Teoría de la Complejidad en Computación Cuántica

Jesús Camacho Moro

Teoría de la Complejidad en Computación Cuántica

Jesús Camacho Moro

Memoria presentada como parte de los requisitos para la obtención del título de Máster Universitario en Matemáticas por la Universidad de Sevilla.

Tutorizada por

Prof. José María Tornero Sánchez

Índice general

Abstract

During the first half of the 20th century, the need of faster calculations brought the first computers to our world. Those machines were designed following the same abstract blueprint: a structure called Turing Machine.

Since then, the creation and improvement of algorithms solving well known mathematical problems have continued. Following the same path, scientists from different areas studied the complexity theory around classical computers.

In 1982 American physicist Richard Feynman stepped out of this paradigm when he realized that quantum systems could not be efficiently simulated using any of the previous computers.

British physicist David Deutsch introduced a new machine based on this idea in 1985, named quantum Turing machines, given the similarities with Turing machines and the fact that it took advantage of the principles of quantum mechanics.

After the formal definition, quantum complexity theory and algorithms appeared hand by hand.

In this memory we will take a look into the construction of an universal quantum Turing machine and its associated complexity theory, which is still an open and fruitful field of current research.

Next, we will present an algoritm stated by David Deutsch and Australian mathematician Richard Jozsa in 1992 as an example of the intrinsic power inside quantum Turing machines.

1 Máquinas de Turing cuánticas

A lo largo de la primera mitad del siglo XX el mundo vio avanzar a la teoría de la computabilidad a pasos agigantados. En 1933, Kurt Gödel y Jacques Herbrand definieron de manera formal la clase de funciones recursivas. Posteriormente, Alonzo Church creó las funciones λ-calculables, a priori sin relación con las recursivas. Al mismo tiempo, Alan Turing construye el modelo teórico de máquinas por el que es conocido y crea un nuevo grupo de funciones que son las que hoy conocemos por Turing computables.

Church y Turing probaron que estas tres clases de funciones coinciden en realidad. Además, afirmaron que dichas clases de funciones computables definidas formalmente coinciden con la noción intuitiva de funciones calculables de forma eficiente.

Al igual que la teoría de la computabilidad recae sobre la tesis Church-Turing, la teoría de complejidad computacional depende de una versión moderna de esta misma tesis. En ella se asegura que cualquier modelo de computación "razonable" puede ser simulado eficientemente en una máquina de Turing probabilística. Entendemos por razonable aquella máquina que puede ser físicamente realizable.

El problema que se plantea con este tipo de máquinas de Turing reside en el uso de modelos físicos clásicos para la realización de una misma. Es por ello que la llegada de grandes avances en el campo de la física durante el siglo XX, como el modelo de partículas cuántico, permitieron abordar el problema de forma distinta.

Este nuevo enfoque para la definición de nuevos modelos computacionales dirigido por resultados como el propuesto por Feynman [7] en 1982 tuvo gran impacto. En el artículo citado afirmaba que la simulación de un sistema físico cuántico en una máquina de Turing probabilística requería un coste exponencial. La posibilidad de definir un nuevo modelo que pudiera ser representado físicamente por algún tipo de máquina y cuyo poder computacional fuera mayor abría un nuevo mundo.

En este capítulo daremos una formalización del concepto de máquina de Turing cuántica. La construcción dada por Deustch [5] es interesante desde el punto de vista de la computabilidad, pero no de la complejidad computacional. Con pequeñas modificaciones sobre dicho modelo se consigue el que definiremos a continuación.

La ventaja del enfoque que vamos a dar es que podremos probar la existencia de una máquina de Turing cuántica universal que simule cualquier otra con un coste en tiempo acotado por un polinomio. Para este desarrollo hemos seguido, fundamentalmente, la referencia [4].

1.1 Definición

Antes de introducir las máquinas de Turing cuánticas es conveniente fijar conceptos básicos de la computación clásica y así poder compararlos con sus equivalentes cuánticos.

Definición 1.1. *Una máquina de Turing determinista (MT) consiste en un triplete* (Σ, Q, δ)*, donde* Σ *es un alfabeto finito con un símbolo especial para representar la casilla en blanco #,* Q *es un conjunto finito de estados con* q_0 *identificando el estado inicial y* $q_f \neq q_0$ *identificando el estado final, y* δ *es la función de transición determinista*

$$\delta : Q \times \Sigma \to \Sigma \times Q \times \{L, R\}.$$

Supondremos que la MT tiene una cinta infinita a ambos lados compuesta de celdas que indexamos con $\mathbf{Z}$ *y una cabeza lectora/escritora que se mueve a lo largo de la cinta.*

Existen muchas definiciones equivalentes de máquinas de Turing: por ejemplo, algunas admiten escribir y moverse al mismo tiempo, mientras que otras permiten un tercer tipo de movimiento consistente en no desplazarse. Elegimos esta descripción porque es fácilmente generalizable a máquinas de Turing cuánticas.

A continuación procedemos a definir las máquinas de Turing probabilísticas. La descripción que daremos está en consonancia con la vista de MT y con la que presentaremos posteriormente para máquinas de Turing cuánticas.

Definición 1.2. *Una máquina de Turing probabilística (MTP) consiste en un triplete* (Σ, Q, δ)*, donde* Σ *es un alfabeto finito con un símbolo especial para representar la casilla en blanco #,* Q *es un conjunto finito de estados con* q_0 *identificando el estado inicial y* $q_f \neq q_0$ *identificando el estado final, y* δ*, la función de transición*

$$\delta : Q \times \Sigma \to [0,1]^{\Sigma \times Q \times \{L,R\}}.$$

La MTP tiene una cinta infinita a ambos lados compuesta de celdas que indexamos con $\mathbf{Z}$ *y una cabeza lectora/escritora que se mueve a lo largo de la cinta.*

Observación 1.1. La función de transición δ puede considerarse como una matriz estocástica definida en el espacio de configuraciones. Sin embargo, no toda matriz estocástica define una MTP.

Observación 1.2. Mientras que la unidad básica de información en computación clásica es el bit, en el caso cuántico recibe el nombre de qubit. Un bit solo puede encontrarse en uno de sus dos posibles estados, generalmente denotados por $\{0, 1\}$. No ocurre lo mismo con un qubit, dado que no sabemos a priori en que estado se encuentra, si no con qué probabilidad se encuentra en un estado u otro. Con intención de reflejar estos patrones, elegimos el espacio de Hilbert $\mathbf{C}^2$ con el producto interno usual y construimos una base de qubits, que representaremos con la notación de Dirac:

1. La notación *ket* se utiliza con los vectores del espacio: $|\psi\rangle$.
2. La notación *bra* representa a los duales: $\langle\psi|$.
3. El producto interno de dos vectores se representa con un *braket*: $\langle\psi|\phi\rangle$.

En tales condiciones, la base canónica está formada por los siguientes vectores:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}.$$

En general, trabajaremos sobre el espacio de Hilbert $\mathbf{C}^{2^n}$ con la base canónica $\{|0\rangle, |1\rangle, \dots, |2^n - 1\rangle\}$. Dado un vector del espacio, llamaremos *amplitudes* a los módulos de sus coeficientes respecto a dicha base, que se identifican con la probabilidad de estar en ese estado preciso. Ver [10] para más detalles sobre esto.

Observación 1.3. Como ocurre en el caso de la MT probabilísticas, debemos limitar las amplitudes a números que aporten cierta eficiencia. Adleman, DeMarrais y Huang [1] mostraron que restringir las máquinas de Turing cuánticas a amplitudes racionales no reducen su poder computacional.

Definición 1.3. *Llamaremos* $\tilde{\mathbf{C}}$ *al conjunto de elementos* $\alpha \in \mathbf{C}$ *tales que hay una algoritmo determinista que computa las partes real e imaginaria de* α *en tiempo polinomial en* n *con un error de, a lo sumo,* 2^{-n}.

Definición 1.4. *Una máquina de Turing cuántica (MTC) consiste en un triplete* (Σ, Q, δ), *donde* Σ *es un alfabeto finito con un símbolo especial para representar la casilla*

en blanco #, Q es un conjunto finito de estados con q_0 identificando el estado inicial y $q_f \neq q_0$ identificando el estado final, y δ, la función de transición cuántica

$$\delta : Q \times \Sigma \to \tilde{\mathbf{C}}^{\Sigma \times Q \times \{L,R\}}.$$

La MTC tiene una cinta infinita a ambos lados compuesta de celdas que indexamos con $\mathbf{Z}$ y una cabeza lectora/escritora que se mueve a lo largo de la cinta.

Observación 1.4. Notaremos por $\delta(p, \sigma, \tau, q, d)$ a la amplitud en $\delta(p, \sigma)$ de $|\sigma\rangle|q\rangle|d\rangle$, para especificar la transición a la que está asociada.

Como ocurre en la versión clásica, nosotros consideramos MTC con una cabeza lectora/escritora que sólo se mueve a izquierda o derecha. En el caso de permitir un movimiento estático N, especificaremos que se trata de una MTC generalizada.

Asociado a la función de transición cuántica, existen una serie de conceptos que tienen las MTC como añadido a las MT clásicas. Dada una MTC M, se define el espacio producto de combinaciones lineales complejas finitas de configuraciones de M tomando la norma euclídea, denotado por $\mathbf{S}$. Llamaremos *superposición* a cada elemento $\phi \in \mathbf{S}$ de M.

Además, la MTC M define de forma inmediata un operador lineal $U_M : \mathbf{S} \to \mathbf{S}$, llamado *operador de evolución en el tiempo*. M se encuentra en una configuración c con estado p y leyendo el símbolo σ. Tras un paso, M estará en una superposición de configuraciones $\sum_i \alpha_i c_i$, en la que cada α_i no nulo corresponde con una transición $\delta(p, \sigma) = (\tau, q, d)$, y cada c_i es la nueva configuración que resultaría de aplicar dicha transición a c. Esta definición se puede extender por linealidad a todo $\mathbf{S}$ y así tener U_M bien definida.

Siguiendo el orden usual, una vez definidas las MTC pasamos a ver qué entenderemos por una configuración de parada.

Definición 1.5. *Una configuración final de una MTC M es cualquier configuración en el estado final q_f. Si al aplicar M con entrada x, la superposición tras T pasos contiene solo configuraciones finales, y en cualquier paso previo la superposición no contiene ninguna, diremos que M para en tiempo T con entrada x.*

Esta definición generaliza de forma natural lo que entendemos por configuración de parada en las MT clásicas. De igual forma, damos varios conceptos que usaremos a lo largo del texto y que son similares para ambos tipos de máquinas.

Definición 1.6. *Diremos que una MTC se comporta bien si para con cualquier entrada y lo hace con una superposición final donde cada configuración tiene la cabeza lectora/escritora en la misma celda. Si además esta celda es la inicial, diremos que la máquina es estacionaria.*
De la misma forma, diremos que una MT determinista es estacionaria si para en todas las entradas y lo hace con su cabezal en la celda inicial.

Definición 1.7. *Una MTC o una MT determinista $M = (\Sigma, Q, \delta)$ está en forma normal si*

$$\forall \sigma \in \Sigma \quad \delta(q_f, \sigma) = (\sigma, q_0, R) = |\sigma\rangle|q_0\rangle|R\rangle.$$

Definición 1.8. *Una MTC $M = (\Sigma, Q, \delta)$ es unidireccional si cada estado es accesible solo desde una dirección. Es decir, si $\delta(p_1, \sigma_1, \tau_1, q, d_1)$ y $\delta(p_2, \sigma_2, \tau_2, q, d_2)$ son no nulos, entonces $d_1 = d_2$.*

Definición 1.9. *Cuándo una MTC M en superposición $\psi = \sum_i \alpha_i c_i$ es observada, la configuración c_i aparece con probabilidad $|\alpha_i|$. Además, la superposición de M se actualiza a $\psi' = c_i$.*

Tanto en el modelo físico como en esta definición, al observar una superposición solo permanece un qubit, perdiendo así toda la información del resto.

Definición 1.10. *Para toda entrada x, una MTC M produce una muestra de una distribución de probabilidad en las configuraciones. Diremos que una máquina M' simula M con coste f y precisión ε si la distancia variacional entre las distribuciones obtenidas al observar M con entrada x tras T pasos y M' con entrada x tras $f(T)$ pasos, es menor o igual que ε.*

Observación 1.1. Recordamos que la distancia variacional entre dos medidas de probabilidad P, Q en un σ-álgebra $\mathcal{F}$ viene dada por la siguiente fórmula:

$$D(P, Q) = \sup_{A \in \mathcal{F}} |P(A) - Q(A)|$$

Por último, necesitamos dar una noción equivalente a las MT clásicas con k cintas. En el caso clásico sabemos que una MT de k cintas puede ser simulada por otra de tan solo una con un coste en tiempo constante. Esta idea es la que seguiremos para la construcción que daremos como generalización.

Definición 1.11. *Una MT con k tramos es una MT cuyo alfabeto Σ es de la forma $\Sigma_1 \times \cdots \times \Sigma_k$ con un símbolo especial # para cada Σ_i, constituyendo $(\#, \ldots, \#)$ como el símbolo en blanco de Σ.*
En tal caso, especificaremos la entrada dando la cadena para cada tramo.

1.2 Máquina universal

El propósito de esta sección es probar la existencia de una máquina de Turing cuántica universal. Es decir, existe una MTC que simula a cualquier otra con, a lo sumo, un coste en tiempo polinomial. El hecho de si estas máquinas se pueden o no construir físicamente es una tema que sigue en debate dentro de la comunidad científica y en el que no vamos a entrar.

Como veremos, las MTC se pueden caracterizar por ser reversibles y tener interferencia. Así pues, debemos simular una MTC que conserve dichas propiedades. Para ello, reduciremos significativamente el conjunto de MTC que simularemos (teniendo en cuenta que el subconjunto de MTC resultante sea representativo del global) y luego probaremos la existencia de una MTC que simule cualquier máquina de esta subclase.

