Francesco e Valentina Sisini

Informatica Quantistica

Introduzione con esempi in linguaggio C

Prima edizione

Edizioni: i Sisini Pazzi, 2020

Edizione: I Sisini Pazzi

Questo testo è dedicato a Carlo Ferrario (1949-2015), relatore della mia tesi di Laurea.
Per la sua lucidità di pensiero e chiarezza espositiva è stato un punto di riferimento per diverse generazioni di studenti e studentesse che si sono formati presso l'università degli studi di Ferrara.
Per negligenza, non l'ho ringraziato allora. Lo faccio adesso.
Francesco

Scuola Sisini ringrazia calorosamente Lauro Galtarossa per la rilettura, non specializzata ma condotta in forma amichevole, del testo.

Ai nostri lettori

Scuola Sisini è una piccola casa di produzione di testi e giochi realizzati con lo scopo di divulgare argomenti complessi.

Esiste una forbice tra richiesta ed offerta di conoscenza: da un lato articoli scientifici destinati solo ad un pubblico iper specialistico, dall'altro la divulgazione nozionistica di contenuti affascinanti, descritti però qualitativamente.

Tra questi due estremi si inserisce il metodo di divulgazione di Scuola Sisini che, attraverso un percorso ragionato, porta ad uscire dalla propria zona di comfort per far propri gli strumenti e le basi che permettono poi di approfondire autonomamente gli argomenti di interesse.

Indice

Introduzione

L'informatica quantistica si è impegnata in una promessa importante: usare i processi della fisica quantistica per codificare, elaborare e trasmettere informazioni.

Rispetto alle tecnologie informatiche *classiche*, la scienza quantistica propone tre importanti novità non previste nella *fisica/informatica classiche* perché basate sul concetto quantistico di sovrapposizione degli stati ed in particolare su quello di entanglement:

- principio di non clonazione
- teletrasporto quantistico (dell'informazione)
- dense coding (codificazione a maggior densità)

La sfida è duplice: da un lato creare un sistema di trasmissione delle informazioni sicuro e a prova di qualsiasi intrusione, dall'altro scrivere algoritmi che risolvano problemi che attualmente richiedono anni di computazioni anche ai super computer di ultima generazione.
Da sola, la promessa di questi risultati è già sufficiente a far *entusiasmare* le grosse compagnie informatiche del momento e a spingerle ad investire ingenti capitali. Ma l'aspetto realmente affascinante della computazione quantistica, che per tanti darà finalmente un senso allo studio della termodinamica compiuto alle scuole superiori o all'università, è che essa è reversibile e, in linea di principio, è eco sostenibile: potrebbe non consumare energia.

Come è organizzato questo libro

Questo testo propone un percorso figurato attraverso idee, concetti, nozioni e dimostrazioni. Raggiungere la fine del percorso è senz'altro importante, ma non bisogna trascurare il valore del percorso in sé che ha l'obiettivo di creare

collegamenti e connessioni tra argomenti diversi che qui si fondono in un filo conduttore che porta infine alla capacità di comprendere, e poi di padroneggiare, la nuova scienza informatica che da più di quarant'anni bussa alle porte e che per ora è stata accessibile solo a pochi.
Per stimolare l'interesse, la curiosità e facilitare la comprensione, anche intuitiva, di questo argomento ancora nuovo, il percorso proposto non sarà sempre lineare, ma arricchito di esempi e anticipazioni che hanno lo scopo di introdurre gradualmente gli argomenti lasciando il tempo necessario affinché essi siano assimilati prima di venire trattati con gli strumenti formali necessari. Si potranno quindi incontrare concetti e termini che vengono inizialmente solo accennati per essere poi ripresi e sviluppati nel seguito del testo.

Questo testo è nato per essere letto in modo completo ed, essendo un'introduzione, predilige la scorrevolezza della lettura allo sviluppo esaustivo degli argomenti trattati.

- Nel capitolo 1 vengono introdotte le prime idee della meccanica quantistica e si percorre rapidamente lo sviluppo del pensiero che ha portato all'ipotesi di poter costruire un computer quantistico.
- Nel capitolo 2 si introducono in modo *informale* i concetti di qubits e di sovrapposizione degli stati. Scopo del capitolo è portare l'attenzione sui veri obiettivi del libro creando l'interesse per proseguirne la lettura e superare il capitolo 3.
- Il capitolo 3 è il più complesso dell'intero testo, ma, dopo averlo compreso, si avranno tutti gli strumenti per capire l'informatica quantistica ad ogni livello. Qui vengono introdotti i principi base della meccanica quantistica necessari per comprendere a pieno il senso del resto della lettura. I principi vengono trattati anche matematicamente.

