turing machines to solve them.

In the following section we deliver some general properties of problems suitable to be solved by Turing machines, along with two relevant problems in Computer Science.

5.4 Definition of a Problem and two examples

We define a problem as a general question, usually possessing several parameters. A problem is specified by giving a general description of all its parameters, and a statement of what properties the solution is required to satisfy. An instance of a problem is obtained by specifying particular values for all the problem parameters. In general, we are interested in finding the “most efficient” algorithm for solving a problem, i.e. the fastest algorithm.

The description of a problem instance is a finite string of symbols under a particular and reasonable encoding scheme, which maps problem instances into the strings describing them. To do this mapping we use the concept of *language*:

Definition 5.2. Language. *For any finite set of symbols Σ , we denote Σ^* the set of all finite strings of symbols from Σ . If $L \subset \Sigma^*$ then L is a language over the alphabet Σ .*

For example, let $\Sigma = \{0, 1\}$. Then, $\Sigma^ = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}$ where ϕ is the empty string. Thus, $L = \{01, 001, 111, 10100101, 1111, 0001\}$ is a language over Σ , as is the set of all binary representations of integers that are perfect cubes.*

The *input length* for an instance I of a problem ζ is the number of symbols in the description of I obtained from the encoding scheme of ζ . An algorithm **solves** a problem ζ if that algorithm can be applied to any instance I of ζ and is guaranteed always to produce a solution for that instance I . Two important problems in Computer Science are:

Definition 5.3. The Traveling Salesman Problem

INSTANCE: A finite set $C = \{c_1, c_2, \dots, c_m\}$ of “cities” and a “distance” $d(c_i, c_j) \in \mathbb{N}$, the set of natural numbers.

QUESTION: Which is the shortest “tour” of all the cities in C , that is, an ordering $[c_{\Pi(1)}, c_{\Pi(2)}, \dots, c_{\Pi(m)}]$ of C such that $[\sum_{i=1}^{m-1} d(c_{\Pi(i)}, c_{\Pi(i+1)})] + d(c_{\Pi(m)}, c_{\Pi(1)})$ is minimum?

Definition 5.4. The Satisfiability (SAT) Problem

Let $S = \{x_1, x_2, \dots, x_n\}$ be a set of Boolean variables. A truth assignment for S is a function $t : S \rightarrow \{T, F\}$, for which if $t(x_i) = T$ we say that x_i is TRUE under t , and FALSE if $t(x_i) = F$. If x_i is a variable under S then x_i and \bar{x}_i are literals over S . A clause over S is the disjunction of a set of literals over S (such as $x_1 \vee x_2 \vee \bar{x}_4$) and it is satisfied by a truth assignment iff at least one of its members x_i is true under that assignment.

A collection C of clauses over S is satisfiable iff there exists some truth assignment for S that simultaneously satisfies all the clauses in C , i.e. C is a conjunction of disjunctions $C = \bigwedge_i [(\bigvee_j x_j)]$. Such a truth assignment is called a satisfying truth assignment for C .

INSTANCE: A set S of variables and a collection C of clauses over S .

QUESTION: Is there a satisfying truth assignment for C ?

In the theory of computation we usually work with decision problems, the reason being the need to build operational definitions of relevant problems. It is a matter of ingenuity to design a decision-problem version of a problem and it is not always the case that totally equivalent versions are obtained. Let us provide a decision-version of problem 5.3 as an example (note that the SAT problem (5.4) is already a decision problem).

Definition 5.5. The Traveling Salesman Problem (Decision problem version)

INSTANCE: A finite set $C = \{c_1, c_2, \dots, c_m\}$ of “cities” a “distance” $d(c_i, c_j) \in \mathbb{N}$ and a bound $B \in \mathbb{N}$.

QUESTION: Is there a “tour” of all the cities in C having total length no more than B , that is, an ordering $[c_{\Pi(1)}, c_{\Pi(2)}, \dots, c_{\Pi(m)}]$ of C such that $[\sum_{i=1}^{m-1} d(c_{\Pi(i)}, c_{\Pi(i+1)})] + d(c_{\Pi(m)}, c_{\Pi(1)}) \leq B$?

We restrict ourselves to work with decision problems because languages, as defined in Def. (5.2), are their natural counterpart, suitable to study in a mathematical fashion. The correspondence between decision problems and languages is brought about by the encoding schemes used to specify problem instances. An encoding scheme used describe each instance of a problem ζ partitions language Σ^* into three classes of strings: strings that are not encoding of instances of ζ , those that encode instances of ζ for which the answer is ‘no’ and those that encode instances of ζ for which the answer is ‘yes’. Since we work with decidable problems, we are interested in the third class of strings.

In the following section we present definitions and main results of three models of computation: Finite Automata, Turing Machines and Quantum Turing Machines.

5.5 Models of Computation and Algorithmic Complexity

Finite Automata, Turing Machines and Quantum Turing Machines are three models of computation. In this chapter, Finite Automata are presented and its results are used to introduce Turing machines; both models of computation are presented in their deterministic and nondeterministic versions. Quantum Turing machines are introduced along with a physics-oriented version of the Church-Turing thesis. Before introducing the above mentioned models, we will provide some concepts to quantify the amount of resources required to find an algorithmic solution to a problem.

5.5.1 Asymptotic Notation

The performance of models of computation in the execution of an algorithm is a fundamental topic in the theory of computation. Since the quantification of resources (in our case, we focus on time) needed to find a solution to a problem is usually a complex process, we just estimate it. To do so, we use a form of estimation called **Asymptotic Analysis** in which we are interested in the maximum number of steps S_m that an algorithm must be run on large inputs. We do so by considering only the highest order term of the expression that quantifies S_m . For example, the function $F(n) = 18n^6 + 8n^5 - 3n^4 + 4n^2 - \pi$ has five terms, and the highest order term is $18n^6$. Since we disregard constant factors, we then say that f is asymptotically at most n^6 . The following definition formalises this idea.

Definition 5.6. Big O Notation. *Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. We say that $f(n) = O(g(n))$ if $\exists \alpha, n_o \in \mathbb{N}$ such that $\forall n \geq n_o$*

$$f(n) \leq \alpha g(n)$$

So, $g(n)$ is an asymptotic upper bound for $f(n)$ (“ f is of the order of g ”). Bounds of the form n^β , $\beta > 0$ are called **polynomial bounds**, and bounds of the form 2^{n^γ} , $\gamma \in \mathbb{R}^+$ are called **exponential bounds**. $f(n) = O(g(n))$ means informally that f grows as g or slower.

Big O notation says that one function is asymptotically no more than another. To state that one function is asymptotically no less than another we use the Ω notation.