Existen varias formas de enfocar la construcción de una MTC universal, por el simple hecho de que se requiere una sucesión de máquinas que puedan ser simuladas por otras más simples. En nuestro caso, vamos a seguir el guión aportado por Bernstein y Vazirani [4]. Otros autores como Adleman, DeMarrais y Huang [1] ó Solovay y Yao [13] probaron que no se pierde potencia computacional si nos restringimos a amplitudes racionales. De hecho, el espacio de amplitudes generado por el subconjunto $\{0, \pm 3/5, \pm 4/5, 1\}$ es suficiente para construir una MTC universal.

1.2.1 Máquinas bien formadas

Definición 1.12. *Diremos que una MTC M está bien formada si el operador de evolución en el tiempo, U_M, preserva la norma L_2.*

Esta condición es esencial, porque queremos asociar el cuadrado de las magnitudes de una configuración con su probabilidad en una observación, como hemos visto en 1.9.

Observación 1.5. Dada una matriz M con coeficientes complejos, se dice que es unitaria si $M^t \cdot M = M \cdot M^t = I$ donde M^t es la traspuesta conjugada de M. Entonces, M preserva la norma L_2 si y sólo si $M^t \cdot M = I$ y M es de dimensión finita. Es decir, si M es unitaria.

Puesto que las matrices que nosotros consideramos tienen dimensión infinita, este resultado no tendría por qué mantenerse. Sin embargo, el operador de evolución en

el tiempo de las MTC si cumple dicha caracterización.

Proposición 1.1. Una MTC está bien formada si y sólo si su operador de evolución en el tiempo es unitario.

Demostración. Para la implicación directa basta probar que el operador en el tiempo M es sobreyectivo. Por reducción al absurdo, supongamos que $M^t \cdot M = I$ y M no es sobreyectiva. Entonces, existe al menos una configuración c que no pertenece al rango del operador. De hecho, cualquier configuración en un entorno de c suficientemente pequeño también estará fuera del rango.

Ahora, seleccionamos n celdas contiguas de nuestra cinta y consideramos el conjunto S_n de configuraciones que tienen la cabeza lectora/escritora encima de una de estas n celdas y cuya cinta está en blanco fuera de las n celdas. Esto se traduce en que las columnas de M solo pueden tener entradas no nulas en las filas correspondientes a las n celdas y en las $2|Q||\Sigma|^n$ filas correspondientes a las configuraciones donde la cabeza lectora sale de las n celdas.

Sin embargo, al menos $(n-2)|\Sigma|^{n-3}$ de las configuraciones en S_n son localmente iguales a c, así que están también fuera del rango. Por lo tanto, para $n > 2|Q||\Sigma|^3 + 2$ no todas las columnas de S_n pueden ser mutuamente ortogonales, lo que nos lleva a contradicción con $M^t \cdot M = I$.

La otra implicación es directa. ∎

1.2.2 Reversibilidad

Definición 1.13. *Una MT reversible es una MT determinista para la que cada configuración tiene, a lo más, un predecesor.*

Esta definición se aleja del guión que veníamos siguiendo de Bennett [2], pero nos sirve para verlas como un caso especial de MTC. Si seguimos esta definición, tenemos que imponer que la función de transición δ sea total, es decir, que esté definida para todas las posibles entradas.

En el caso de MTC, además de revertir la computación, tenemos que decidir qué sucede con las amplitudes cuando revertimos la acción de δ. Sabemos por la proposición 1.1 que el control cuántico de estados finitos o control de estados finitos cuánticos δ conserva la norma si y sólo si $U_M U_M^t$ es la matriz identidad. Por lo que U_M^t revierte δ. Una forma similar de expresarlo es la dada por el siguiente lema:

Lema 1.1. Dada una MTC $\boldsymbol{M}$ con función de transición δ, $\boldsymbol{M}$ está bien formada si y sólo si la aplicación de configuraciones δ' que revierte la acción de δ y conjuga las amplitudes deshace la computación de δ.

Es claro que esta formalización generaliza el concepto clásico, y como apuntaba Deutsch [5], las MT reversibles son una subclase de MTC.

Este concepto de reversibilidad encaja muy bien con las MTC bien formadas. Ya conocemos una caracterización global de MTC bien formadas gracias a la proposición 1.1. Más adelante, exigiremos condiciones locales a las máquinas cuánticas y así podremos trabajar con un conjunto de máquinas más pequeño.

Teorema 1.1. *Cualquier MT reversible es también una MTC bien formada.*

Demostración. La función de transición δ de una MT determinista lleva el estado actual y el símbolo leído a un triplete. Podemos plantearlo como enviar la superposición unitaria de amplitud 1 para dicho triplete y 0 en el resto. Entonces, δ es también una función de transición y define una MTC.

La matriz de evolución en el tiempo correspondiente a dicha MTC tiene como entradas 0 o 1. Puesto que cada configuración tiene un solo predecesor en MT reversibles, esta matriz representa realmente una permutación. Por lo tanto, cualquier superposición de configuraciones $\sum_i \alpha_i |c_i\rangle$ es enviada por la matriz de evolución temporal a otra superposición $\sum_i \alpha_i |c_i'\rangle$. Es decir, preserva la norma. ∎

Teorema 1.2 (Sincronización). *Sea f una aplicación entre cadenas que puede ser computada en tiempo polinomial determinista y tal que la longitud de $f(x)$ depende solo de la longitud de x. Entonces existe una MT reversible en forma normal estacionaria que dada una entrada x, devuelve $f(x)$ en tiempo sólo dependiente de la longitud de x.*

Si f es una función de cadenas en cadenas tal que f y f^{-1} pueden ser computadas en tiempo polinomial determinista y tal que la longitud de $f(x)$ sólo depende de la longitud de x, entonces existe una MT reversible en forma normal, estacionaria que dada una entrada x, produce la salida $f(x)$ con un tiempo de ejecución que depende únicamente de la longitud de x.

Demostración. La prueba se puede encontrar en el texto [4]. ∎

1.2.3 Herramientas

Introducimos varios lemas técnicos de gran utilidad. Con ellos podremos reordenar las cintas de forma adecuada para poder, entre otras cosas, encadenar la acción de varias máquinas.

Lema 1.2. Dada una MTC $M = (\Sigma, Q, \delta)$ y un conjunto cualquiera Σ', existe una MT de dos tramos $M' = (\Sigma \times \Sigma', Q, \sigma')$ tal que su primer tramo se comporta como M mientras que deja su segundo tramo sin modificar.

Lema 1.3. Dada una MTC $M = (\Sigma_1 \times \ldots \times \Sigma_k, Q, \delta)$ y una permutación $\pi : [1, k] \to [1, k]$, existe una MTC $M' = (\Sigma_{\pi(1)} \times \ldots \times \Sigma_{\pi(k)}, Q, \delta')$ tal que M' se comporta igual que M excepto que sus tramos se permutan según indica π.

Ambos resultados son directos, por lo que no daremos demostración.

Lema 1.4. Existe una MT M estacionaria, reversible en forma normal y una constante c con las siguientes propiedades: para cualquier entrada entera positiva k en notación binaria, M trabaja en tiempo $\mathcal{O}(k \log^c k)$ y para con su cinta sin modificar. Más aún, M tiene un estado especial q^* tal que al introducir k, M visita el estado q^* exactamente k veces y cada una de ellas con la cabeza lectora/escritora en la celda inicial.

Demostración. Usando el Teorema 1.2 de sincronización podemos construir una MT reversible $M_1 = (\Sigma, Q, \delta)$ estacionaria, en forma normal, con tres tramos y trabajando en tiempo polinomial en $\log k$. Esta máquina devuelve $(b', x + 1, k)$ al introducir (b, x, k), $b \in \{0, 1\}$, donde b' es el opuesto de b si $x = 0$ o $k - 1$ (no al mismo tiempo) y $b' = b$ en otro caso.

Si llamamos q_0, q_f a los estados inicial y final de M_1, construimos una máquina reversible M_2 como sigue.

Denotamos por q_a, q_z a los nuevos estados inicial y final a los que exigimos una serie de propiedades:

1. Comenzando en el estado q_a con un 0 en el primer tramo, M_2 mueve a izquierda y vuelve a derecha, cambiando el símbolo 0 por 1, además de entrar en el estado q_0.
2. Cuándo entramos en el estado q_f con un 0 en el primer tramo, M_2 mueve a izquierda y vuelve a derecha, además de entrar en el estado q_0.

3. Cuándo entramos en el estado q_f con un 1 en el primer tramo, M_2 mueve a izquierda y vuelve a derecha, cambiando el símbolo 1 por 0 y parando.

En estas condiciones, con entrada $(0, 0, k)$ M_2 avanza a izquierda y vuelve a derecha entrando en el estado q_0 y modificando la cinta en $(1, 0, k)$. Entonces la máquina M_1 cambia dicho contenido por $(0, 1, k)$ y para en estado q_f. Cuándo M_2 se encuentra en estado q_f con 0 en el primer tramo, entra al estado q_0 y repite el proceso, Continuando con este bucle, M_2 trabajará $k-1$ veces más hasta llegar a $(1, k, k)$. En dicho momento, M_2 cambia el contenido a $(0, k, k)$ y para. Es decir, para la entrada $(0, 0, k)$ la máquina M_2 visita el estado q_f exactamente k veces, cada una de ellas con el cabezal en la celda inicial. Nuestra máquina objetivo M será la concatenación de M_2 entre dos MT reversibles, obtenidas mediante el Teorema de sincronización, al transformar k en $(0, 0, k)$ y $(0, k, k)$ en k.

Para completar la prueba basta construir una M_2 reversible, en forma normal que satisfaga las tres propiedades anteriores. Asociamos a M_2 el mismo alfabeto de M_1 y añadimos los estados q_a, q_b, q_y, q_z. La función de transición de M_2 será la misma en los estados $Q - q_f$ y en el resto sólo dependerá del primer tramo de cinta. Describimos dicha función con la siguiente tabla:

	#	0	1
q_a		$(1, q_b, L)$	
q_b	$(\#, q_0, R)$		
q_f		$(0, q_b, L)$	$(0, q_y, L)$
q_y	$(\#, q_z, R)$		
q_z	$(\#, q_a, R)$	$(0, q_a, R)$	$(1, q_a, R)$

Las comprobaciones de que M_2 está efectivamente en forma normal, al igual que las propiedades exigidas, se dejan para el lector interesado.

1.2.4 Cambios de base

La herramienta de cambiar bases a lo largo de las computaciones en una MTC es muy útil para probar varios resultados centrales de la teoría de computación cuántica. Generalmente, trataremos de pasar a una base ortonormal para la función de transición y así simular una MTC general con una que sea unidireccional.

De hecho, también nos servirá para extender MTC parciales a MTC bien formadas. Esto nos permite seguir el guión comentado al principio de la sección.

Lema 1.5. Dada una MTC $M = (\Sigma, Q, \delta)$ y un conjunto de vectores $B \in \tilde{\mathbf{C}}^Q$ que forme una base ortonormal de $\mathbf{C}^Q$, existe una MTC $M' = (\Sigma, Q, \delta')$ que evoluciona igual que M bajo un cambio de base entre Q y B.

Demostración. Sea $M = (\Sigma, Q, \delta)$ una MTC y sea B una base ortonormal de $\tilde{\mathbf{C}}^Q$. Como B es base ortonormal, esta define una transformación unitaria entre el espacio de superposiciones de estados en Q y el espacio de superposiciones de estados en B. Es decir, para cada $p \in Q$ tenemos la aplicación

$$|p\rangle \to \sum_{v \in B} \langle p|v\rangle \cdot |v\rangle$$

De igual forma, tenemos una transformación unitaria entre los espacios de configuraciones. En este caso, una configuración con estado p se asocia con la configuración correspondiente cuyo estado v aparezca con amplitud $\langle p|v\rangle$.

Fijado un estado p y leyendo σ, la máquina evoluciona en un paso a la siguiente superposición:

$$\delta(p, \sigma) = \sum_{\tau,q,\sigma} \delta(p, \sigma, \tau, q, d)\ |\tau\rangle|q\rangle|d\rangle.$$

Con el cambio de bases, la superposición será:

$$\sum_{\tau,v,\sigma} \left(\sum_{q} \langle q|v\rangle\ \delta(p, \sigma, \tau, q, d) \right)\ |\tau\rangle|v\rangle|d\rangle.$$

Puesto que el símbolo de estado $|v\rangle|\sigma\rangle$ de M' corresponde en M con la configuración

$$\sum_{p} \langle v|p\rangle\ |p\rangle|\sigma\rangle,$$

debemos tener en M'

$$\delta'(p, \sigma) = \sum_{p} \langle v|p\rangle \left(\sum_{\tau,v',\sigma} \left(\sum_{q} \langle q|v'\rangle\ \delta(p, \sigma, \tau, q, d) \right) \right)\ |\tau\rangle|v'\rangle|d\rangle.$$

Por lo tanto, M' se comportará como M bajo el cambio de base si definimos δ' como hemos visto previamente.

Dado que los vectores de B están contenidos en $\tilde{\mathbf{C}}^Q$, cada amplitud de δ' estará en $\tilde{\mathbf{C}}$. Además, el operador de evolución en el tiempo de M' que hemos definido preserva la norma L_2 por herencia de δ bajo un cambio de base ortonormal. Por tanto, δ' está bien definido. ∎

Como ya vimos previamente, las MT reversibles son una buena base de trabajo. Este tipo de máquinas se caracterizaban por ser MTC bien formadas (Teorema 1.1), por lo que es conveniente que estudiemos con mayor detenimiento esta propiedad.