Si raccomanda la lettura dell'intero capitolo.

- La relazione tra reversibilità della computazione e la meccanica quantistica è trattata nel capitolo 4. Questo capitolo rappresenta il cuore dell'argomento e, senza eccedere nei formalismi matematici, presenta un aspetto affascinante delle tecnologie quantistiche emergenti.
- Nel capitolo 5 vengono presentati i gates quantistici. Si concretizzano i concetti visti nel capitolo 3 in componenti circuitali che sono capaci di trasformare lo stato dei qubits da 1 a 0 e da 0 ad 1, analogamente a quanto fanno le porte logiche (logics gates) con i bits *classici*.
- Il principio di non clonazione, il teletrasporto e il dense coding sono presentati e spiegati in dettaglio nel capitolo 6. Fondamenti su cui si basa la potenzialità dell'informatica quantistica, capiti i quali, saranno acquisiti gli strumenti intellettuali per entrare consapevolmente nello studio e nella pratica della computazione quantistica.
- Nel capitolo 7 vengono proposti una serie di esercizi in C da eseguire per fissare le idee sui concetti appresi.
  Al termine del capitolo sono presentate le soluzioni dei problemi proposti.
- Nel capitolo 8 viene presentato un esempio di programmazione quantistica usando il linguaggio QASM.

A chi è rivolto questo testo?

Questo libro può essere letto in chiavi diverse a seconda della propria preparazione di base.

Completamente digiuno di fisica e di informatica

Lo yoga ci insegna che il primo movimento della respirazione deve essere l'espirazione, perché nei polmoni serve spazio prima di inspirare nuovo ossigeno. Quindi, se vi trovate nella situazione di non sapere nulla, non disperate, perché allora c'è

posto per il nuovo.

Sazio di informatica ma digiuno di meccanica quantistica

Se la teoria di Shannon e il concetto di entropia dell'informazione sono già note, la loro applicazione nell'ambito quantistico sarà frutto di sorpresa e soddisfazione. In questo caso potrete apprezzare come concetti già noti possono essere rivisitati in un'ottica completamente nuova.

Sazio di fisica ma digiuno di informatica

È difficile completare il corso di studi in fisica senza aver scritto qualche riga di codice, ma si sa che la programmazione è solo una piccola fetta dell'informatica e quindi non disperate perché troverete nel testo le informazioni essenziali per applicare le vostre conoscenze di meccanica quantistica alla teoria informatica e, dove il testo deve glissare, troverete i giusti riferimenti per completare lo studio in modo indipendente.

Esperto di fisica e di informatica

La fisica e l'informatica si sono trovate già a stretto contatto nei primi anni '40. Ora, di nuovo, è la fisica che ha la responsabilità di guidare lo sviluppo tecnologico fino a rendere l'informatica quantistica una realtà ingegnerizzata. Questo testo ha l'ambizione di fornire una prima guida introduttiva per mettere in relazione la meccanica quantistica e la teoria dell'informazione.

Convenzioni tipografiche

Nel testo sono inseriti dei **Box** che contengono degli approfondimenti sull'argomento trattato nel paragrafo. I box sono segnalati dalla scritta **Box**, incorniciati e con lo sfondo

giallo.

Nel testo si è fatto uso del **grassetto** per evidenziare i termini chiave in una frase e del *corsivo* per le parole che hanno un significato specifico nel contesto del discorso.
Per esempio:

> La meccanica quantistica inserisce all'interno della teoria anche il concetto di **misura**. Si deve notare che anche la teoria classica prevedeva le misure sperimentali...
>
> Per programmare un computer *classico* è necessario prendere confidenza con la tecnologia: tastiera, monitor, editor ecc., e apprendere il concetto di *linguaggio formale* con il quale è possibile descrivere un algoritmo.