Definition 5.7. Ω Notation. *Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. We say that $f(n) = \Omega(g(n))$ if $\exists \alpha, n_o \in \mathbb{N}$ such that $\forall n \geq n_o$*

$$g(n) \leq \alpha f(n)$$

Finally, to say that two functions grow at the same rate we use the Θ notation.

Definition 5.8. Θ Notation. *Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. We say that $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. Thus, $f(n) = \Theta(g(n))$ means that f and g have the same rate of growth.*

5.5.2 Deterministic Finite Automata

Deterministic Finite Automata is a model for computers with an extremely limited amount of memory. An example of a Deterministic Finite Automaton (DFA) is a machine R used to determine the parity of a sequence of characters like the one shown in Fig. (2).

Suppose that R is at one of the ends of a wireless communication system. R can receive as valid input the characters ‘0’ and ‘1’, and, since any communication system is subject to errors, R can receive a third character ‘#’ which is used to state that an error has occurred in the transmission of the data stream. So, the input for R is an arbitrarily long set of characters (also known as a string) taken from the alphabet $\Sigma = \{0, 1, \#\}$. The parity of a sequence of 0s and 1s is defined as follows: a sequence of 0s and 1s is odd if its number of 1s is odd, and it is even if its number of 1s is even.

Let us now elaborate on the properties R must have. First, we state that only strings with no errors will be suitable for parity computation. Thus, only sequences of 0s and 1s will be accepted and any string has at least one error character must be rejected. Second, an empty set of characters has no 1s, thus we set the initial parity of a string as even.

Let us show the behaviour of R with an example. Suppose that R receives as input the string 100110001 (read from left to right). The first input character is ‘1’ thus R goes from state ‘Even’ to state ‘Odd’. We then read ‘0’ and therefore we stay in state ‘Odd’. The third input character is again a ‘0’ and we remain in state ‘Odd’. As fourth input we receive a ‘1’ and then we move from state ‘Odd’ to state ‘Even’ as now we have an even number of ‘1’s in our account. The fifth input is a ‘1’ and consequently we move from state ‘Even’ to state ‘Odd’ as the total number of ‘1’s we have received so far is odd. The 6th, 7th and 8th characters are ‘0’ thus the current state of machine R remains being ‘Odd’. Finally, the last input character is ‘1’ and therefore R goes from state ‘Odd’ to state ‘Even’. The final outcome of the computation is the accept state ‘Even’.

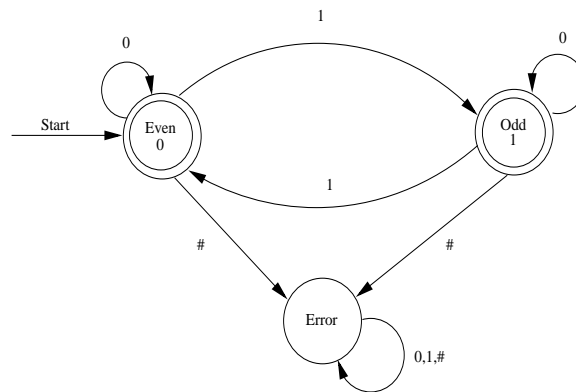


Figure 2: A finite automaton R for parity computation. Double-circled states are accept states and single-circled state is a reject state. Input strings for R are taken from the set of any combination of characters taken from the alphabet $\Sigma = \{0, 1, \#\}$.

Formally speaking, a DFA is a list of five objects: set of states, input alphabet, rules for moving, start state, and accept states. We use a **transition function** to define the rules for moving. If the DFA has an arrow from state x to a state y labeled with the input symbol 1, that means that, if the automaton is in state x when it reads a 1, it then moves to state y . We can indicate the same thing with the transition function by saying that $\delta(x, 1) = y$.

Definition 5.9. Deterministic Finite Automaton. A DFA R is a 5-tuple $(Q, \Sigma, \delta, q_o, F)$, where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subset Q$ is the set of accept states.

The formal description of the DFA R depicted in Fig. (2) is as follows: $Q = \{\text{Even}, \text{Odd}, \text{Error}\}$, $\Sigma = \{0, 1, \#\}$, $q_0 = \text{Even}$, $F = \{\text{Even}, \text{Odd}\}$ and $L(R) = \{x \in \{0, 1\}^n\}$. The transition function is given in Table 1.

Table 1. Transition Function δ

$\delta(\text{Even}, 0) = \text{Even}$	$\delta(\text{Even}, 1) = \text{Odd}$	$\delta(\text{Even}, \#) = \text{Error}$
$\delta(\text{Odd}, 0) = \text{Odd}$	$\delta(\text{Odd}, 1) = \text{Even}$	$\delta(\text{Odd}, \#) = \text{Error}$
$\delta(\text{Error}, 0) = \text{Error}$	$\delta(\text{Error}, 1) = \text{Error}$	$\delta(\text{Error}, \#) = \text{Error}$

Definition 5.10. Computation with a Deterministic Finite Automaton. Let $R = (Q, \Sigma, \delta, q_o, F)$ be a DFA and let $w = w_1w_2 \dots w_n$ be a string where each w_i is a member of the alphabet Σ . Then R **accepts** w if a sequence of states r_0, r_1, \dots, r_n in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \dots, n - 1$, and
3. $r_n \in F$.

Condition 1 means that the machine starts in the start state. Condition 2 states that the machine goes from state to state according to the transition function. Condition 3 says that the machine accepts its input if it ends up in an accept state. We say that R **accepts** language A if $A = \{w \mid R \text{ accepts } w\}$, i.e. A is the set of all strings accepted by R .

5.5.3 Nondeterministic Finite Automata

When every step of a computation follows in a unique way from the preceding step (as in a DFA) we are doing deterministic computation. In a nondeterministic machine, several choices may exist for the next state at any point. Non determinism is a generalization of determinism, so every DFA is automatically a Nondeterministic Finite Automaton (NFA).

How does an NFA compute? Suppose that we are running an NFA on an input string and come to a state with multiple states to proceed. For example, say that we are in state q_i in NFA N_1 and that the next input symbol is 1. After reading that symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel. Each copy of the machine takes one of the possible ways to proceed and continues as before. If there are subsequent choices, the machine splits again. If the next input symbol does not appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it. Finally, if any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string.