Teorema 1.3. *Una MTC $M = (\Sigma, Q, \delta)$ está bien formada si y sólo si satisface lo siguiente:*

1. *El vector de amplitudes que dejan cualquier par de estado y símbolo, tiene norma constante igual a* 1*:*

$$\forall\, p, \sigma \in Q \times \Sigma \quad \sum_{\tau,q,d} |\delta(p, \sigma, \tau, q, d)|^2 = 1$$

2. *Las superposiciones con un símbolo de escritura, un estado de llegada y un movimiento son ortogonales para cualesquiera dos estados-símbolos distintos:*

$$\forall\, (p_1, \sigma_1) \neq (p_2, \sigma_2) \in Q \times \Sigma \quad \sum_{\tau,q,d} \delta(p_1, \sigma_1, \tau, q, d) \cdot \delta^*(p_2, \sigma_2, \tau, q, d) = 0$$

3. *Si fijamos todos los parámetros de δ salvo el nuevo estado, δ describe una superposición de nuevos estados que deben ser consistentes con los datos fijados. El espacio de estados de superposiciones consiste en dos subespacios mutuamente ortogonales, uno para cada dirección, tales que cubren todo el espacio.*

$$\forall (p_1, \sigma_1, \tau_1), (p_2, \sigma_2, \tau_2) \in Q \times \Sigma \times \Sigma \quad \delta(p_1, \sigma_1, \tau_1, q, L) \cdot \delta(p_2, \sigma_2, \tau_2, q, R) = 0$$

Demostración. Sea U el operador de evolución en el tiempo de una MTC $M = (\Sigma, Q, \delta)$. Sabemos por la Proposición 1.1 que M está bien formada si y sólo si existe U^* cumpliendo $U^* \cdot U = I$ o, equivalentemente, las columnas de U tienen norma uno y son mutuamente ortogonales. Obviamente, la primera condición es consecuencia directa.

En general, configuraciones cuyas cintas difieren en una celda que no estén debajo de ninguna de las cabezas lectoras, o configuraciones cuyas cabezas lectoras no estén en la misma celda o exactamente a dos celdas de distancia, no pueden coincidir en la misma configuración en un solo paso. Por tanto, tales pares de columnas deben ser ortogonales y sólo tenemos que considerar las configuraciones en las que esto no ocurre. La segunda condición justamente especifica la ortogonalidad de pares de columnas de configuraciones que difieran solo en la que en el estado p_1 se lee σ_1 mientras que la otra pareja está en el estado p_2 y se lee σ_2.

Finalmente, consideramos parejas de configuraciones con sus cabezas lectoras a dos celdas de distancia. Si quisiéramos pasar de una configuración a la otra en un

sólo paso es porque como mucho difieren en sus estados y en el símbolo que vayan a escribir en la celda. La tercera condición precisamente especifica la ortogonalidad de parejas de columnas para configuraciones que son idénticas salvo que la cabeza lectora está desplazada exactamente dos celdas.

La condición 3 de separabilidad vista en el Teorema 1.3 nos permite simular cualquier MTC con una que sea unidireccional al aplicar un cambio de base. El mismo cambio nos permite completar cualquier función de transición parcial cuántica que preserve la norma.

Es directo simular una MT determinista con otra que sea unidireccional. Simplemente dividimos los estados q en dos nuevos estados q_r y q_l. El problema es que la máquina resultante no es reversible, puesto que perdemos la relación uno a uno entre estados. Para corregir esto damos el siguiente lema:

Lema 1.6. Dados k conjuntos de estados $Q_0, \dots, Q_{k-1}, Q_k = Q_0$ y k funciones de transición $\delta_i : Q_i \times \Sigma \to \tilde{\mathbf{C}}^{\Sigma \times Q_{i+1} \times \{L,R\}}$ que preserven la norma, existe una MTC bien formada M con conjunto de estados $\bigcup_i (Q_i, i)$ que son accesibles según indican las k funciones de transición.

Demostración. Supongamos que tenemos k funciones de transición. Entonces tomamos M la MTC con el mismo alfabeto, con conjunto de estados dado por la unión de estados $\bigcup_i (Q_i, i)$ y con función de transición asociada a los δ_i de la siguiente forma:

$$\delta((p,i),\sigma) = \sum_{\tau,q,d} \delta_i(p,\sigma,\tau,q,d)|\tau\rangle|q,i+1\rangle|d\rangle.$$

Claramente, la máquina M va saltando según indican $\delta_0, \dots, \delta_{k-1}$ y su operador de evolución en tiempo preserva la norma por herencia de las δ_i.

Lema 1.7 (Unidireccional). Cualquier MTC M es simulada, con coste un factor de 5, por una MTC unidireccional M'. Además, si M se comporta bien y está en forma normal, M' también cumplirá dichas propiedades.

Demostración. La clave está en la condición de separabilidad del las MTC bien formadas que vimos en el Teorema 1.3. Esto significa que podemos dividir $\mathbf{C}^Q$ en dos subespacios mutuamente ortogonales $\mathbf{C}_L$ y $\mathbf{C}_R$ tales que para cada $q \in Q$,

$$\delta(p_1,\sigma_1,\tau_1,q,d) \in \mathbf{C}_d \quad \forall\, (p_1,\sigma_1,\tau_1) \in Q \times \Sigma \times \Sigma,$$

donde $\mathbf{C}_d$ representa el conjunto de amplitudes asociadas a la dirección d.

Como vimos en el Lema 1.5, bajo un cambio de base desde el conjunto de estados Q al conjunto de estados B la nueva función de transición se define como

$$\delta'(v, \sigma, \tau, v', d) = \sum_{p,q} \langle v|p\rangle\langle q|v'\rangle \delta(p, \sigma, \tau, q, d).$$

Así que, tomamos bases ortonormales B_L y B_R de los espacios $\mathbf{C}_L$ y $\mathbf{C}_R$ respectivamente y definimos $M' = (\sigma, B_L \cup B_R, \delta')$ como la MTC resultante del Lema 1.5 que evoluciona exactamente como M bajo el cambio de base desde Q a $B = B_L \cup B_R$.

Entonces cualquier estado en M' es accesible desde una única dirección. Para probar esta afirmación, observamos en primer lugar que para $\delta(p, \sigma, \tau, q, d) \in B_d$, con $d \in \{L, R\}$ y $v = \sum_q \langle v|q\rangle\, |q\rangle$, la condición de separabilidad implica que para todo $v \in B_{\overline{d}}$,

$$\sum_q \delta(p, \sigma, \tau, q, d)\, \langle v|q\rangle^* = 0.$$

Por tanto, para todo $v, \sigma, \tau, v' \in B \times \Sigma \times \Sigma \times B_{\overline{d}}$,

$$\delta'(v, \sigma, \tau, v', d) = \sum_p \langle v|p\rangle \sum_q \langle q|v'\rangle\, \delta(p, \sigma, \tau, q, d) = 0.$$

Es decir, cualquier estado en B es accesible moviéndonos en una sola dirección. El problema de este proceso es que no nos asegura la simulación de M por parte de M'. Para solucionarlo, usamos 5 pasos para simular cada uno de los de M e intercalar las 5 funciones de transición que nos indica el Lema 1.6.

1. Mover a la derecha dejando la cinta y el estados sin modificar:

$$\delta_0(p, \sigma) = |\sigma\rangle|p\rangle|R\rangle.$$

2. Cambiar de la base Q a B mientras nos desplazamos a la izquierda:

$$\delta_1(p, \sigma) = \sum_{b \in B} \langle p|b\rangle|\sigma\rangle|b\rangle|L\rangle.$$

3. M' realiza un paso de la computación de M. Así que, δ_2 es simplemente la función de transición δ' que construimos antes.
4. Deshacemos el cambio de base mientras nos movemos a la izquierda:

$$\delta_3(b, \sigma) = \sum_{b \in Q} \langle b|p\rangle|\sigma\rangle|p\rangle|L\rangle.$$

5. Mover a la derecha dejando la cinta y el estado sin modificar:

$$\delta_4(p,\sigma) = |\sigma\rangle|p\rangle|R\rangle.$$

Si construimos M' con conjunto de estados $(Q\times\{0,1,4\}\cup(B\times\{2,3\})$ usando el Lema 1.6 y fijando los estados inicial y final como $(q_0,0)$, $(q_f,0)$ respectivamente, entonces M' simula M con coste un factor de 5.

Debemos probar que todas las funciones de transición implicadas preservan la norma para que la máquina resultante sea bien formada. Está claro que $\delta_2 = \delta'$ cumple las condiciones porque las hereda de δ. Como δ_0 y δ_4 son deterministas y reversibles, también preservan la norma. Por último, δ_1 y δ_3 cumplen por un lado las propiedades de longitud unitaria y ortogonalidad al ser un cambio de base, y por otro, son separables al realizar un movimiento en una sola dirección.

Para terminar, nos falta probar que si M se comporta bien y está en forma normal, transmite dichas propiedades a M'.

Supongamos por tanto que M se comporta bien y está en forma normal. Entonces existirá un T tal que transcurrido tiempo T la superposición incluye configuraciones con estado q_f con la cabeza lectora/escritora en la celda de salida y para cualquier tiempo previo las configuraciones no contienen a q_f. Esto significa que al introducir x en M', las superposiciones tras $5T$ pasos tiene la cabeza lectora/escritora en la celda de salida e incluyen sólo configuraciones con estado $(q_f,0)$, pero no lo incluyen en ningún tiempo previo. Por lo tanto, M' se comporta bien.

Entonces para cada entrada x existe un T tal que M sucede $T-1$ superposiciones de configuraciones con estados en $Q\setminus\{q_0,q_f\}$ y una superposición de configuraciones finales , todas ellas con estado q_f y la cabeza lectora/escritora en la celda inicial. De lo que se deduce que al introducir x en M' entramos en una serie de $5T-1$ superposiciones de configuraciones con estados en $Q\setminus\{(q_f,0),(q_0,4)\}$y una superposición de configuraciones finales con estado $(q_f,0)$ y la cabeza lectora/escritora sobre la celda inicial. De igual forma que antes, M' se comporta bien y nos permite intercambiar los estados $(q_f,0)$ y $(q_0,4)$ para pasarla a forma normal sin alterar las computaciones de x por M'.

Combinando reversibilidad con patrones de interferencia conseguimos simplificar en gran medida las máquinas a considerar. El efecto de interferencia se ilustra con el siguiente ejemplo:

Ejemplo 1.1. Consideramos la máquina dada por $Q = \{q\}$, $\Sigma = \{0, 1\}$ y la función $\delta(q, b_1, b_2, q, R) = -\frac{1}{\sqrt{2}}$ si $b_1 = b_2 = 1$ ó $\frac{1}{\sqrt{2}}$, y $\delta = 0$ en otro caso.

Puesto que δ cumple las condiciones del Teorema 1.3 trivialmente, se trata de una máquina bien formada. Si observamos las configuraciones c_1, c_2 que difieran solo en la cabeza lectora, nos vemos obligados a acabar en la misma configuración final asociada a 0 cuya amplitud es $\frac{1}{\sqrt{2}}$. Por tanto, esta máquina tan simple exhibe las dificultades de interferencias.

1.2.5 Matrices especiales

Definición 1.14. *Una matriz $d \times d$ unitaria M se dice casi trivial si satisface una de las siguientes condiciones:*

- *M es la identidad exceptuando una de las entradas de la diagonal, que es de la forma $e^{i\theta}$ con $\theta \in [0, 2\pi]$.*
- *M es la identidad exceptuando que tiene una submatriz de dimensión 2×2 definida por los índices i, j, con $i \neq j$, que constituye una rotación de ángulo $\theta \in [0, 2\pi]$. Es decir, si existe un θ y $i \neq j$ tales que $Me_i = (\cos\theta)e_i + (\sin\theta)e_j$, $Me_j = -(\sin\theta)e_i + (\cos\theta)e_j$ y $Me_k = e_k$ en el resto.*

Observación 1.6. Describiremos las matrices casi triviales de la siguiente forma:
Si M es del primer tipo, $e^{i\theta}$ de dimensión j, lo denotaremos por (j, j, θ).
Si M es del segundo tipo, con ángulo de rotación θ, lo denotaremos por (i, j, θ).

Para dichas matrices tenemos el siguiente lema técnico del cual encontramos la prueba en el texto de Bernstein y Vazirani [4].

Lema 1.8. Existe un algoritmo determinista que con entrada $v \in \mathbf{C}^d$ y una cota $\varepsilon > 0$ computa matrices casi triviales $U_1, \ldots, U_{2d-1}$ tal que

$$\|(U_1 \cdots U_{2d-1}v - |v|e_1)\| \leq \varepsilon$$

donde e_1 es el vector unitario $(1, 0, \ldots, 0)$. El tiempo de computación de este algoritmo está acotado polinomialmente en d, $\log\frac{1}{\varepsilon}$ y la longitud de la entrada.

Observación 1.7. La norma usada en el lema anterior es

$$\|M\| = \max_{|v|=1} |Mv|.$$

Otro tipo de matrices simples que nos interesará para aproximar matrices (y por tanto, evoluciones en tiempo de las máquinas) son las siguientes:

Definición 1.15. *Un operador lineal U se dice ε-cercano a unitario si existe un operador unitario V tal que $\|U - V\| \leq \varepsilon$.*

Asociados a los operadores lineales tenemos siempre una matriz a la que podemos otorgar de una definición completamente análoga. Para dichas matrices, existe una serie de lemas técnicos con demostraciones simples. Por ello los dejaremos sin demostrar. En cualquier caso se puede encontrar el razonamiento en el texto de Bernstein y Vazirani [4].

Lema 1.9. Si una matriz compleja M de dimensión d es ε-cercana a unitaria, entonces

$$1 - \varepsilon \leq |M_i| \leq 1 + \varepsilon, \tag{1.1}$$

$$\|M_i M_j^*\| \leq 2\varepsilon + 3\varepsilon^2 \quad \forall i \neq j. \tag{1.2}$$

Entendemos por M_i la fila i-ésima de la matriz M.