Pagina web del libro

Sul sito `https://pumar.it` è presente la pagina web del libro (`https://pumar.it/libroQuantistica.php`)
Sulla pagina si possono trovare diversi link di interesse e il link con gli **errata corrige** che emergeranno nel corso del tempo.
Sono state programmate una serie di revisioni del testo con scadenza mensile, gli eventuali errori rilevati saranno riportati e corretti.cI codici C presentati nel capitolo 7 sono disponibili all'indirizzo:
`https://github.com/francescosisini/LIBRO-Informatica-Quantistica`
Gli argomenti sono inoltre trattati sul canale di ScuolaSisini: https://www.youtube.com/channel/UCDwLlFqa0xZ71PEOdHA0aSQ

Premessa

Questo testo è una prima introduzione ai concetti e ai principi dell'informatica quantistica. Qui sono presentati e spiegati tutti gli strumenti matematici, fisici e informatici necessari per una comprensione completa e consapevole. Il testo è auto consistente e, per una prima lettura, non richiede l'ausilio di altri testi specifici. Per mettere in pratica i concetti presentati alla fine del libro sono inseriti in un capitolo apposito degli esercizi di programmazione in linguaggio C. Il capitolo può essere saltato senza pregiudicare la comprensione del resto.

1. Punto di partenza

La fisica è una scienza impegnativa perché non si accontenta di risposte verosimili, ma pretende una continua verifica sperimentale di ogni affermazione che viene fatta in suo nome. Nonostante questo obiettivo è ironico sapere che ogni teoria è probabilmente falsa, e quelle che crediamo vere, sono in attesa di essere dimostrate false o appunto *falsificate*.
Con questo, si deve rinunciare al proposito di studiarle? No certo, perché le teorie della fisica, entro certi limiti, funzionano.
Nel XX secolo è stato dimostrato che la teoria della gravitazione universale è falsa, ciò nonostante, essa è ancora molto utile per numerosi calcoli di balistica spaziale e per i conti più *grossolani* di astronomia del sistema solare.
La teoria di Newton sulla gravitazione è stata sostituita da quella di Einstein della relatività generale. Negli ultimi decenni però anche quest'ultima ha dato segni di cedimento e in futuro potrebbe essere messa in discussione.

Le leggi che la fisica identifica come leggi fondamentali costituiscono la *meccanica* del nostro universo.
Il primo incontro *scolastico* che si ha con la meccanica è la *cinematica*. Questa descrive le relazioni tra posizione e velocità di un *punto materiale*. Il concetto di punto materiale è un concetto *classico* che parte dall'idea che la materia possa essere pensata in modo analogo alla geometria euclidea, dove i solidi hanno superficie e volume definiti, ma sono costituiti da insiemi di punti ognuno dei quali ha dimensione nulla.
Allo stesso modo il punto materiale ha dimensione nulla, ma ad esso è associata una massa.
Lo stato del punto materiale è completamente descritto quando sono definite le sue tre coordinate spaziali x, y e z e le tre componenti cartesiane della sua velocità: v_x, v_y e v_z.
La posizione e la velocità del punto materiale possono variare con il tempo e per questo si specifica che sia le coordinate che

la velocità sono funzioni di quest'ultimo: $x(t)$, $y(t)$ e $z(t)$ e $v_x(t)$, $v_y(t)$ e $v_z(t)$.

La meccanica quantistica trova il suo primo **disaccordo** con la meccanica classica già qui: secondo la meccanica quantistica non è possibile che ad un dato istante di tempo, per una particella, si conoscano **esattamente** la sua posizione e la sua velocità. Questo è noto come *principio di indeterminazione di Heisenberg*.

Come è stato chiarito all'inizio del paragrafo, la meccanica quantistica si discosta notevolmente dalla meccanica classica, ma non è definibile senza di essa.
Nella meccanica classica un sistema fisico si ritiene completamente definito quando siano note tutte le posizioni dei punti che lo compongono, tutte le velocità e le eventuali forze che stanno agendo. Ovviamente non può essere lo stesso per un sistema quanto-meccanico, visto che queste grandezze non possono essere note simultaneamente con precisione arbitraria.
La meccanica quantistica inserisce all'interno della teoria anche il concetto di **misura**. Si deve notare che anche la teoria classica prevedeva le misure sperimentali, anzi queste fanno parte del metodo sperimentale con cui la teoria è stata costruita, ma nella meccanica classica tali misure sono intese come esterne alla teoria.

La differenza tra i due approcci è che nel *classico* il processo di misura è visto come un'azione necessaria all'osservatore dell'esperimento per conoscere empiricamente il valore di una grandezza che sta osservando, ma che detta grandezza abbia un ben dato valore indipendente dal processo di misura. Nel *quantistico* invece, secondo l'interpretazione della scuola di Copenaghen, le grandezze fisiche di un dato sistema non hanno sempre valori definiti, ma li assumono nel momento stesso in cui accade un evento *speciale*, come appunto una misura.