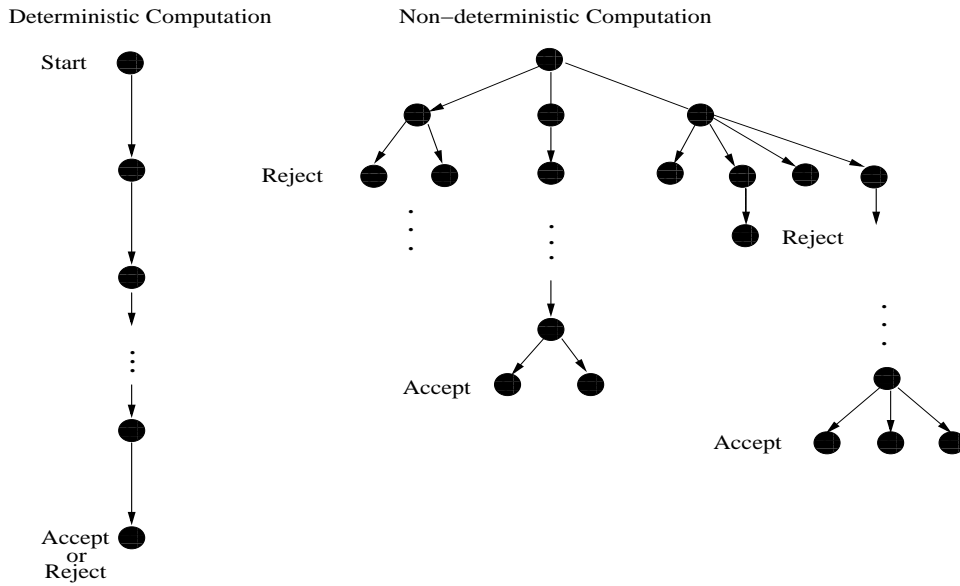


Figure 3: In a DFA, every single step is fully determined by the previous step. In an NFA, a step may be followed by n new steps or, equivalently, an NFA makes n copies of itself, one for each possibility.

Another way to think of a nondeterministic computation is as a tree of possibilities. The root of the tree corresponds to the start of the computation. Every branching point in the tree corresponds to a point in the computation at which the machine has multiple choices. The machine accepts if at least one of the computation branches ends in an accept state. A graphical illustration of a nondeterministic computation is given in Fig. (3).

An NFA is not a fully realistic model of computation as it assumes the capability of producing several instances of NFAs to run in parallel (it would be like suddenly producing as many computers as instances for each computation step). However, nondeterminism may be viewed as a kind of parallel computation wherein multiple independent processes can be running concurrently, and this view does prepare the grounds for introducing the concept of probabilistic computation, which will be reviewed in the following pages of this chapter.

Definition 5.11. Nondeterministic Finite automaton. An NFA R_N is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where 1) Q is a finite set of states; 2) Σ is a finite alphabet; 3) $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ is the transition function; (\mathcal{P} is the power set of set Q); 4) $q_0 \in Q$ is the start state, and 5) $F \subseteq Q$ is the set of accept states.

Definition 5.12. Computation on a Nondeterministic Finite Automaton. Let $R_N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ . Then we say that R_N **accepts** w if we can write w as $w = y_1 y_2 \dots y_m$, where each y_i is a member of Σ and a sequence of states r_0, r_1, \dots, r_m exists in Q with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \dots, m - 1$, and
3. $r_m \in F$.

Condition 1 states that the machine starts out in the start state. Condition 2 means that state r_{i+1} is one of the allowable next states when N is in state r_i and reading y_{i+1} . Observe that $\delta(r_i, y_{i+1})$ is the *set* of allowable next states and so we say that r_{i+1} is a member of that set. Finally, condition 3 establishes that the machine accepts its input if the last state is an accept state. We say that R_N **accepts** language A if $A = \{w | R_N \text{ accepts } w\}$, i.e. A is the set of all strings accepted by R_N .

It can be proved that DFAs and NFAs recognize the same class of languages [68]. However, *the number of states of a deterministic counterpart of an NFA can be exponential in the number of branches of such an NFA* and this is a very important difference between these two models of computation.

5.5.4 Deterministic Turing Machines

Similar to a DFA but with an unlimited and unrestricted memory, a Deterministic Turing Machine (DTM) is a much more accurate model of a general purpose computer. A DTM, pictured schematically in Fig. (4), can do everything a real computer can do.

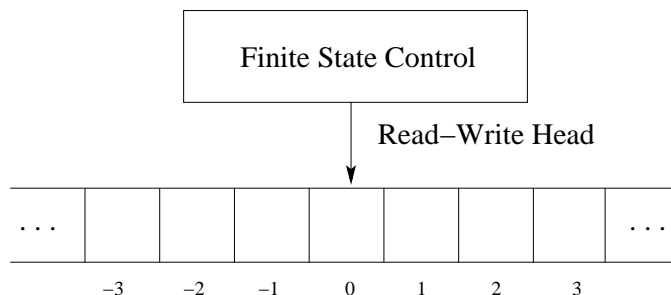


Figure 4: The ‘hardware’ elements of a Deterministic Turing Machine (DTM) are a limitless memory-type (the tape is divided into squares or cells), and a scanner which consists of read-write head *plus* a finite state control system. The scanner has two purposes: to read and write information on the cells of the tape as well as to control the state of the DTM.

A DTM consists of a scanner and a limitless memory-tape that moves back and forth past the scanner. The scanner is composed of a read-write head as well as a finite state control system. The scanner does two tasks: it controls the state of the DTM through the finite state control system, and reads and writes information on the memory cells through the read-write head. The tape is divided into squares. Each square may be blank (\square) or may bear a single symbol (0 or 1, for example). The scanner is able to examine only one square of tape at a time (the ‘scanned square’). The scanner has mechanisms that enable it to write and erase the symbol on the scanned square, and to move the tape to the left or right, one square at a time. Also, the scanner is able to alter the state of the machine: a device within the scanner is capable of adopting a number of different states, and the scanner is able to alter the state of this device whenever necessary. The operations just described - erase, print, move, and change state - are the basic operations of a DTM. Complexity of operation is achieved by chaining together large numbers of these simple basic operations.

Note that according to the definitions of effective procedure and Turing machines, and under the assumption that the Church-Turing thesis holds, the concepts of Turing machine and algorithm are interchangeable.

Definition 5.13. Deterministic Turing Machine. A *Deterministic Turing Machine (DTM)* is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol \sqcup ,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$.

Definition 5.14. Computation with a Deterministic Turing Machine.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ be a DTM.

Initially M receives its input $w \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is filled with blank symbols. The head starts on the leftmost square of the tape (Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.) Once M has started, the computation proceeds according to the rules described by δ . The computation continues until it enters either the accept or reject states at which point it halts. If neither occurs, M just does not stop.

As a DTM computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a **configuration** of the DTM. A configuration C_1 yields configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step. In an accepting configuration the state of the configuration is q_{accept} . In a rejecting configuration the state of the configuration is q_{reject} . Accepting and rejecting configurations are halting configurations and do not yield further configurations.