Lema 1.10. Si M es una matriz compleja de dimensión $d \times d$ tal que $|m_{i,j}| \leq \varepsilon$ para todo i, j, entonces $\|M\| \leq d\varepsilon$.

Definición 1.16. *Se dice que una matriz compleja U es k-simple para sus primeras k filas y columnas si forman la matriz identidad*

Estas matrices actúan como puente entre las matrices casi triviales, que son similares a las k-simples, y otras más complejas de las que tengamos menos propiedades.

1.2.6 Construcción de una máquina universal

Estamos en disposición de mostrar uno de los pilares fundamentales para alcanzar nuestro objetivo de construir una máquina universal. Para ello nos basaremos en todos los lemas que hemos descrito previamente, que trabajan con matrices sencillas, pero que tienen ciertamente grandes propiedades a la hora de acotar los tiempos de ejecución.

Teorema 1.4. *Existe un algoritmo determinista con tiempo de ejecución polinomial en d, $\log 1/\varepsilon$ y la longitud de la entrada tal que al darle (U, ε) como entrada, $\varepsilon > 0$ y U*

matriz $d \times d$ compleja $\frac{\varepsilon}{2\left(10\sqrt{d}\right)^d}$-cercana a unitaria, computa matrices d-dimensionales casi triviales $U_1, \ldots, U_n$, con n polinomial en d tal que $\|U - U_n \cdots U_1\| \leq \varepsilon$.

Demostración. Si U es d-simple, tendríamos que $U = I$ y la computación sería trivial. Nuestra idea es hacer una recursión en la dimensión de la matriz k-simple, para que en cada paso del algoritmo podamos reducirlo a aproximar una matriz $(k+1)$-simple hasta llegar al caso base.

Supongamos que queremos aproximar U k-simple con una serie de matrices casi triviales con el producto V. Entonces el problema se reduce a computar una serie de matrices casi triviales cuyo producto V es tal que UV^* esté cerca de ser $k+1$-simple. Esto lo conseguimos aplicando el Lema 1.8.

Así pues, sea U una matriz k-simple que sea ε-cercana a unitaria, y sea Z la submatriz $(d-k) \times (d-k)$ inferior derecha de U. De nuevo aplicamos el Lema 1.8 sobre Z_1^t (la primera columna) y δ. Como consecuencia obtenemos una sucesión de matrices $(d-k)$-dimensionales casi triviales $V_1, \ldots, V_{2d-2k-1}$ tales que su producto $V = V_1 \cdot \cdots \cdot V_{2d-2k-1}$ tiene la propiedad $\|(V Z_1^t - |Z_1|e_1)\| \leq \delta$.

Ahora supongamos que extendemos tanto V como los V_i a dimensión d, haciéndolas k-simples y llamamos al producto $W = UV^*$. Entonces V es unitaria y W k-simple. De hecho, puesto que U es δ-cercana a unitaria, W también lo será. Es más, W está cerca de ser $(k+1)$-simple y afirmamos que la $(k+1)$-ésima fila de W satisface $\|W_{k+1} - e_{k+1}^t\| \leq 2\delta$ y que las entradas de la $(k+1)$-ésima columna de W satisfacen $|w_{j,k+1}| \leq 6\delta$ para $j \neq k+1$.

Así que, sea X la matriz $d \times d$ k-simple tal que

$$x_{11} = 1,\ x_{j1} = 0 \text{ para } j \neq 1,\ x_{1j} = 0 \text{ para } j \neq 1,\ x_{jl} = w_{jl} \text{ para } j, k < l+1$$

De nuestras cotas sobre W sabemos que $\|W - X\| \leq 2\delta - 6\sqrt{d}\delta$. Puesto que W es δ-cercana a unitaria, podemos concluir que X es $3\delta + 6\delta\sqrt{d}$-cercana a unitaria.

Como no podemos computar las entradas de $W = UV^*$, necesitamos usar una aproximación de la misma. Para ello, apelamos al Lema 1.10 usando como cota para las entradas δ/d, obteniendo una matriz W' tal que $\|W' - W\| \leq \delta$. De igual forma que antes, asociamos a W' una matriz X', que con propiedades básicas de desigualdades se prueba $\|W - X'\| \leq 3\delta + 6\sqrt{d}\delta$ y X' será $4\delta + 6\sqrt{d}\delta$-cercana a unitaria. Está claro que si nos permitimos cometer un error de $4\delta + 6\sqrt{d}\delta$, tan solo nos quedaría aproximar X' k-simple como producto de matrices casi triviales.

Si renombramos $\delta' = 10\sqrt{d}\delta$, claramente δ' nos sirve simultáneamente como cota para el error de aproximar W a X' y para que X sea δ'-cercana a unitaria. Como consecuencia, el error total de aproximación está acotado por ε

$$\sum_{j=1}^{d} \left(10\sqrt{d}\right)^{j} \delta \leq 2\left(10\sqrt{d}\right)^{d} \delta \leq \varepsilon,$$

puesto que por hipótesis U es δ-cercana a unitaria con $\delta = \frac{\varepsilon}{2\left(10\sqrt{d}\right)^{d}}$.

Este pequeño algoritmo toma tiempo de ejecución polinomial en d y $\log \frac{1}{\varepsilon}$. El algoritmo completo consistirá en iterar d veces este proceso, primero aplicando el Lema 1.8 para computar V y luego haciendo lo propio para X'. Dado que cada iteración toma tiempo polinomial en d y $\log \frac{\left(10\sqrt{d}\right)^{d}}{\varepsilon}$, las d iteraciones en total toman tiempo polinomial en d y $\log \frac{1}{\varepsilon}$.

Finalmente, resta probar nuestra afirmación previa sobre la $(k+1)$-ésima fila y columna de W. Para probar esto, primero recordamos que V vista solo en $(d-k) \times (d-k)$ satisface $\|(VZ_1^t - |Z_1|e_1)\| \leq \delta$. Por tanto, V extendida a la dimensión completa satisface $\|(VU_{k+1}^t - |U_{k+1}|e_{k+1})\| \leq \delta$. Como $1-\delta \leq |U_{k+1}| \leq 1+\delta$, se sigue que $\|VU_{k+1}^t - e_{k+1}\| \leq 2\delta$. Por tanto, la $(k+1)$-ésima fila de W satisface $\|W_{k+1} - e_{k+1}^t\| \leq 2\delta$.

Para probar la propiedad en las columnas, recordamos que W era δ-cercana a unitaria. Esto significa por el Lema 1.9 que $|W_{k+1}W_j^*| \leq 2\delta + 3\delta^2$. Usando la condición previa $\|W_{k+1} - e_{k+1}^t\| \leq 2\delta$ obtenemos $|w_{k+1,k+1}| \geq 1 - 2\delta$. Si denotamos por $\widehat{W}_j$ al vector $(d-1)$-dimensional obtenido al eliminar $w_{j,k+1}$ de W_j, la condición de que W_{k+1} está cerca de e_{k+1}^t también implica que $|\widehat{W}_{k+1}| \leq 2\delta$. De igual forma, el hecho de que W sea δ-cercana a unitaria implica que $|\widehat{W}_j| \leq 1+\delta$.
Combinando las desigualdades, obtenemos

$$2\delta + 3\delta^2 \geq |W_{k+1}W_j^*| = |w_{k+1,k+1}w_{j,k+1}^* + \widehat{W}_{k+1}\widehat{W}_j^*| \geq |w_{k+1,k+1}w_{j,k+1}^*| + |\widehat{W}_{k+1}\widehat{W}_j^*|.$$

Despejando obtenemos $|w_{k+1,k+1}w_{j,k+1}^*| \leq 4\delta + 5\delta^2$. Es decir,

$$|w_{j,k+1}| \leq \frac{4\delta + 5\delta^2}{1-2\delta}.$$

Como podemos asumir que $\delta \leq 1/10$, tenemos que $|w_{j,k+1}| \leq 6\delta$.

Lema 1.11. Sea $\mathcal{R} = 2\pi \sum_{i=1}^{\infty} 2^{-2^i}$. Entonces existe un algoritmo determinista tomando tiempo polinomial en $\log(1/\varepsilon)$ y la longitud de la entrada, que con entrada (θ, ε),

$\theta \in [0, 2\pi]$ y $\varepsilon > 0$ produce como salida un entero k tal que

$$|k\mathcal{R} - \theta| \leq \varepsilon \quad \text{mód } 2\pi.$$

Demostración. En primer lugar, buscamos un n que sea potencia de 2 y cumpla $\varepsilon > \frac{2\pi}{2^{n-1}}$. Después aproximamos $\frac{\theta}{2\pi}$ como una fracción con denominador 2^n. Es decir, encontrar un $m \in [1, 2^n]$ tal que

$$\left| \frac{\theta}{2\pi} - \frac{m}{2^n} \right| \leq \frac{1}{2^n}.$$

Entonces tomamos $k = m2^n$ para que cumpla

$$\begin{aligned} m2^n\mathcal{R} \quad \text{mód } 2\pi &= \left(2\pi m \sum_{i=1}^{\infty} 2^{n-2^i} \right) \\ &= \left(2\pi m \sum_{i=\log n+1}^{\infty} 2^{n-2^i} \right) \\ &= \left(\frac{2\pi m}{2^n} + 2\pi m \sum_{i=\log n+2}^{\infty} 2^{n-2^i} \right), \end{aligned}$$

y teniendo en cuenta

$$m \sum_{i=\log n+2}^{\infty} 2^{n-2^i} \leq m2^{n-4n+1} \leq 2^{n-3n+1} \leq 2^{-2n+1},$$

llegamos a la desigualdad requerida

$$|m2^n\mathcal{R} - \theta| \leq \left| m2^n\mathcal{R} - \frac{2\pi m}{2^n} \right| + \left| \frac{2\pi m}{2^n} - \theta \right| \leq \frac{2\pi}{2^{2n-1}} + \frac{2\pi}{2^n} < \frac{2\pi}{2^{n-1}} < \varepsilon \quad \text{mód } 2\pi.$$

Llegados a este punto debemos expresar lo que significará para nosotros el que una MTC pueda llevar a cabo una secuencia de transformaciones casi triviales con una precisión especificada. En el caso clásico de MT no existe este problema, puesto que las computaciones no tienen un factor probabilístico y siempre nos dan resultados exactos.

Definición 1.17. *Sea $\Sigma \cup \#$ el alfabeto del primer tramo de M una MTC. Sea $\mathcal{V}$ el espacio vectorial complejo de superposiciones de cadenas de longitud k en Σ. Sea U*

una transformación lineal en $\mathcal{V}$ y x_U una cadena que codifique U. Diremos que x_U produce que M lleve a cabo la transformación U con precisión ε en tiempo T si para cada $|\phi\rangle \in \mathcal{V}$, con entrada $|\phi\rangle|x_U\rangle|\varepsilon\rangle$, M para en exactamente T pasos dejando el cabezal lector/escritor en la celda inicial en superposición $(U'|\phi\rangle)|x\rangle$, donde U' es una transformación unitaria de $\mathcal{V}$ tal que $\|U - U'\| \leq \varepsilon$.
Además, para una familia A de transformaciones, diremos que M lleva a cabo dichas transformaciones en tiempo polinomial si T está acotado por un polinomio en $1/\varepsilon$ y la longitud de la entrada.
En el caso de que A contenga transformaciones no unitarias, diremos que M lleva a cabo el conjunto de transformaciones A con factor de cercanía c si para cualquier $\varepsilon > 0$ y $U \in A$ que sea $c\varepsilon$-cercano a unitario, existe una transformación unitaria U' con $\|U' - U\| \leq \varepsilon$ tal que $|x_U\rangle|\varepsilon\rangle$ produce que M lleve a cabo la transformación U' con tiempo polinomial en $1/\varepsilon$ y la longitud de la entrada.

Lema 1.12. Existe una MTC M estacionaria, en forma normal y con alfabeto en el primer tramo $\{\#, 0, 1\}$ que lleva a cabo el conjunto de transformaciones casi triviales en su primer tramo en tiempo polinomial.

Demostración. Usando la notación (x, y, θ) para transformaciones casi triviales vamos a construir dos MTC M_1, M_2, que lleven a cabo los dos tipos de transformaciones. Posteriormente estas dos máquinas se pueden componer de forma apropiada usando los Lemas 1.2 y 1.3, junto con el Teorema de sincronización 1.2.

Así pues, procedemos a la construcción de la MTC M_1 que lleve a cabo las transformaciones casi triviales del tipo $(j, j, 0)$. Esta debe realizar las siguientes tareas:

1. Calcular k tal que $|k\mathcal{R} - \theta| \leq \varepsilon \mod 2\pi$. (El Lema 1.11 nos asegura que es posible)
2. Transformar (w, x, y) en (b, x, y, z), donde $b = 0$ si $w = x$, $b = 1$ si $w = y$, $w \neq x$ y $b = \#$ en otro caso. Mientras que z vale w si $b = \#$ y la cadena vacía en el resto de caso.
3. Aplicar la rotación k veces en el primer qubit de z.
4. Revertir el paso 2 transformando $(\#, x, y, w)$ con $w \neq x$ en (w, x, y), transformando $(0, x, y)$ en (x, x, y) y transformando $(1, x, y)$ con $x \neq y$ en (y, x, y).
5. Revertir el paso 1 borrando k.

Observamos que la longitud de la salida en los pasos 1, 2, 4 y 5 dependen exclusivamente de la longitud de la entrada. Por tanto, usando el Lema 1.11 y el Teorema de sincronización 1.2, podemos construir una MTC de tiempo polinomial, en forma

normal y estacionaria en cada uno de esos pasos. De hecho, el tiempo de computación solo depende de la longitud de (w, x, y).

Para completar la construcción debemos dar una máquina para la rotación del paso 3 y que tenga las mismas propiedades que el resto de pasos. Definimos la MTC $\boldsymbol{R}$ con alfabeto $\{\#, 0, 1\}$, conjunto de estados $\{q_0, q_1, q_f\}$, función de transición dada por

	#	0	1
q_0	$\|\#\rangle\|q_1\rangle\|L\rangle$	$\cos\mathcal{R}\|0\rangle\|q_1\rangle\|L\rangle+$ $+\sin\mathcal{R}\|1\rangle\|q_1\rangle\|L\rangle$	$-\sin\mathcal{R}\|0\rangle\|q_1\rangle\|L\rangle+$ $+\cos\mathcal{R}\|1\rangle\|q_1\rangle\|L\rangle$
q_1	$\|\#\rangle\|q_f\rangle\|R\rangle$		
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

que toma tiempo constante y aplica la rotación $\mathcal{R}$ entre el contenido de la celda de inicio $|0\rangle$ y $|1\rangle$, mientras que deja las demás entradas sin modificar. Insertando $\boldsymbol{R}$ para el estado especial en el Lema 1.4 podemos construir una MTC en forma normal que aplica la rotación $k\mathcal{R}$ entre las entradas $|0, k\rangle$ y $|1, k\rangle$, mientras que deja $|\#, k\rangle$ sin modificar. Puesto que la máquina que estamos aplicando en bucle es estacionaria y de tiempo constante, la máquina resultante es estacionaria y toma tiempo dependiente de k.

Finalmente, aplicando los Lemas 1.2 y 1.3 podemos concatenar las cinco máquinas resultantes de cada paso, obteniendo una MTC $\boldsymbol{M}_1$ estacionaria y en forma normal que realiza la computación deseada.

El caso de $\boldsymbol{M}_2$ llevando a cabo transformaciones casi triviales del tipo (j, k, θ) es igual que el de $\boldsymbol{M}_1$, salvo la función de transición que debe ser reemplazada por la siguiente:

	#	0	1
q_0	$\|\#\rangle\|q_1\rangle\|L\rangle$	$e^{i\mathcal{R}}\|0\rangle\|q_1\rangle\|L\rangle$	$\|1\rangle\|q_1\rangle\|L\rangle$
q_1	$\|\#\rangle\|q_f\rangle\|R\rangle$		
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

Ahora que podemos aproximar transformaciones unitarias por el producto de transformaciones casi triviales, que podemos llevar a cabo posteriormente, estamos en disposición de construir una MTC que aplique una aproximación de una transformación unitaria dada.

Teorema 1.5. *Existe una MTC M estacionaria, en forma normal, con alfabeto de primer tramo $\{\#, 0, 1\}$ que lleva a cabo el conjunto de todas las transformaciones en su primer tramo y lo hace en tiempo polinomial con factor de cercanía requerido $\frac{1}{2\left(10\sqrt{d}\right)^d}$ para transformaciones de dimensión d.*

Demostración. Dado $\varepsilon > 0$ y una transformación U de dimensión $d = 2^k$ que sea $\frac{\varepsilon}{2\left(10\sqrt{d}\right)^d}$-cercana a unitaria, podemos llevar a cabo U con error ε en las primeras k celdas del primer tramo usando los siguientes pasos:

1. Calcular y escribir en tramos vacíos una lista de transformaciones casi triviales $U_1, \ldots, U_n$ tales que $\|U - U_n \cdots U_1\| \leq \varepsilon/2$, con n polinomial en 2^k. También anotar $\varepsilon/2n$.
2. Aplicar las transformaciones $U_1, \ldots, U_n$, cada una de ellas con error máximo $\varepsilon/2n$.
3. Borrar $U_1, \ldots, U_n$ y $\varepsilon/2n$.

Podemos construir una MTC que complete estos pasos como sigue.
Primero, usamos el Teorema 1.4 y el Teorema 1.2 de sincronización para construir una MTC estacionaria, en forma normal y de tiempo polinomial en U y ε para los pasos 1 y 3.
Para finalizar, construimos una MTC estacionaria, en forma normal, que tome tiempo polinomial en 2^k y $1/\varepsilon$ para completar el paso 2. Para ello, hacemos uso del Lema 1.12, que nos proporciona un MTC estacionaria, en forma normal, que puede aplicar cualquier transformación casi trivial con un error acotado por ε. De nuevo aplicamos el Teorema de sincronización para construir una máquina que rote U_1 al final de la lista de transformaciones. Si concatenamos ambas máquinas obtenemos una MTC estacionaria cuyo tiempo de computación sólo depende de ε y los U_i. Así que podemos insertarlo como el estado especial del Lema 1.4, que ahora sí nos devuelve la máquina deseada para el paso 2.

La concatenación de las tres máquinas nos aporta la máquina deseada M'. Puesto que los tiempos de computación de estas tres máquinas son independientes del contenido del primer tramo, también lo será M'. Además, cuándo ejecutamos M' utilizamos las primeras k celdas del primer tramo y aplicamos una transformación unitaria U' cumpliendo

$$\|U' - U\| \leq \|U' - U_n \cdots U_1\| + \|U_n \cdots U_1 - U\| \leq n\frac{\varepsilon}{2n} + \frac{\varepsilon}{2} \leq \varepsilon.$$

Teorema 1.6. *Existe una MTC $\mathcal{M}$ en forma normal tal que para cualesquiera MTC M bien formada, $\varepsilon > 0$ y entero T, $\mathcal{M}$ simula T pasos de M con precisión ε y coste en tiempo polinomial en T y $1/\varepsilon$.*

Demostración. El guión que seguiremos para esta prueba consiste en construir una MTC M' unidireccional con el Lema 1.7 que simule M con coste en tiempo un factor de orden 5. Posteriormente simular M' para tener la construcción completa. Este sería el proceso natural, pero nosotros probaremos primero la factibilidad de simular M' y luego volveremos al Lema 1.7.

Supongamos que $M = (\Sigma, Q, \delta)$ es una MTC unidireccional y queremos simular nuestra MTC universal. Usaremos una cinta de nuestra MTC universal específicamente para simular la configuración actual de M. Dado que el alfabeto y conjunto de estados de M puede tener cualquier tamaño prefijado, necesitemos usar $\log(\text{card}(Q \times \Sigma))$ celdas de nuestra cinta, que llamaremos supercelda, para simular cada una de las celdas de M. Cada una de estas superceldas contiene una pareja de enteros p, σ, donde $\sigma \in [1, \text{card}(\Sigma)]$ representa el contenido de la celda correspondiente de M, y $p \in [0, \text{card}(Q)]$ representando el estado de M si la cabeza lectora/escritora observa la celda correspondiente ($p = 0$ en otro caso). Puesto que la cabeza lectora/escritora de M solo puede moverse a una distancia T de la celda inicial en tiempo T, solo necesitamos superceldas para las $2T + 1$ celdas centradas en la casilla inicial de M.

Con este pequeño truco podemos dejar de preocuparnos por actualizar la dirección. Entonces δ será una transformación unitaria U de dimensión $d = \text{card}(Q \times \Sigma)$ que va desde un par (estado, símbolo) leído, a una superposición nueva de estado y símbolo. Esto significa que podemos dividir el proceso de actualizar la superposición en dos pasos, uno para aplicar U y otro para movernos la especificación de nuevos estados a izquierda o derecha de la supercelda, según indique la dirección en la que se entra al estado en M.

A continuación construimos una MTC STEP que lleva a cabo un paso de la simulación. Además de la cinta que usemos para simular, la máquina recibe como entrada la precisión deseada γ, una especificación de U que sea $\frac{\gamma}{2\left(10\sqrt{d}\right)^d}$-cercana a unitaria, y una cadena $s \in \{0,1\}^{\text{card}(Q)}$ que indique la dirección en la que se entra a los estados de M. La máquina STEP opera como sigue:

1. Transferir el estado actual y símbolo (p, σ) a la zona más cercana a la celda inicial que esté vacía y dejar una marca especial en su lugar.
2. Aplicar U a (p, σ) tomando γ en consideración, que los transforma en un nuevo

par de estado y símbolo (q, τ).
3. Revertir 1, transfiriendo (q, τ) a donde dejamos el marcador y vaciando el contenido de la supercelda de partida.
4. Transferir la especificación del estado q una supercelda a izquierda o derecha en función de si el q-ésimo qubit de s es 0 o 1.

Usando el Teorema 1.2 de sincronización, podemos construir MTCs en forma normal y estacionarias para los pasos 1, 3 y 4, que toman tiempo polinomial en T y solo dependen de T (fijado el M). El paso 2 del algoritmo puede ejecutarse en tiempo polinomial en $\text{card}(\Sigma)$, $\text{card}(Q)$ y γ usando la construcción del Teorema 1.5 de transformaciones unitarias. Aplicando de forma apropiada los Lemas 1.2 y 1.3, obtenemos de estas cuatro MTCs en forma normal la MTC en forma normal que llamamos STEP.

Puesto que cada una de estas cuatro MTCs toman un tiempo que solo depende de T y ε, también lo hará STEP (siempre para una M prefijada). Por lo tanto, si introducimos STEP como el estado especial en la MT reversible construida en el Lema 1.4 y añadimos T a la entrada, la MTC resultante STEP' para tras un tiempo polinomial en T y $1/\varepsilon$, además de haber simulado T pasos de M con precisión $T\varepsilon$.

Finalmente, construimos la MTC universal $\mathcal{M}$ ensamblando STEP' tras una MTC que realice el preproceso adecuado.

En general, la máquina universal debe simular MTCs que no sean unidireccionales. Así pues, el preproceso sobre una MTC M, con una entrada x y una precisión ε consistirá en llevar a cabo la construcción del Lema 1.7 para obtener una MTC M' unidireccional que simule M con coste un factor de 5. Entonces, las entradas que realmente recibe STEP' son:

1. La representación de las $2T + 1$ superceldas de la configuración inicial de M' con entrada x.
2. La transformación d-dimensional U para M' con cada entrada escrita con precisión $\frac{\varepsilon}{40T\left(10\sqrt{d}\right)^{d+2}}$.
3. La cadena de direcciones s para M'.
4. EL número de pasos a simular, $5T$, y la precisión deseada $\gamma = \frac{\varepsilon}{40T}$.

Veamos de dónde provienen estas exigencias.

Todas estas entradas para $\mathcal{M}$ pueden ser computadas de forma determinista con tiempo polinomial en T, $1/\varepsilon$ y la longitud de la entrada.

Además, si la transformación U se computa con la precisión especificada en (2), la transformación para STEP estará en un rango de $\frac{\varepsilon}{40T\left(10\sqrt{d}\right)^d}$ de la unitaria deseada U, es decir, será $\frac{\varepsilon}{40T\left(10\sqrt{d}\right)^d}$-cercana a unitaria, como pedíamos cuando definimos STEP.

Así, cada vez que aplicamos STEP con precisión $\varepsilon/40T$, habremos aplicado una transformación unitaria a distancia $\varepsilon/20T$ de U. Por tanto, tras $5T$ procesos de STEP, habremos aplicado una transformación unitaria que está a distancia $\varepsilon/4$ de la transformación de $5T$ pasos de M'.

Esto significa que observando la cinta de simulación de $\mathcal{M}$ tras haberse completado, podremos dar una muestra de una distribución que esté a distancia variacional total ε de la distribución que pudiéramos obtener al observar M con entrada x en tiempo T.

2 Clases de complejidad

2.1 Complejidad en tiempo en el paradigma cuántico

Durante el primer capítulo hemos definido rigurosamente lo que entendemos por una MTC. Al igual que sucede en el entorno de MT clásicas, estas máquinas tienen asociado un lenguaje aceptado. Como suele ser habitual, la noción de aceptación de un lenguaje tiene asociada una medida de complejidad computacional, generalmente espacio y tiempo. Nuestra misión en este capítulo será la de extender los conceptos de clases de complejidad clásicos a nuestras nuevas máquinas y si es posible, compararlos. La mayoría de relaciones de contención entre las clases están aún por determinar, aunque gracias a Bernstein y Vazirani [4] sabemos que **BQP** $\subseteq$ **PSPACE**.

Además, se cree que **NP** $\nsubseteq$ **BQP**. Como veremos en el próximo capítulo, existen varios algoritmos que resuelven problemas clásicos pero desarrollados en el entrono de la computación cuántica. El ejemplo que estudiaremos se debe a David Deutsch y Richard Jozsa [6], que aportaron un algoritmo cuántico con oráculo de coste en tiempo polinomial que resuelve el problema planteado por el propio Deutsch. Se escoge específicamente este problema porque no es posible encontrar un algoritmo clásico con el mismo oráculo que tome tiempo polinomial.

Definición 2.1. *Sea una MTC M estacionaria, en forma normal y con múltiples tramos cuyo último tramo tiene alfabeto $\{\#, 0, 1\}$. Si ejecutamos M con entrada x en el primer tramo y la cadena vacía en el resto, la máquina para y observamos la celda de entrada de su último tramo, veremos un 1 con probabilidad p. Diremos que M acepta x con probabilidad p y rechaza x con probabilidad $1 - p$.*

La aceptación de una cadena está irremediablemente ligada a fijar una casilla de

aceptación y observarla. Puesto que el proceso de observación tiene una probabilidad asociada a las magnitudes de la configuración, nuestra definición de aceptación de un lenguaje debe darse en términos probabilísticos como los expuestos anteriormente.

Para un lenguaje genérico $\mathcal{L} \subseteq (\Sigma - \#)^*$ tenemos la siguiente definición.

Definición 2.2. *Diremos que una MTC M acepta exactamente $\mathcal{L}$ si M acepta cada cadena $x \in \mathcal{L}$ con probabilidad 1 y rechaza cada cadena $x \in (\Sigma - \#)^* - \mathcal{L}$ con probabilidad 1.*

Esta definición aún persigue en cierta medida el concepto clásico de aceptar un lenguaje, puesto que impone que no haya errores al observar. Podemos relajar cuanto queramos dicha restricción, aunque generalmente se exige una probabilidad estrictamente superior a $1/2$.