A DTM M **accepts** input w if a sequence of configurations C_1, C_2, \dots, C_k exists, where

1. C_1 is the start configuration of M on input w ,
2. each C_i yields C_{i+1} , and
3. C_k is an accepting configuration.

We say that M **accepts** language A if $A = \{w \mid M \text{ accepts } w\}$, i.e. A is the set of all strings accepted by M . We now show how to use a DTM to recognize all numbers divisible by 4.

Example 5.1. Running a program in a Deterministic Turing Machine. The program specified by $\Gamma = \{0, 1, \sqcup\}$ (\sqcup is the blank symbol), $Q = \{q_0, q_1, q_2, q_3, q_{\text{accept}}, q_{\text{reject}}\}$ and δ , provided in Table 1, is used in a deterministic Turing machine M to determine whether a number can be divided by 4 or, equivalently, whether the last two digits of a binary number, reading from left to right, are two consecutive zeros.

Table 1. Example of a deterministic Turing machine.

Q/Γ	0	1	\sqcup
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(q_1, \sqcup, L)
q_1	(q_2, \sqcup, L)	(q_3, \sqcup, L)	(q_{reject}, \sqcup, L)
q_2	(q_{accept}, \sqcup, L)	(q_{reject}, \sqcup, L)	(q_{reject}, \sqcup, L)
q_3	(q_{reject}, \sqcup, L)	(q_{reject}, \sqcup, L)	(q_{reject}, \sqcup, L)

The program works as follows. For a state $q_i \in \{q_0, q_1, q_2, q_3\}$ specified in the LHS column and a given alphabet symbol $s_j \in \{0, 1, \sqcup\}$ specified in the top row, the box that corresponds to row q_i and column s_j and contains three symbols: the first symbol is the new state of M , the second symbol is the new alphabet symbol that will be written in the current cell (substituting symbol s_i) and the third symbol specifies the motion direction of the read-write head. So, row q_i and column s_j are the current configuration of M and the symbols contained in the box corresponding to row q_i and column s_j are the next configuration of M .

For example, let $X = 10100$ be an input binary string (we shall read the input string from left to right). The initial state of M is q_0 and M 's tape reads as $\sqcup \mathbf{1} 0 1 0 0 \sqcup$ where our first input symbol (in bold face) is the leftmost $\mathbf{1}$. So, the initial configuration of M is $q_0, \mathbf{1}$.

The transition function specifies that for a state q_0 and input symbol 1 , M must take q_0 as new state, write the value 1 in the cell where its read-write head is now located ($\mathbf{1}$) and take its read-write head one cell forward, i.e. to the right. So, M is now in the configuration $q_0, \mathbf{0}$ and its tape reads as

$\sqcup \mathbf{1} \mathbf{0} 1 0 0 \sqcup$.

The full run of this program is given in the following sequence

- Step 1: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 2: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 3: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 4: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 5: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 6: $q_0, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_1, \sqcup \mathbf{1} 0 1 0 0 \sqcup$
- Step 7: $q_1, \sqcup \mathbf{1} 0 1 0 0 \sqcup \rightarrow q_2, \sqcup \mathbf{1} 0 1 0 \sqcup \sqcup$
- Step 8: $q_2, \sqcup \mathbf{1} 0 1 0 \sqcup \sqcup \rightarrow q_y, \sqcup \mathbf{1} 0 1 \sqcup \sqcup \sqcup$

We have said that DTMs can do everything a real computer can do, and real computers compute values of functions. Transducer Machines, a DTM variant, are capable of those computations.

Definition 5.15. Transducer Machines. A transducer machine T is used to compute functions. T has a string $w \in \Sigma^*$ as input and produces another string y as output. Output y is stored in a special tape called the output tape. Given a function f , a transducer T computes f if the computation $T(w)$ reaches a final state containing $f(w)$ as output whenever $f(w)$ is defined (if f is not defined, T never reaches a final state). A function f is computable if a Turing Machine T exists capable of computing it.

5.5.5 Algorithmic Complexity for DTMs

A DTM can be used to find a solution to a problem, so how efficiently can such a solution be found? As stated previously, we shall be interested in finding the fastest algorithms. Let us now introduce a few concepts needed to quantify the efficiency of an algorithm.

The time complexity of an algorithm A expresses its time requirements by giving, for each input length, the largest amount of time needed by A to solve a problem instance of that size.

Definition 5.16. Time Complexity Function for a DTM. Let M be a DTM. We define $f : \mathbb{N} \rightarrow \mathbb{N}$ as the time complexity function of M , where $f(n)$ is the maximum number of steps that M uses on any input of length n .

Definition 5.17. Time Complexity Class for DTMs. Let $t : \mathbb{N} \rightarrow \mathbb{R}^+$ be a function. We define the time complexity class $\mathbf{TIME}(t(n))$, as the collection of all languages that are decidable by an $O(t(n))$ time DTM.

Definition 5.18. Class P. The class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine is denoted by \mathbf{P} and is defined as

$$\mathbf{P} = \bigcup_k \mathbf{TIME}(n^k)$$

The language of Ex. (5.1) is in \mathbf{P} .

A *polynomial time* or *tractable algorithm* is defined to be one whose time complexity function is $O(p(n))$ for some polynomial function p , where n is used to denote the input length. Any algorithm whose time complexity function cannot be so bounded is called an *exponential time* or *intractable algorithm*. Tractable algorithms are considered as acceptable, as a sign that a satisfactory solution for a problem has been found. Finding a tractable algorithm is usually the result of obtaining a deep mathematical insight into a problem. In contrast, intractable algorithms are usually solutions obtained by exhaustion, the so called brute-force method, and are not considered satisfactory solutions.

For example, no tractable algorithm for the Travelling Salesman Problem (5.3) is known so far ([67] and [74]). All solutions proposed so far are based on enumerating all possible solutions. Why is the Travelling Salesman problem intractable? Nobody knows for sure. It could be either that we need a deeper knowledge of the characteristics of this problem or, simply, that the Travelling Salesman problem is inherently intractable. However, no proof for neither of these two alternatives has been supplied so far.

DTMs are powerful machines. However, there are many decidable problems that require an unreasonable amount of resources from a DTM to be solved. In the following lines we introduce another model of computation used to tackle some of those problems.

We shall study two of those problems in detail in the last part of this chapter, but before doing so we must introduce some formal definitions in order to have the tools required to define and attack those problems.

5.5.6 Nondeterministic Turing Machines

The definition of a Nondeterministic Turing Machine (NTM) is similar to that of an NFA. The computation of an NTM is a tree whose branches correspond to different possibilities for the machine (see Fig. (3)). If some branch of the computation leads to the accept state q_{accept} then the machine accepts its input.