2.2 Clases de complejidad cuánticas

Una vez aclarado qué entendemos por aceptar un lenguaje en este nuevo marco, podemos definir una serie de clases de complejidad asociadas al tiempo de computación que toma la máquina para aceptar o rechazar un lenguaje.

Definición 2.3. *Definimos la clase* **EQP** *(error-free quantum polynomial time) como el conjunto de lenguajes que son aceptados exactamente por alguna MTC de tiempo polinomial.*

Definición 2.4. *Definimos la clase* **BQP** *como el conjunto de lenguajes que son aceptados con probabilidad $2/3$ por alguna MTC de tiempo polinomial.*

Estas son las dos clases principales, aunque existen muchas otras que podrían llegar a ser útiles. Está claro que **EQP** $\subseteq$ **BQP**. Por la caracterización que vimos en el capítulo anterior 1.1, toda MT reversible es un caso especial de MTC, por lo que **P** $\subseteq$ **EQP**.

2.3 Comparación con clases de complejidad clásicas

Antes de entrar en materia, es conveniente recordar la versión clásica de la clase **BQP**, para luego comprobar si las contenciones de clases asociadas son iguales en el mundo clásico como en el cuántico.

Definición 2.5. *Definimos la clase* **BPP** *como el conjunto de todos los lenguajes que se pueden computar por una máquina clásica (probabilística) en tiempo polinomial.*

De esta definición deducimos que **BPP** $\subseteq$ **BQP**, aunque es recomendable precisar los detalles al menos una vez:

Sea $\mathcal{L}$ un lenguaje en **BPP**. Entonces existe un polinomio $p(n)$ y una MT determinista de tiempo polinomial M con salida $\{0,1\}$ que satisface lo siguiente: Para cualquier entrada x de longitud n, si llamamos S_x al conjunto de $2^{p(n)}$ qubits computados por M con entradas $(x;y)$ con $y \in \{0,1\}^{p(n)}$, entonces la proporción de 1's en S_x es al menos $2/3$ siempre que $x \in \mathcal{L}$ y $1/3$ en otro caso. Podemos usar una MTC para decidir si una cadena x está en el lenguaje $\mathcal{L}$ creando primero una superposición que divida equitativamente todos los $|x\rangle|y\rangle$ y luego usar el algoritmo determinista.

Primero, encadenamos una MTC estacionaria y en forma normal que tome x como entrada y devuelva $(x;0^{p(n)})$ con otra MTC estacionaria y en forma normal construida como sigue. Tomamos como alfabeto $\{\#,0,1\}$ y conjunto de estados $\{q_0,q_a,q_b,q_c,q_f\}$. Describimos la función de transición con una tabla:

	#	0	1
q_0		$\|0\rangle\|q_a\rangle\|R\rangle$	$\|1\rangle\|q_a\rangle\|R\rangle$
q_a	$\|\#\rangle\|q_b\rangle\|L\rangle$		
q_b	$\|\#\rangle\|q_c\rangle\|L\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle\|q_b\rangle\|R\rangle + \frac{1}{\sqrt{2}}\|1\rangle\|q_b\rangle\|R\rangle$	$\frac{1}{\sqrt{2}}\|0\rangle\|q_b\rangle\|R\rangle - \frac{1}{\sqrt{2}}\|1\rangle\|q_b\rangle\|R\rangle$
q_c	$\|\#\rangle\|q_f\rangle\|L\rangle$	$\|0\rangle\|q_c\rangle\|L\rangle$	$\|1\rangle\|q_c\rangle\|L\rangle$
q_f	$\|\#\rangle\|q_0\rangle\|R\rangle$	$\|0\rangle\|q_0\rangle\|R\rangle$	$\|1\rangle\|q_0\rangle\|R\rangle$

Esta máquina es estacionaria, se encuentra en forma normal y para en tiempo polinomial respecto a la entrada. Aunque no se exprese como una función total, es sencillo extender su función de transición para que así lo sea, sin afectar a la funcionalidad de la versión parcial aquí mostrada.

Esto nos da una MTC estacionaria y en forma normal que con entrada x produce una superposición

$$\sum_{y\in\{0,1\}^{p(n)}} \frac{1}{2^{p(n)/2}}|x\rangle|y\rangle.$$

Aplicando una vez más el Teorema 1.2 de sincronización para concatenarlo con M obtenemos una MTC de tiempo polinomial que con entrada x produce la superposi-

ción

$$\sum_{y\in\{0,1\}^{p(n)}} \frac{1}{2^{p(n)/2}}|x\rangle|y\rangle|M(x;y)\rangle.$$

Puesto que la proporción de 1's en S_x es de al menos $2/3$ si $x \in \mathcal{L}$ y como mucho $1/3$ en otro caso, observando el qubit del tercer tramo nos dará la clasificación correcta de x con probabilidad al menos $2/3$. Esto completa la comprobación de **BPP** $\subseteq$ **BQP**.

Siguiendo con esta cadena de contenciones observamos que **BQP** está dentro de la clase exponencial en tiempo. Esta cota se puede refinar como veremos posteriormente, pero antes debemos introducir una clase de máquinas que siguen el mismo patrón que las ε-cercanas a unitarias.

Definición 2.6. *Diremos que dos MTC M y M' son ε-cercanas si tienen el mismo conjunto de estados, alfabeto y la diferencia entre cada pareja de amplitudes correspondiente tiene magnitud acotada por $\varepsilon > 0$.*

Al igual que ocurría en el caso de máquinas ε-cercanas a unitarias, esta clase de máquinas también tiene un lema técnico que acota la diferencia de los operadores de evolución en el tiempo. Como consecuencia, tenemos el siguiente corolario.

Corolario 2.1. Sea $M = (\Sigma, Q, \delta)$ una MTC bien formada y sea M' una MTC λ_T^ε-cercana a M, con $\lambda_T^\varepsilon = \frac{\varepsilon}{24\,\mathrm{card}(\Sigma)\,\mathrm{card}(Q)T}$ y $\varepsilon > 0$. Entonces M' simula M durante T pasos con precisión ε.

Los detalles de ambos resultados pueden encontrarse en el texto de Bernstein y Vazirani [4].

Teorema 2.1. **BQP** $\subseteq$ **PSPACE**.

Demostración. Sea $M = (\Sigma, Q, \delta)$ una máquina que decida un lenguaje en **BQP** con tiempo de computación $p(n)$. Por el corolario 2.1, cualquier MTC M' que sea $\lambda_{p(n)}^\varepsilon$-cercana a M simulará M durante $p(n)$ pasos con precisión ε. Si simulamos M con precisión $1/12$, entonces la probabilidad de éxito será al menos $7/12$. Por tanto, nos basta trabajar con la MTC M', donde cada amplitud de M se computará por los primeros $\log(288\,\mathrm{card}(\Sigma)\,\mathrm{card}(Q)p(n))$ qubits.

Recordamos que la amplitud de cualquier configuración tras T pasos es la suma de las amplitudes de cada posible camino de decisiones en M' de longitud T desde la configuración inicial hasta la deseada. En nuestro caso, si mantenemos a lo más $p(n)$ configuraciones intermedias podemos llevar a cabo una búsqueda profunda en el árbol computacional asociado a M para calcular la amplitud de una configuración

deseada. El coste en tiempo es exponencial, pero el espacio usado es polinomial en las entradas.

Por último, debemos determinar si una cadena x de longitud n es aceptada por M. Para ello calculamos la norma de los vectores de magnitud de cada configuración alcanzable en tiempo $p(n)$ por M' con un 1 en la celda de entrada y comparamos con $7/12$. Puesto que las únicas configuraciones de aceptación alcanzables por M' son aquellas que tienen un 1 en la casilla inicial y todas las casillas a distancia mayor de $p(n)$ están en blanco, sólo necesitamos espacio de orden polinomial.

Esta cota puede mejorarse a $\mathbf{P}^{\#\mathbf{P}}$ usando el siguiente teorema de [3], del que no veremos la prueba por motivos de espacio. Aún así, es interesante definir los nuevos conceptos que aparecen en este resultado.

Definición 2.7. *Definimos la clase* $\mathbf{BQTime}(T(n))$ *como el conjunto de lenguajes que son aceptados con probabilidad* $2/3$ *por alguna MTC cuyo tiempo de funcionamiento en cualquier entrada de longitud* n *esté acotado por* $T(n)$.

Definición 2.8. *Diremos que una función* f *es constructible en tiempo si existe una MT* M *tal que al introducir una cadena* $\{1\}^n$*, devuelve la representación binaria de* $f(n)$ *en tiempo* $\mathcal{O}(f(n))$.

Recordamos que $\#\mathbf{P}$ es el conjunto de funciones f que mandan una cadena en un entero para las que existe un polinomio $p(n)$ y un lenguaje $\mathcal{L} \in \mathbf{P}$ tales que al introducir x como entrada, el valor $f(x)$ es el número de cadenas y de longitud $p(|x|)$ para los que xy está contenido en $\mathcal{L}$.

Por último, vamos a entender por una MTC oráculo aquella que tiene un tramo especial para realizar las preguntas al oráculo. Las MTC oráculo tienen dos estados distinguidos correspondientes al estado previo a la pregunta q_q y al posterior q_a. Una pregunta es ejecutada siempre que la máquina entre en el estado q_q con un único bloque no vacío de celdas en el tramo destinado a las preguntas.

Teorema 2.2. *Si el lenguaje* $\mathcal{L}$ *está contenido en la clase* $\mathbf{BQTime}(T(n))$*, con* $T(n) > n$ *y* $T(n)$ *constructible en tiempo, entonces para todo* $\varepsilon > 0$*, existe una MTC* M' *que acepta* $\mathcal{L}$ *con probabilidad* $1-\varepsilon$ *y tiene la siguiente propiedad: cuándo introducimos una entrada* x *de longitud* n*,* M' *opera en tiempo acotado por* $cT(n)$*, donde* c *es polinomial en* $\log(1/\varepsilon)$*, y produce una superposición final en la que* $|x\rangle|\mathcal{L}(x)\rangle$*, con* $\mathcal{L}(x) = 1$ *si* $x \in \mathcal{L}$ *y* 0 *en otro caso, tiene norma al menos* $1-\varepsilon$.

Este resultado nos permite depurar la cota que vimos anteriormente para **BQP** de

la siguiente forma.

Teorema 2.3. $\mathbf{BQP} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

Demostración. Sea $M = (\sigma, Q, \delta)$ una máquina que decida un lenguaje en **BQP** con tiempo de observación $p(n)$. Siguiendo el Teorema 2.2 podemos suponer sin pérdida de generalidad que M se comporta apropiadamente, tal como indica dicho resultado.

Esta máquina, a su vez, puede simularse por otra MTC M' que sea $\lambda^{\varepsilon}_{p(n)}$-cercana a M gracias al Teorema 2.1. De hecho, hemos probado que cada amplitud de M puede ser computada para los primeros $\log(288\ \mathrm{card}(\Sigma)\ \mathrm{card}(Q)p(n))$ qubits, además de mantener la norma de cada configuración final de aceptación por encima de $7/12$.

Puesto que queremos demostrar la pertenencia a $\mathbf{P}^{\#\mathbf{P}}$, vamos a usar **#P** como oráculo para computar la amplitud de la configuración final $(x; 1)$ de M' con un error menor de $1/36$ en tiempo T. Puesto que la amplitud real es como mucho 1, la norma de nuestra configuración aproximada debe estar a distancia $1/12$ de la real. Veamos que es cierto suponiendo que α es el valor real y $\|\alpha' - \alpha\| < 1/36$,

$$\left|\|\alpha\|^2 - \|\alpha'\|^2\right| \leq \|\alpha' - \alpha\|^2 + 2\|\alpha\|\|\alpha' - \alpha\| \leq \frac{1}{36}\left(2 + \frac{1}{36}\right) < \frac{1}{12}.$$

Dado que la probabilidad de éxito de M' es al menos $7/12$, podemos clasificar x correctamente.

Resta probar que efectivamente podemos aproximar esta configuración final. En primer lugar, podemos estudiar cada amplitud separando parte real e imaginaria, obteniendo así una suma de 2^T términos para cada camino que nos lleve a una configuración en tiempo T. De nuevo podemos separar esta suma en cuatro: Reales positivos, reales negativos, imaginarios positivos e imaginarios negativos. Con la ayuda de un algoritmo en **#P** podemos computar cada una de estas sumas con un error de magnitudes menor de $1/144$. Tomando la diferencia entre ellos nos proporcionará la amplitud de a configuración deseada con un error de $1/36$ a lo sumo.

Por último, computemos la suma de los reales positivos como ejemplo del algoritmo. Supongamos que para una constante c polinomial en T recibimos las siguientes entradas (todas de longitud polinomial en T): El camino p de M' en T pasos, una especificación t de alguno de los 2^T términos y un entero w entre 0 y 2^{cT}.

Entonces es sencillo ver que podemos decidir en tiempo polinomial determinista en t si p es realmente un camino desde la configuración inicial de M en x hasta la configuración final deseada, si t es real y positivo, y si el término t de la amplitud del camino p es mayor que $w/2^{cT}$. Si fijamos una camino p y un término t que satisfaga las

condiciones anteriores, entonces el número de enteros w para los que este algoritmo acepta, dividido por 2^{cT}, está a distancia $1/2^{cT}$ del valor del término t del camino p.

Así, si fijamos tan solo un camino p que satisfaga las condiciones previas, el número de términos t y enteros w para los que el algoritmo acepta, dividido por 2^{cT}, está a distancia $1/2^{(c-1)T}$ de la suma de términos reales positivos para el camino p.

Por tanto, el número de elementos p, t, w para los que el algoritmo acepta, dividido por $1/2^{cT}$, está a distancia $N/2^{(c-1)T}$ de la suma de todos los términos reales positivos de todos los caminos de M' con T pasos que parten desde la configuración inicial y terminan en la configuración deseada. Puesto que hay $2\,\mathrm{card}(\Sigma)\,\mathrm{card}(Q)$ posibles sucesores de cualquier configuración, como mucho, eligiendo

$$c > 1 + \frac{\log(144)}{T} + \frac{2\,\mathrm{card}(\Sigma)\,\mathrm{card}(Q)\log(T)}{T}$$

cumple $N/2^{(c-1)T} < 1/144$ como requeríamos.
El razonamiento para el resto de sumas es análogo.

2.3.1 NP y BQP

El hecho de que tanto **NP** como **BQP** contengan a **P** nos lleva a plantear qué relación existe entre ambas clases. Peter Shor construyó en [11] el algoritmo que lleva su nombre, permitiendo resolver en tiempo polinomial con una MTC el problema de la factorización de enteros, cuya pertenencia a **P** se desconoce. De igual forma, existen indicios de que $\mathbf{NP} \not\subseteq \mathbf{BQP}$. Cualquier demostración rigurosa que aclare alguna de estas contenciones se asocia directamente con la respuesta a $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

La aproximación para estudiar $\mathbf{NP} \not\subseteq \mathbf{BQP}$ que veremos se basa en la dada por C.H. Bennett, E. Bernstein, G. Brassard y U. Vazirani [3]. Siguiendo su notación, llamaremos $\mathbf{BQTime}(T(n))^A$ al conjunto de lenguajes aceptados con probabilidad al menos $2/3$ por una MTC con oráculo M^A cuyo tiempo de ejecución esté acotado por $T(n)$. Asociado al oráculo A podemos definir una permutación $A(x)$ que consiste en interpretar la respuesta del oráculo al par (x, i) como el i-ésimo qubit del valor de la función. Para realizar este cálculo, el par (x, i) debe escribirse como una cadena binaria.

En lo que resta de capítulo supondremos sin pérdida de generalidad que las máquinas con las que trabajemos tendrán como alfabeto $\{0, 1, \#\}$ para cada tramo. Además, todos los tramos comienzan en blanco, excepto la destinada a la entrada.

Lema 2.1. Sean $\phi, \psi \in \mathbf{S}$ tales que $\|\phi\| = \|\psi\| = 1$ y $\|\phi - \psi\| \leq \varepsilon$. Entonces la distancia variacional entre las distribuciones de probabilidad resultantes de medir en φ y ψ es como máximo 4ε.

Demostración. Sea $\phi = \sum_i \alpha_i |i\rangle$, $= \sum_i \beta_i |i\rangle$ y $\pi = \phi - \psi = \sum_i (\alpha_i - \beta_i)|i\rangle = \sum_i (-\gamma_i)|i\rangle$. Entonces podemos expresar la probabilidad de i en ψ como

$$\beta_i \beta_i^* = |\alpha_i|^2 + |\gamma_i|^2 + \alpha_i \gamma_i^* + \gamma_i \alpha_i^*.$$

Por tanto, la distancia variacional total entre ambas distribuciones es como mucho

$$\sum_i |\gamma_i|^2 + \alpha_i \gamma_i^* + \gamma_i \alpha_i^* \leq \varepsilon^2 + 2\|\alpha\|\|\gamma\| \leq \varepsilon^2 + 2\varepsilon.$$

Finalmente, notamos que al trabajar con superposiciones unitarias, podemos hacer $\varepsilon \leq 2$.

Definición 2.9. *Sea $|\phi_i\rangle$ la superposición de M^A con entrada x en tiempo i. Denotaremos por $q_y(|\phi_i\rangle)$ a la suma de las magnitudes al cuadrado en $|\phi_i\rangle$ de configuraciones de M que pregunten al oráculo con la cadena y. Llamaremos a $q_y(|\phi_i\rangle)$ la magnitud de pregunta de y en $|\phi_i\rangle$.*

Teorema 2.4. *Sea $|\phi_i\rangle$ la superposición de M^A con entrada x en tiempo i. Sea $\varepsilon > 0$ y $F \subseteq [0, T-1] \times \Sigma^*$ el conjunto de pares tiempo-cadenas tales que*

$$\sum_{(i,y) \in F} q_y(|\phi_i\rangle) \leq \frac{\varepsilon^2}{T}.$$

Si suponemos que la respuesta a una pregunta $(i, y) \in F$ se modifica para algún $a_{i,y}$ prefijado y elegido aleatoriamente y llamamos $|\phi_i'\rangle$ a la superposición resultante, entonces $||\phi_T\rangle - |\phi_T'\rangle| \leq \varepsilon$.

Demostración. Sea U el operador de tiempo unitario asociado a M^A. Sea A_i un oráculo para el que si $(i, y) \in F$ entonces $A_i(y) = a_{i,y}$ y en otro caso $A_i(y) = A(y)$. Sea U_i el operador unitario de M^{A_i}. El error causado por reemplazar A con A_i en el paso i-ésimo será

$$|E_i\rangle = U_i|\phi_i\rangle - U|\phi_i\rangle.$$

Aplicando esta diferencia de forma continuada obtenemos

$$|\phi_T\rangle = U|\phi_{T-1}\rangle = U_T|\phi_{T-1}\rangle - |E_{T-1}\rangle = \ldots = U_t \cdots U_1|\phi_0\rangle - \sum_{i=0}^{T-1} U_{T-1} \ldots U_i|E_i\rangle.$$

Dado que todos los U_i son unitarios, $|U_{T-1} \cdots U_i|E_i\rangle| = ||E_i\rangle|$.

La suma de las magnitudes de los E_i al cuadrado coincide con $\sum_{(i,y)\in F} q_y(|\phi_i\rangle)$, que está acotado por $\frac{\varepsilon^2}{T}$ según las hipótesis. Es decir, el cuadrado de las magnitudes de la suma de los $U_{T-1}\cdots U_i|E_i\rangle$ es como máximo T veces la suma de los cuadrados de las magnitudes. Por tanto $||\phi_T\rangle - |\phi'_T\rangle|^2 \leq \varepsilon^2$.

Corolario 2.2. Sea A un oráculo sobre el alfabeto $\Sigma = \{0,1\}$. Para $y \in \Sigma^*$, sea A_y cualquier oráculo tal que para todo $x \neq y$, $A_y(x) = A(x)$. Sea $|\phi_i\rangle$ la superposición de M^A con entrada x en tiempo i y $|\phi_i\rangle^{(y)}$ la superposición de M^{A_y} con entrada x en tiempo i. Entonces, para cada $\varepsilon > 0$ existe un conjunto S de cardinalidad a lo más $\frac{2T^2}{\varepsilon^2}$ tal que para todo $y \notin S$, $\left||\phi_T\rangle - |\phi_T\rangle^{(y)}\right| \leq \varepsilon$.

La prueba es directa tras el Teorema 2.4 visto previamente.

Teorema 2.5. *Para cualquier $T(n)$ que pertenezca a $o(2^{n/2})$, relativo a un oráculo cualquiera A, con probabilidad* 1, $\mathbf{BQTime}(T(n))^A$ *no contiene* **NP**.

Demostración. Sea $\mathcal{L}_A = \{y : \exists x\ A(x) = y\}$. Claramente este lenguaje pertenece a $\mathbf{NP}^A$. Vamos a probar que para cualquier MTC con oráculo M^A que opere en tiempo acotado superiormente por $T(n)$, con probabilidad 1, no acepta dicho lenguaje.

Dado que $T(n) \in o(2^{n/2})$, para un n suficientemente grande podemos conseguir $T(n) \leq \frac{2^{n/2}}{20}$. Afirmamos que la probabilidad de que M devuelva la respuesta errónea con entrada $\{1\}^n$ es al menos $1/8$ por cada forma de fijar las respuestas del oráculo a las entradas de longitud distinta de n. Esta probabilidad está tomada sobre las decisiones aleatorias del oráculo con entradas de longitud n.

En efecto, fijemos en primer lugar una permutación para cadenas de longitud n. Denotemos por $\mathcal{C}$ al conjunto de oráculos consistentes con dicha permutación y por $\mathcal{A}$ al conjunto de oráculos de $\mathcal{C}$ tales que $\{1\}^n$ no pertenece a $\mathcal{L}_A$. Si el oráculo responde a cadenas de longitud n elegidas de forma aleatoria uniformemente, entonces la probabilidad de que el oráculo esté en $\mathcal{A}$ es al menos $1/4$. Esto se debe a que la probabilidad de que $\{1\}^n$ no pertenezca a $\mathcal{L}_A$ es $\left(\frac{2^n-1}{2^n}\right)^{2^n}$, que es al menos $1/4$ para un $n > 1$. Sea $\mathcal{B}$ el conjunto de oráculos en $\mathcal{C}$ tales que $\{1\}^n$ proviene de un único elemento. Al igual que antes, la probabilidad de que un oráculo escogido aleatoriamente esté en $\mathcal{B}$ es de $\left(\frac{2^n-1}{2^n}\right)^{2^n-1}$, que está acotado superiormente por $1/e$ para $n > 1$.

Dado un oráculo $A \in \mathcal{A}$, podemos modificar su respuesta para una entrada individual y en $\{1\}^n$ y así obtener un oráculo $A_y \in \mathcal{B}$. Por el Corolario 2.2 existe un conjunto S de a lo sumo $338T^2(n)$ cadenas tales que la diferencia entre la i-ésima superposición de M^{A_y} y M^A con entrada $\{1\}^n$ tiene norma $1/13$ como máximo. Usando

el Lema 2.1 podemos concluir que la diferencia entre las probabilidades de aceptación es $1/13 \cdot 4 < 1/3$ como mucho. Puesto que M^{A_y} debe aceptar $\{1\}^n$ con probabilidad al menos $2/3$ y M^A rechazarlo con probabilidad $2/3$, podemos afirmar que M no acepta ni $\mathcal{L}_A$ ni $\mathcal{L}_{A_y}$.

Así, cada oráculo $A \in \mathcal{A}$ que decida correctamente $\{1\}^n \in \mathcal{L}_A$ puede transformarse en al menos $(2^n - \text{card}(S)) \geq 2^{n-1}$ oráculos diferentes que denotaremos por $A_f \in \mathcal{B}$. Estos nuevos oráculos fallan al decidir correctamente si $\{1\}^n \in \mathcal{L}_{A_f}$. Es más, el número de oráculos en $\mathcal{B}$ para los que M falla debe ser al menos la mitad de oráculos de $\mathcal{A}$ para los que M funciona correctamente. Por tanto, si llamamos a al número de oráculos de $\mathcal{A}$ para los que M falla, M debe fallar en al menos $a + \frac{1}{2}(\text{card}(\mathcal{A}) - a)$ de ellos. Es decir, M falla en decidir si $\{1\}^n \in \mathcal{L}_A$ con probabilidad al menos $\frac{1}{2}P[\mathcal{A}] \geq \frac{1}{8}$, donde $P[\mathcal{A}]$ es la probabilidad de que el oráculo esté en $\mathcal{A}$.

En contraposición, M decide la pertenencia a $\mathcal{L}_A$ con probabilidad 0 para un oráculo cualquiera A escogido aleatoriamente de manera uniforme.

Aunque el resultado no tiene implicaciones directas sobre la cuestión $\mathbf{NP} \overset{?}{\subseteq} \mathbf{BQP}$, sí plantea un nuevo enfoque con el que atacar problemas **NP**-completos mediante el uso de una MTC con oráculo. El algoritmo de Grover [8] es un claro ejemplo de que esta aproximación es más eficiente que la clásica. De hecho, este algoritmo permite probar que la cota estudiada en el Teorema 2.5 es ajustada.

3 Algoritmos

Hasta ahora hemos visto cómo las MTC presentan un nuevo paradigma computacional, del que muchos piensan que es esencialmente distinto al clásico. Nos hemos centrado en las teoría computacional de clases que genera y en cómo se relaciona con las principales clases de complejidad clásicas. Nuestra misión en este capítulo es alejarnos del mundo puramente teórico y dar un ejemplo de algoritmos concretos que trabajen en MTC. El algoritmo que estudiaremos a continuación fue desarrollado por Deutsch y Jozsa [6] en el año 1992, promoviendo así la utilidad de máquinas con oráculos. Este algoritmo no está escogido al azar, se trata del primer gran salto en vías de una mejora en eficiencia en cuanto a implementación y resolución de un problema que aparece con gran frecuencia en computación.

3.1 De máquina universal a puertas cuánticas

La presentación del algoritmo de Deutsch-Jozsa se puede hacer con las herramientas que hemos descrito hasta ahora. No obstante, vamos a dar un pequeño rodeo que simplifica su explicación y nos sirve de excusa para introducir las puertas cuánticas. Como su propio nombre indica, estas puertas no son más que la contrapartida de las puertas lógicas clásicas: AND, OR, NOT,...

En primer lugar, destacamos los estados $|0\rangle$ y $|1\rangle$ que actúan como base del espacio de configuraciones de dimensión 2. Además de la expresión canónica, existen otras formas de elegir los qubits para una base. Un ejemplo de ello es el de la base de Hadamard definida por

$$|+\rangle := \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle := \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Definición 3.1. *Una puerta cuántica que opera sobre un qubit se representa por una matriz unitaria* $A \in U_{\mathbf{C}}(2)$.

Es decir, muchas de las transformaciones que realizamos durante el primer capítulo son en realidad puertas cuánticas. En lo que resta de capítulo podemos pensar que trabajamos con MT reversibles. El Teorema 1.1 junto con la Proposición 1.1 asegura que el operador asociado es unitario.

La puerta que vamos a mostrar a continuación también lleva el nombre del matemático francés Jacques Hadamard. Esta puerta es de gran importancia, pues nos permite crear configuraciones equiprobables.