Definition 5.19. Nondeterministic Turing Machine. *A Nondeterministic Turing Machine is a 7-tuple $M_N = (Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets, $\mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the power set of $Q \times \Gamma \times \{L, R\}$, and*

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol \sqcup ,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$,
4. $\Delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the transition relation,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$

Notice that Δ is **not a function anymore but a relation**, reflecting the fact that an NTM does not have a single, uniquely defined next action but a choice between several next actions. In other words, for each state and symbol combination, there may be more than one appropriate next step, or none at all.

Definition 5.20. Computation with an NTM. *An NTM M_N accepts input $w \in \Sigma^*$ if at least one branch of its computation tree is a sequence of configurations C_1, C_2, \dots, C_k such that*

1. C_1 is the start configuration of M_N on input w ,
2. each C_i yields C_{i+1} , and
3. C_k is an accepting configuration.

M_N **accepts language** A if $A = \{w \mid M_N \text{ accepts } w\}$, i.e. A is the set of all strings accepted by M_N .

NTMs are powerful because of the asymmetrical input-output relation found in the way these machines compute. In order to have an NTM M_N accept one string w it suffices to find just one branch b in the computation tree that accepts w .

It can be shown that an NTM can be simulated by a DTM, i.e. that every NTM has an equivalent DTM ([68] and next section). However, simulating an NTM by a DTM may be at the cost of an exponential loss of efficiency [67]. Whether this loss is inherent to this ‘translation’ between models or is just a consequence of our limited understanding of nondeterminism is the famous $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem [67].

5.5.7 Algorithmic Complexity for NTMs

We start by offering the definitions of time complexity function and time complexity class for NTMs.

Definition 5.21. Time Complexity Function for an NTM. *Let M_N be an NTM. We define $g : \mathbb{N} \rightarrow \mathbb{N}$ as the time complexity function of M_N , where $g(n)$ is the maximum number of steps that M_N uses on any branch of its computation on any input length n .*

Definition 5.22. Time Complexity Class for NTMs. Let $t : \mathbb{N} \rightarrow \mathbb{R}^+$ be a function. We define the time complexity class $\mathbf{NTIME}(t(n))$, as the collection of all languages that are decidable by an $O(t(n))$ time nondeterministic Turing machine.

For an NTM to accept string w it is enough to find just one branch b in its computation tree that accepts w . However, a practical problem with this definition is to find b as an NTM can have an infinite (or exponentially big) number of different branches. Therefore, a more operational method for doing nondeterministic computation is needed. An important discovery in the theory of computation is the fact that the complexities of many problems are linked by means of a concept called verifiability.

For example, let us suppose we have a proposed solution for the Travelling Salesman problem (5.5) and another solution for the SAT problem (5.4). In both cases, we could easily check whether each proposal is indeed a solution, all we need is a DTM that *verifies* whether proposed solutions are right or not. Let us formalize these concepts.

Definition 5.23. Verifier. A verifier for a language A is an algorithm V , where

$$A = \{w \mid V \text{ accepts } (w, c) \text{ for some string } c\}$$

We measure the time of a verifier only in terms of the length of w , so a polynomial time verifier runs in polynomial time in the length of w . A language A is polynomially verifiable if it has a polynomial time verifier. The string c , a certificate, is additional information needed by the verifier. For example, in the case of problem (5.5), c is the set of cities and distances, along with the bound B . In the case of the SAT problem (5.4), c is the actual clause collection to be tested.

Note that a fundamental difference between an NTM (Def. (5.19)) and a verifier is that an NTM *finds* solutions, while a verifier only checks whether a proposal is a solution or not.

We now proceed to define a most important class of languages in Computer Science:

Definition 5.24. Class NP. The class of languages that have polynomial time verifiers is known as **NP**.

What is the relation between the abstract model of an NTM and the concepts of verifiers and NP languages class? The answer is given in Theorem (1) and its proof can be found in [68].

Theorem 1. A language is in **NP** if and only if it is decided by a nondeterministic polynomial time Turing machine.

5.5.8 $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ and NP-complete problems

The problem $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ is a fundamental topic in the theory of computation. It is known that $\mathbf{P} \subset \mathbf{NP}$ as any polynomial language can be checked with a polynomial verifier. Also, it can be proved [66] that

Theorem 2. If a problem $\zeta \in \mathbf{NP}$ then \exists a polynomial p such that ζ can be solved by a deterministic algorithm having time complexity $O(2^{p(n)})$.

Due to theorem (2) there is a widespread belief that $\mathbf{P} \neq \mathbf{NP}$ although no proof has been delivered and therefore $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ remains an open problem.

There is a particular set of problems in \mathbf{NP} that plays a key role in the theory of computation: **NP-complete** problems. In order to characterise this important set of problems we shall introduce the notion of polynomial transformations.

Definition 5.25. Polynomial Transformation. A polynomial transformation from a language $L_1 \subset \Sigma_1^*$ to a language $L_2 \subset \Sigma_2^*$, denoted by $L_1 \propto L_2$, is a function f such that

1. There is a polynomial time DTM that computes f .
2. $\forall x \in \Sigma_1^*, x \in L_1 \Leftrightarrow f(x) \in L_2$

Definition 5.26. NP-Complete Languages and Problems. A language L is **NP-complete** if $L \in \mathbf{NP}$ and, for all other languages $L_i \in \mathbf{NP}$ we find that $L_i \propto L$.

Due to our capacity to go from problem instances to languages by means of encoding schemes, we can also say that a decision problem ζ is **NP-complete** if $\zeta \in \mathbf{NP}$ and, for all other decision problems $\zeta_i \in \mathbf{NP}$ we find that $\zeta_i \propto \zeta$.

There is a plethora of **NP-complete** problems. The first NP-complete problem (chronologically speaking) was found by Stephen Cook ([75]) and it is stated in the following theorem (its proof can be found in [75] and [66]).

Theorem 3. NP-Completeness of SAT problem. SAT problem is NP-complete.

Therefore, studying the properties of SAT is an important and active area of research, not only because a polynomial-time solution to SAT would imply $\mathbf{P} = \mathbf{NP}$, but also because SAT is used to model problems and procedures in several areas of applied computer science and engineering like Artificial Intelligence (e.g. [76]) and hardware verification (e.g. [77] and [78]), using the following approach ([79]):

1. Represent the problem in propositional logic
2. Identify the proposition to be decided by satisfiability
3. Solve the SAT problem
4. Interpret the result in the original domain

Surveys of algorithms for solving several variations and instances of SAT can be found in [76], [80] and [79]. Also, good introductions to the vast field of computational complexity can be found in [81], [82], [74], [83], and [68].