Definición 3.2. *La puerta de Hadamard para un qubit viene representada por la matriz unitaria*

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Es interesante estudiar el efecto que tiene la puerta de Hadamard en un estado básico:

$$\mathbf{H}: \ |j\rangle \to \frac{1}{\sqrt{2}} \Big(|0\rangle + (-1)^j |0\rangle \Big).$$

Esto quiere decir, como ya apuntábamos previamente, que aplicar dicha puerta al qubit $|0\rangle$ tiene el siguiente efecto:

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

Supongamos que la MTC posee 2^n estados, que llamaremos n-qubits para seguir con la nomenclatura introducida en este capítulo, y que denotaremos por $|\psi\rangle_n$. Para definir una puerta equivalente a la anterior necesitamos de la siguiente definición.

Definición 3.3. *Sean V y W dos espacios vectoriales de dimensión n y m respectivamente. El producto tensorial de V y W, denotado por $V \otimes W$, es un espacio nm-dimensional cuyos elementos son combinaciones lineales de los símbolos $v \otimes w$ satisfaciendo:*

- $\alpha(v \otimes w) = (\alpha v) \otimes w = v \otimes (\alpha w)$
- $(v_1 + v_2) \otimes w = (v_1 \otimes w) + (v_2 \otimes w)$
- $v \otimes (w_1 + w_2) = (v \otimes w_1) + (v \otimes w_2)$

con $\alpha \in \mathbf{C}$, $v, v_1, v_2 \in V$ *y* $w, w_1, w_2 \in W$.

Además, si tenemos dos operadores lineales A *y* B *definidos sobre* V *y* W *respectivamente, entonces* $(A \otimes B)(v \otimes w) = Av \otimes Bw$ *es un operador lineal sobre* $V \otimes W$.

Siguiendo la notación de Dirac, el producto tensorial de dos vectores se puede expresar como $|\psi\rangle \otimes |\phi\rangle = |\psi\rangle|\phi\rangle = |\psi\phi\rangle$.

En dichas condiciones tiene sentido definir $\mathbf{H}^{\otimes n}$ como el producto de n puertas de Hadamard que producen en el n-qubit $|0\rangle_n$ la siguiente transformación:

$$\mathbf{H}^{\otimes n}|0\rangle_n = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle_n.$$

Usamos la notación $A^{\otimes n}$ para distinguir el producto tensorial del usual. Es importante resaltar que cualquier n-qubit puede representarse como

$$|\phi\rangle_n = |a_{n-1} \cdots a_1 a_0\rangle = \bigotimes_{i=0}^{n-1} |a_i\rangle$$

Pasamos a definir la segunda puerta que vamos a necesitar para el algoritmo de Deutsch-Jozsa.

Definición 3.4. *Sea* $f : \{0,1\}^n \to \{0,1\}^m$ *una función, la puerta oráculo* $\mathbf{O}_f$ *es la transformación unitaria que tiene el siguiente efecto en los estados básicos:*

$$\mathbf{O}_f : |j\rangle_n \otimes |k\rangle_m \mapsto |j\rangle_n \otimes |k \oplus f(j)\rangle_m.$$

donde $\oplus$ *es la operación de disyunción exclusiva (XOR) aplicada bit a bit.*

Existen muchas otras puertas de considerable utilidad, aunque con estas nos es suficiente para abordar el problema que nos ocupa. Para una lista extensa de las puertas más usadas nos remitimos al texto de Chuang y Nielsen [9].

3.2 Algoritmo de Deutsch-Jozsa

El algoritmo que veremos a continuación fue presentado por David Deutsch y Richard Jozsa [6]. Éste sirvió como inspiración para el algoritmo presentado por Daniel

R. Simon, que resuelve el problema que el mismo planteó en [12]. Basándose en dicho trabajo, Peter Shor construyó su rompedor algoritmo cuántico [11] para la factorización en primos tomando tiempo polinomial, con las implicaciones que esto tiene sobre la teoría de la complejidad.

El problema a resolver consiste en dada una función $f : \{0,1\}^n \to \{0,1\}$ que sea constante para todos sus valores o bien equilibrada en 0 y 1 (i.e. la mitad de las entradas devuelven un 0 y la otra mitad un 1), determinar si dicha función es constante o equilibrada, permitiendo el uso de dicha función como caja negra. Este se conoce como el problema de Deutsch, que en su versión clásica requiera $2^{n-1}+1$ evaluaciones de la función f en el peor caso.

En primer lugar, supondremos que la máquina recibe como entrada $|\psi_0\rangle_{n,1} = |0\rangle_n \otimes |1\rangle$, usando esta notación para especificar que se trata del producto tensorial entre un n-qubit y un qubit. Por tanto, trabajamos con una MTC de al menos $n+1$ qubits. El resto del algoritmo consiste en la ejecución de los siguientes pasos:

1. $|\psi_1\rangle_{n,1} \leftarrow \mathbf{H}^{\otimes n+1}\left(|\psi_0\rangle_{n,1}\right)$
2. $|\psi_2\rangle_{n,1} \leftarrow \mathbf{O}_f\left(|\psi_1\rangle_{n,1}\right)$
3. $|\psi_3\rangle_{n,1} \leftarrow \left(\mathbf{H}^{\otimes n} \otimes \mathbf{I}\right)\left(|\psi_2\rangle_{n,1}\right)$
4. $\overline{s} \leftarrow$ medir el n-qubit de $|\psi_3\rangle_{n,1}$,

donde puerta notada por $\mathbf{I}$ representa la asociada a la matriz identidad. Veamos que efectivamente el algoritmo resuelve el problema.

En el primer paso se aplica la puerta de Hadamard al $(n+1)$-qubit de partida. Como ya hemos visto anteriormente, la acción de esta puerta sobre el n-qubit $|0\rangle_n$ es la de construir una superposición equiprobable en los estados básicos. Por otro lado, el qubit $|1\rangle$, que nos es necesario para la posterior aplicación de la puerta oráculo de f se convierte en el qubit $|-\rangle$, obteniendo así el nuevo estado

$$|\psi\rangle_{n,1} = \left(\mathbf{H}^{\otimes n}|0\rangle_n\right) \otimes (\mathbf{H}|1\rangle) = \left(\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle_n\right) \otimes |-\rangle.$$

En el segundo paso, la aplicación del oráculo sobre el estado actual produce las siguientes transformaciones:

$$|\psi_2\rangle_{n,1} = \mathbf{O}_f \left[\left(\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle_n\right) \otimes |-\rangle\right] \tag{3.1}$$

Usando la linealidad en el operador, podemos establecer que

$$
\begin{aligned}
(3.1) &= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \mathbf{O}_f \left(|j\rangle_n \otimes |-\rangle \right) = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \mathbf{O}_f \left(|j\rangle_n \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \\
&= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} \frac{|j\rangle_n \otimes |f(j)\rangle - |j\rangle_n \otimes |1 \oplus f(j)\rangle}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{f(j)} |j\rangle_n \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{f(j)} |j\rangle_n \otimes |-\rangle
\end{aligned}
$$

Tras aplicar la puerta oráculo obtenemos una configuración de estados parecida a la que teníamos previamente. La observación obvia es que ahora aparece nuestra función f desconocida en las amplitudes cambiando el signo de los estados básicos para los que $f(j) = 1$. Alcanzar nuestro objetivo pasará por explotar esta propiedad al aplicar los pasos restantes del algoritmo.

En el último paso de cálculo puro, aplicamos la puerta de Hadamard de dimensión n al n-qubit que contiene la información sobre f, mientras que el qubit adicional quedará invariante. Dado que los cálculos son análogos al primer paso, no los detallaremos en exceso:

$$
\mathbf{H}^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} (-1)^{f(j)} |j\rangle_n \right) = \frac{1}{2^n} \sum_{i=0}^{2^n-1} \left(\sum_{j=0}^{2^n-1} (-1)^{f(j)+j\cdot i} \right) |i\rangle_n
$$

donde $j \cdot k$ representa el producto aplicado bit a bit, visto módulo 2. Esta forma de escribirlo es más concisa y fácilmente generalizable a partir de la definición de la puerta de Hadamard aplicada a un qubit genérico:

$$
\mathbf{H}(|j\rangle) = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} (-1)^{j\cdot i} |i\rangle
$$

Una vez termina el proceso de transformaciones aplicadas a la configuración de entrada, que sólo ha requerido una evaluación de f actuando como oráculo, medimos el n-qubit que contiene la información sobre f. La máquina devolverá uno de los 2^n posibles estados básicos con una distribución de probabilidad dada por las amplitudes, que en nuestro caso tienen la siguiente forma:

$$
\alpha_i = \frac{1}{2^n} \sum_{j=0}^{2^n-1} (-1)^{f(j)+j\cdot i}.
$$

La verdadera importancia del paso 3 aparece ahora, al mirar detalladamente la probabilidad asociada al valor $i = 0$. Sin más que sustituir, obtenemos

$$|\alpha_0|^2 = \left| \frac{1}{2^n} \sum_{j=0}^{2^n-1} (-1)^{f(j)} \right|^2 .$$

Puesto que f devuelve los valores 0 o 1, es directo comprobar que

$$|\alpha_0|^2 = \begin{cases} 1 & \text{si } f \text{ es constante} \\ 0 & \text{si } f \text{ es equilibrada} \end{cases}$$

Por tanto, un simple test que compruebe $|0\rangle_n$ como posible salida del algoritmo, resuelve el problema planteado en esta sección.

Durante el capítulo 2 hicimos incidencia en varias contenciones entre clases de complejidad que aún no se conoce si son estrictas. En particular, este algoritmo induce a pensar que $\mathbf{P} \neq \mathbf{EQP}$, aunque solo confirma que se diferencian bajo la acción de un oráculo.

Además, existen varios herederos del algoritmo de Deutsch-Jozsa. El algoritmo de Simon es un claro ejemplo, junto con el algoritmo de factorización de Shor. Alejado de ellos se encuentra el algoritmo de búsqueda Grover [8], cuyo nombre se debe al informático Lov K. Grover. Gracias a él, se establece cómo aceptar la clase **NP** con una MTC y el uso de un oráculo en tiempo $\mathcal{O}(2^{n/2})$. A su vez, Bennett, Bernstein, Brassard y Vazirani prueban en [3] que la cota es realmente óptima, es decir, la clase $\mathbf{NP} \cap \mathbf{co-NP}$ no puede ser aceptada por una MTC con oráculo elegido aleatoriamente en tiempo $(2^{n/3})$.

Una guía completa y ordenada de los algoritmos nombrados anteriormente se encuentra en [10].

4 Conclusión

Echando la vista atrás, nos hemos adentrado en el mundo de la computación cuántica, centrándonos en construir una máquina que generalice, en la medida de lo posible, la definida en la primera mitad del siglo XX por Alan Turing.

Una vez vistas las propiedades que comparten, avanzamos en busca de las que son distintas. Nuestro objetivo final era, sin duda, emular cualquier máquina de Turing cuántica a un bajo precio.

Asociado a este nuevo modelo se construye una teoría de la complejidad computacional paralela a la clásica. Encontramos dificultades similares a las que plantea el problema $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$, además de abrir nuevos problemas aún sin resolver.

Un ejemplo de la potencia de estas máquinas se observa con el algoritmo de Deutsch-Jozsa. Este resuelve de manera exacta y eficiente un problema especialmente diseñado para que cualquier algoritmo clásico que lo resuelva sea altamente costoso. Eso, por supuesto, no quiere decir que las máquinas de Turing cuánticas sean más potentes. En 2019 la compañía Google anunció que había alcanzado la 'supremacía cuántica' con su nueva máquina, aunque generó controversia sobre las condiciones en las que trabaja y los problemas que puede resolver eficientemente. En general, aún se considera un problema físico abierto en el que muchas empresas tecnológicas están apostando.

Queda patente el interés que suscita el paradigma cuántico a todos los niveles. Un descubrimiento importante en el campo de complejidad computacional tendría gran repercusión en las matemáticas y la ciencia en general. Lo mismo ocurre si hablamos de la creación de nuevos algoritmos eficientes, dado que la mayoría de problemas clásicos aún no tienen una resolución elegante que aproveche el potencial de estas máquinas en sus algoritmos.

Bibliografía

[1] Adleman, L. M., DeMarrais, J., and Huang, M.-D. A. Quantum computability. *SIAM J. Comput. 26*, 5 (Oct. 1997), 1524–1540.

[2] Bennett, C. H. Logical reversibility of computation. *IBM J. Res. Dev. 17*, 6 (Nov. 1973), 525–532.

[3] Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. Strengths and weaknesses of quantum computing. *SIAM J. Comput. 26*, 5 (Oct 1997), 1510–1523.

[4] Bernstein, E., and Vazirani, U. Quantum complexity theory. *SIAM J. Comput. 26*, 5 (Oct. 1997), 1411–1473.

[5] Deutsch, D. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Series A 400*, 1818 (July 1985), 97–117.

[6] Deutsch, D., and Jozsa, R. Rapid Solution of Problems by Quantum Computation. *Proceedings of the Royal Society of London Series A 439*, 1907 (Dec. 1992), 553–558.

[7] Feynman, R. P. Simulating physics with computers. *Int. J. Theor. Phys. 21* (1982), 467–488.

[8] Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1996), STOC '96, Association for Computing Machinery, p. 212–219.

[9] Nielsen, M. A., and Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. Cambridge University Press, USA, 2011.

[10] Ossorio-Castillo, J., and Tornero, J. M. Quantum computing from a mathematical perspective: a description of the quantum circuit model. *arXiv: Quantum Physics* (2018).

[11] Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (USA, 1994), SFCS '94, IEEE Computer Society, p. 124–134.

[12] Simon, D. R. On the power of quantum computation. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (USA, 1994), SFCS '94, IEEE Computer Society, p. 116–123.

[13] Solovay, R., and Yao., A. Manuscript, 1996.