6 Physics and the Theory of Computation

Considerations about the physical properties of systems used to do computation and/or transmission of information have been studied for several decades. Consequently, physics and computer science have cross-fertilised each other for long time. As early as in the 1940s, in the beginning of the digital computer era, scientists wondered about the existence and quantification of the minimum amount of energy required to perform a computation. J. von Neumann, in a set of lectures delivered in 1949 [?], showed that “a minimum amount of energy required per elementary decision of a two-way alternative and the elementary transmittal of one unit of information” was close to kT , where k is Boltzmann’s constant and T is the temperature of the system. Later on, R. Landauer studied the relationship between energy consumption and reversible computation (a computational step is reversible iff given the output of that step, its input is uniquely determined⁵.) Among those results published by Landauer in [?] we have the following principle.

Landauer’s principle. Suppose a computer erases a single bit of information. The amount of energy dissipated into the environment is at least $kT \ln 2$, where k is Boltzmann’s constant, and T is the temperature of the environment of the computer.

Landauer’s principle became a big motivation to do research in reversible computation. Among those works about reversible models of computation we find [84], [86] and [85].

Since evolution in quantum mechanics is reversible due to the use of unitary operators, the next step in the cross-fertilisation between computer science and physics was to link quantum mechanics and computer science. Benioff introduced the notion of Quantum Turing Machines and proposed a quantum mechanical model for the simulation of a classical computer ([87], [90], [89], [88] and chapter 6 of [93]). Additionally, R. Feynman, in his traditional and celebrated style, lectured at MIT in 1981 [92] about the fundamental capabilities and limitations of classical computers to simulate quantum systems. A gentle and concise introduction to this blend of physics, computer science and information theory, as well as Feynman’s main ideas behind physics and computation can be found in [93].

In 1985 D. Deutsch made two key contributions in [3]: a design of a *Universal Quantum Turing Machine*, and a physics-oriented version of the Church-Turing thesis which he called ‘Church-Turing principle’:

The Church-Turing principle [3]. Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.

In Deutsch’s words, the rationale behind the Church-Turing principle was “to reinterpret Turing’s ‘functions which would be naturally regarded as computable’ as the functions which may in principle be computed by a real physical system. For it would surely be hard to regard a function ‘naturally’ as computable if it could not be computed in Nature, and conversely”. The Universal Quantum Turing machine proposed in [3] was further developed and improved by Yao [95] and Bernstein and Vazirani [96].

⁵For example, the logical operation **OR** is *not* reversible, while the operation **NOT** is indeed reversible.

We now define a Probabilistic Turing Machine and a Quantum Turing Machine.

Definition 6.1. [8] Probabilistic Turing Machine. A Probabilistic Turing Machine (PTM) is a Nondeterministic Turing Machine which randomly chooses between the available transitions at each point according to a probability distribution. Thus, a PTM $M_N = (Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, is a 7-tuple where Q, Σ, Γ are all finite sets, $\mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the power set of $Q \times \Gamma \times \{L, R\}$, and

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol \sqcup ,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$,
4. $q_0 \in Q$ is the start state,
5. $q_{\text{accept}} \in Q$ is the accept state, and
6. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$
7. The transition relation is given by $\Delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\} \times [0, 1])$, so that for a given configuration C_0 , each of its successor configurations C_1, C_2, \dots, C_n is assigned a probability p_1, p_2, \dots, p_n , where n is the cardinality of $\mathcal{P}(Q \times \Gamma \times \{L, R\} \times [0, 1])$ and $\sum_{i=1}^n p_i = 1$.

Definition 6.2. [8] Quantum Turing Machine. A Quantum Turing Machine is defined analogously to a PTM but with a different transition relation. The transition relation includes the use of complex numbers which are the corresponding amplitudes of quantum states used for computation. A QTM is a 7-tuple $M_N = (Q, \Sigma, \Gamma, \Delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets, $\mathcal{P}(Q \times \Gamma \times \{L, R\})$ is the power set of $Q \times \Gamma \times \{L, R\}$, and

1. Q is the set of states
2. Σ is the input alphabet not containing the blank symbol \sqcup ,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$,
4. $q_0 \in Q$ is the start state,
5. $q_{\text{accept}} \in Q$ is the accept state, and
6. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{accept}} \neq q_{\text{reject}}$
7. The transition relation is given by $\Delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\} \times \mathbb{C}_{[0,1]})$, where $\mathbb{C}_{[0,1]} = \{z \in \mathbb{C} \mid |z|^2 \leq 1\}$. So, for a given configuration C_0 , each of its successor configurations C_1, C_2, \dots, C_n is assigned an amplitude z_1, z_2, \dots, z_n , where n is the cardinality of $\mathcal{P}(Q \times \Gamma \times \{L, R\} \times \mathbb{C}_{[0,1]})$ and $\sum_{i=1}^n |z_i|^2 = 1$.

Quantum computation can be regarded as the study and development of methods that, by using quantum mechanical properties, solve problems in finite time (from a different computational point of view, quantum computation is a sub-field of *unconventional models of computation* [94]). Quantum information can be defined as the field devoted to understanding how information is represented and communicated using quantum states. Due to the advances made over the last few years, both disciplines are now huge areas of research where diverse interests of several scientific communities can be found. Quantum walks is one of those interests, mainly contained in the sub-field of quantum algorithms.

7 Algoritmos cuánticos

Con el objetivo de adquirir práctica en el uso de símbolos y operaciones propios del cómputo cuántico, los trabajos de esta sección se harán en el pizarrón. Los temas que se estudiarán son:

7.1 Una mala noticia: the no cloning theorem

7.2 Si los qubits no pueden ser copiados, ¿cómo calcular?

7.3 Circuitos cuánticos

7.3.1 Compuertas controladas

7.3.2 Circuito cuántico para la generación de estados de Bell

7.4 Paralelismo cuántico

7.5 El algoritmo de Deutsch

References

- [1] Richard P. Feynman. *Simulating Physics with Computers*. International Journal of Theoretical Physics 21(6/7), pp. 467-488, 1982.
- [2] Richard P. Feynman. *Feynman Lectures on Computation*. Penguin Books, 1999.
- [3] David Deutsch. *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*. En Proceedings of the Royal Society of London, series A, Mathematical and Physical Sciences, Volume 400, Issue 1818, pp. 97–117, 1985.
- [4] Peter Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Algorithms on a Quantum Computer*. En Proceedings of the 35th Annual Symposium on Foundations of Computer Science, pp. 124–134, 1994. IEEE Computer Society Press.
- [5] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. En Proceedings of the 28th annual ACM symposium on the Theory of Computing, pp. 212–219, 1996.
- [6] Dirk Bouwmeester, Artur Ekert y Anton Zeilinger (Comps.). *The Physics of Quantum Information*. Springer, 2001.
- [7] Michael A. Nielsen e Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [8] Josef Gruska. *Quantum Computing*. McGraw-Hill, 1999.
- [9] Tom Apostol. *Calculus vol. II*. Editorial Reverté, 1980.
- [10] K.F. Riley, M.P. Hobson y S.J. Bence. *Mathematical Methods for Physics and Engineering*. Cambridge University Press, 1998.
- [11] H. Ted Davis y Kendall T. Thomson. *Linear Algebra and Linear Operators in Engineering*. Academic Press, 2000.
- [12] *Head and Tails. Quantum Computers are another step closer*. Technology Quarterly, The Economist, 2 de enero de 2003.
- [13] *Dream code. Programming languages for quantum computers are now being written*. Technology Quarterly, The Economist, 3 de abril de 2003.
- [14] *Quantum computing. Computing is driving the philosophical understanding of quantum theory*. Technology Quarterly, The Economist, 1 de abril de 2004.
- [15] Paul A. Benioff, *Space searches with a quantum robot*. En *Quantum Computation and Quantum Information: A millenium volume*. S. Lomonaco y H.E. Brandt (Comps.), AMS Contemporary Mathematics (305), 2002.
- [16] Carlo Trugenberger. *Probabilistic Quantum Memories*. Physical Review Letters, 87:067901, 2001.

- [17] Carlo Trugenberger. *Phase Transitions in Quantum Pattern Recognition*. Physical Review Letters, 89:277903, 2002.
- [18] Carlo Trugenberger. *Quantum Pattern Recognition*. Quantum Information Processing, 1(6), pp. 471-493, 2002.
- [19] M. Cristina Diamantini y Carlo Trugenberger. *Quantum Pattern Retrieval by Qubit Networks with Hebb Interactions*. quant-ph/0609117
- [20] Salvador E. Venegas-Andraca y Sougato Bose. *Quantum Computation and Image Processing: New Trends in Artificial Intelligence*. En Proceedings of the International Conference on Artificial Intelligence IJCAI-03, pp. 1563-1564, 2003.
- [21] Salvador E. Venegas-Andraca y Sougato Bose. *Storing, processing and retrieving an image using Quantum Mechanics*. En Proceedings of the SPIE Conference Quantum Information and Computation, pp. 137-147, 2003.
- [22] Salvador E. Venegas-Andraca y Jonathan L. Ball. *Storing Images in Entangled Systems*. Enviado a la revista Journal of Electronic Imaging, SPIE, 2006.
- [23] Sanjay Gupta. *Quantum Neural Networks*. Journal of Computer and System Sciences, 63, pp. 355-383, 2001.
- [24] Chun-Yang Zhang, Hsin-Chih Yeh, Marcos T. Kuroki y Tza-Huei Wang. *Single Quantum-Dot-Based DNA nanosensor*. Nature materials (4) pp. 826-831, 2005.
- [25] Yaakov S. Weinstein, C. Stephen Hellberg y Jeremy Levy. *Quantum-Dot Cluster-State Computing with Encoded Qubits*. Physical Review A 72(2): 020304, 2004.
- [26] Lloyd C.L. Hollenberg. *Fast Quantum Search Algorithm in Protein Sequence Comparison-Quantum Biocomputing*. Physical Review E 62, pp. 7532-7535 (2000).
- [27] *Uncrackable beams of light. Quantum Cryptography is finally going commercial*. Technology Quarterly, The Economist, 4 de septiembre de 2003.
- [28] <http://www.quantum.univie.ac.at/zeilinger/>
- [29] <http://ultrafast.physics.ox.ac.uk/>
- [30] <http://www.magiqtech.com/>
- [31] <http://www.qubit.org>
- [32] <http://cam.qubit.org>
- [33] <http://aspuru.chem.harvard.edu/>
- [34] <http://www.quantum.univie.ac.at/>

- [35] <http://www.iqi.caltech.edu/>
- [36] <http://www-math.mit.edu/~shor/>
- [37] <http://www.bell-labs.com/org/physicalsciences/>
- [38] <http://www.hpl.hp.com/research/qsr/>
- [39] <http://www.research.ibm.com/quantuminfo/>
- [40] <http://jp.fujitsu.com/group/labs/en/business/activities/activities-3/> (apartado Nanotecnología).
- [41] *Gana físico español Asturias de ciencia*. Periódico Reforma, 24 de mayo de 2006.
- [42] *El físico español Cirac Sastrain, Premio Príncipe de Asturias de investigación*. Periódico El País, 24 de mayo de 2006.
- [43] Oscar Rosas-Ortiz. *Manipulando el mundo cuántico: ingeniería cuántica*. Avance y Perspectiva (CINVESTAV), octubre-diciembre 2004.
- [44] Isaac Hernández Calderón. *Los semiconductores: de los transistores a las nanoestructuras y la computación cuántica*. Avance y Perspectiva (CINVESTAV), abril-junio 2005.
- [45] Carlos Fuentes. *Tecnología mexicana: noticias desde Londres*. Periódico Reforma, 27 de octubre de 2003.
- [46] José Galán. *Explora mexicano los alcances de la computación cuántica*. Entrevista con Salvador Venegas Andraca. Periódico La Jornada, 20 de mayo de 2005.
- [47] Salvador Elías Venegas Andraca. *Un vistazo a la tecnología del futuro I: ¿qué es la computación cuántica?* Periódico Milenio, 3 de enero de 2006.
- [48] Salvador Elías Venegas Andraca. *Un vistazo a la tecnología del futuro II: caminatas cuánticas* Periódico Milenio, 10 de enero de 2006.
- [49] Ariadne Gallardo. *La mecánica cuántica al servicio de la computación. Entrevista con Salvador Venegas Andraca*. Periódico en línea Casanchi. Enero de 2005.
- [50] Shahan Hacyan. *Teleportación cuántica*. Periódico Reforma, 1 de julio de 2004.
- [51] Shahan Hacyan. *Criptogramas cuánticos*. Periódico Reforma, 9 de septiembre de 2004.
- [52] *Teletransportan fotones*. Periódico Reforma, 19 de agosto de 2004.
- [53] Olivia Aguayo. *Teletransportación: Mito o realidad?* Periódico Reforma, 20 de febrero de 2006.
- [54] *Cómo construir una computadora cuántica*. Periódico La Jornada (The Economist Intelligence Unit), 18 de mayo de 2006.

- [55] *Computation Summer School in Yucatán*. Mexico Update, revista de comunicación oficial de la embajada de México en el Reino Unido. octubre de 2004.
- [56] Resumen de actividades de la primer escuela mexicana de verano en computación cuántica, <http://www.cem.itesm.mx/dia/escuelaverano> y escoger la opción “Escuela de verano 2004”.
- [57] C. Cohen-Tannoudji and B. Diu and F. Laloe. *Quantum Mechanics, Vols. 1 & 2*. Wiley-Interscience, 1977.
- [58] R. Coleman. *Stochastic Processes*. George Allen & Unwin, Ltd., 1974.
- [59] R.P. Feynman and R.B. Leighton and M. Sands. *The Feynman Lectures on Physics, vol. III*. Addison-Wesley Publishing Co., 1965.
- [60] A. Einstein. *Ideas and Opinions*. Wing Books, 1954.
- [61] W. Heisenberg. *Physics and Philosophy*. Penguin Group, 1962.
- [62] D. Preston. *Before the Fall-out: From Marie Curie to Hiroshima*. Doubleday, 2005.
- [63] P.A.M. Dirac. *The Principles of Quantum Mechanics*. Oxford University Press, 1930.
- [64] E. Rieffel y W. Polak. *An introduction to quantum computing for non-physicists*. ACM Computing Surveys, vol. 32(3)”, pp. 300–335, 2000.
- [65] B.J. Copeland. *The essential Turing*. Oxford University Press, 2004.
- [66] M.R. Garey y D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., NY, 1979.
- [67] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley Publishing Co., 1995.
- [68] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Co., 2005.
- [69] Alan M. Turing. *On Computable Numbers, with an application to the Entscheidungs problem*. Proceedings of the London Mathematical Society, vol. 42, pp. 230-265, 1936-37.
- [70] . Richard Zach. *Hilbert’s Program*. The Stanford Encyclopedia of Philosophy, Edward N. Zalta, <http://plato.stanford.edu/archives/fall2003/entries/hilbert-program/>, Fall 2003.
- [71] David Hilbert. *Mathematische Probleme*. Archiv der Mathematik und Physik 1 (1901) 4463, 213237. English translation by M.W. Newson in Bulletin of the American Mathematical Society 8 (1902), 437-479, url = ”<http://aleph0.clarku.edu/~djoyce/hilbert/problems.html>
- [72] David Hilbert. *Neubegründung der Mathematik: Erste Mitteilung*. Series of talks given at the University of Hamburg. English translations given by Mancosu, Paolo (ed.), 1998a, From Brouwer to Hilbert. The Debate on the Foundations of Mathematics in the 1920s, Oxford. Oxford University Press, 1922.

- [73] D. Hilbert y W. Ackerman. *Grundzuge der theoretischen Logik*. Springer-Verlag, 1928. English translation of the second edition *Principles of Mathematical Logic*, by L. M. Hammond et al., Chelsea, New York, 1950.
- [74] Stephan Mertens. *Computational Complexity for Physicists*. Computing in Science and Engineering, IEEE, pp. 31-47, May-June 2002.
- [75] Stephen A. Cook. *The Complexity of Theorem-Proving Procedures*. Proc. 3rd Ann. ACM Symposium on the Theory of Computing, pp. 151-158, 1971.
- [76] I. Gent y T. Walsh. *The search for Satisfaction*. Internal report, department of computer science, University of Strathclyde, 1999.
- [77] P. Bjesse, T. Leonard y A. Mokkedem. *Finding bugs in an alpha microprocessor using satisfiability solvers*. Proc. 13th International Conference on Computer Aided Verification, 2001.
- [78] M.N. Velev y R.E. Bryant. *Effective use of Boolean satisfiability procedures in the formal verification of superscalar and vliw microprocessors*. Proc. 38th Design Automation Conference (DAC '01), pp. 226-231, 2001.
- [79] H. Rantanen. *Analyzing the random-walk algorithm for SAT*. Master's thesis, Helsinki University of Technology, 2004.
- [80] H. Kautz y B. Selman. *The state of SAT*. Preliminary version in Proc. of CP-2003. To appear in Discrete and Applied Mathematics, 2005.
- [81] Stephen A. Cook. *An overview of computational complexity*. Turing award lecture 1983, Association for Computing Machinery, 1983.
- [82] L. Fortnow y S. Homer. *A Short History of Computational Complexity*. D. van Dalen, J. Dawson, y A. Kanamori (Eds.), The History of Mathematical Logic. North-Holland, Amsterdam, 2003.
- [83] C.H. Papadimitriou. *On selecting a satisfying truth assignment*. Proceedings 32nd IEEE Symposium on the Foundations of Computer Science, pp. 163 - 169, 1991.
- [84] C.H. Bennett. *Logical Reversibility of Computation*. IBM Journal of Research and Development, vol. 17, pp. 525-532, 1973.
- [85] Y. Lecerf. *Logique Mathématique : Machines de Turing réversibles*. Comptes rendus des séances de l'académie des sciences, vol. 257, pp. 2597-2600, 1963.
- [86] E. Fredkin y T. Toffoli. *Conservative Logic*. International Journal of Theoretical Physics, vol. 21, pp. 219-253, 1982.
- [87] P.A. Benioff. *The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing Machines*. Journal of Statistical Physics, vol. 22(5), p. 563, 1980.

- [88] P.A. Benioff. *Quantum Mechanical models of Turing Machines that Dissipate no energy*. Phys. Rev. Lett., vol. 48, pp. 1581–1585, 1982.
- [89] P.A. Benioff. *Quantum mechanical Hamiltonian models of Turing machines*. Journal of Statistical Physics, vol. 3(29), pp. 515–546, 1982.
- [90] P.A. Benioff. *Quantum Mechanical Hamiltonian Models of Discrete Processes That Erase Their Own Histories: Application to Turing Machines*. International Journal of Theoretical Physics, vol. 21, pp. 177-201, 1982.
- [91] P.A. Benioff. *Space searches with a quantum robot*. In Quantum Computation and Quantum Information: A millenium volume. S. Lomonaco and H.E. Brandt (Eds.), AMS Contemporary Mathematics (305), 2002.
- [92] R.P. Feynman. *Simulating Physics with Computers*. International Journal of Theoretical Physics, vol. 21(6/7), pp. 467-488, 1982.
- [93] R. P. Feynman. *Feynman Lectures on Computation*. Penguin Books, 1999.
- [94] C.S. Calude, J. Casti y M.J. Dineen (Eds.). *Unconventional Models of Computation*. Springer-Verlag, 1998.
- [95] A. C. Yao. *Quantum Circuit Complexity*. Proceedings of Thirty-fourth IEEE Symposium on Foundations of Computer Science, pp. 352–361, 1993.
- [96] E. Bernstein y U. Vazirani. *Quantum complexity theory*. SIAM Journal of Computing, vol. 5(26), pp. 1411–1473, 1997.