A questo proposito è d'obbligo chiarire subito per evitare malintesi. Il concetto di **misura** in meccanica quantistica non prevede la presenza di un essere intelligente: con misura si intende (per esempio) l'interazione di un sistema molto piccolo, cioè di dimensioni atomiche, sub-atomiche o particellare con un sistema classico, cioè di grande massa. Non si pensi che grande massa significhi la massa di un edificio, basta prendere venti grammi di acqua per avere (circa) un numero di Avogadro $\left(6 \times 10^{23}\right)$ di molecole, quindi il concetto di grande massa va rapportato alla scala atomica (Landau, 1947; Bohr 1928).

Questo aspetto della teoria è difficile da capire perché si scontra contro l'esperienza quotidiana che per l'appunto è un'esperienza *classica* della fisica.

Domanda n. 1 Il principio di indeterminazione di Heisemberg riguarda:

a. L'impossibilità di misurare simultaneamente e con precisione arbitraria la posizione e la quantità di moto di una particella
b. L'impossibilità di predire correttamente l'evoluzione dello stato di un sistema fisico

1.1 Mondo classico e mondo quantistico: la doppia fenditura

Se lasciamo cadere un oggetto dalla mano, durante la sua caduta, siamo convinti che esso abbia una posizione ed una velocità ben definite.
Non importa se lo stiamo osservando o meno: secondo il senso comune l'oggetto si trova sempre in uno stato fisico ben definito.
Le cose stanno diversamente secondo la meccanica quantistica

(Tonomura, 1989). Per fare un esempio famigliare, consideriamo un vecchio televisore a tubi catodici e proviamo a seguire il percorso di un elettrone che fuoriesce dal catodo. Supponiamo di porre tra il catodo e lo schermo televisivo una targhetta in metallo con due fenditure vicine, in modo che se l'elettrone non azzecca una delle due venga fermato dalla targhetta.
Dopo aver acceso la TV, vederemo comparire dei puntini luminosi sullo schermo secondo una data disposizione, e penseremo:

ogni puntino sullo schermo corrisponde ad un elettrone che è riuscito ad attraversare una fenditura

Nel nostro pensiero, chè segue l'intuizione classica, un elettrone che è riuscito a passare deve aver attraversato l'una o l'altra fenditura della targhetta.
Secondo la meccanica quantistica, invece, le cose non sono andate esattamente così.
Se durante il moto dell'elettrone non è stato condotto alcun esperimento per misurare la sua posizione, allora non si può dire se l'elettrone abbia seguito una specifica traiettoria e neanche se sia passato dall'una o dall'altra fenditura, ma si può dire solo che:
se è arrivato sullo schermo, allora è passato per l'una e/o per l'altra fenditura.

A prima vista questa considerazione può sembrare priva di interesse pratico, quasi un sofisma, ma dal 1980 in avanti ha assunto un ruolo chiave nella tecnologia futura, cioè da quando il fisico Richard Feynman ha pensato che la meccanica quantistica poteva essere la base per la costruzione di un nuovo modello di calcolo, differente da quello sviluppato da Alan Turing (Feynman, 1982).
Questo aspetto della natura è chiamato **principio di sovrapposizione degli stati**.

Da esso discendono un teorema e due importanti applicazioni dell'informatica quantistica: il teorema del **no cloning principle** e le due applicazioni **dense coding** e **quantum teleportation**, che saranno discusse dettagliatamente più avanti nel testo.

1.2 Si deve imparare la meccanica quantistica per comprendere la computazione quantistica?

Per comprendere la meccanica quantistica servono almeno tre anni di studio: uno dedicato alle basi matematiche, partendo dal calcolo differenziale in una sola dimensione e sul campo dei numeri reali, per arrivare al calcolo sul campo dei numeri complessi; uno dedicato allo studio della meccanica hamiltoniana ed uno alla meccanica quantistica, teoria non relativistica e teoria relativistica.
Questo studio **non può** essere condotto dopo cena come passatempo, ma deve impegnare il giorno intero.

Per fortuna si possono comprendere i principi dell'informatica quantistica e la base del funzionamento di un computer quantistico anche senza conoscere tutta la meccanica quantistica.
La situazione presente non è diversa dallo scenario in cui furono presentati i primi computer nel secolo scorso.
Le prime macchine erano la sintesi perfetta dello stato dell'arte della logica matematica e dell'elettronica, che allora era una scienza su cui avevano competenza solo i fisici perché i corsi di ingegneria elettronica ancora non esistevano.
Negli anni '50 del XX secolo per spiegare il funzionamento di un computer era necessario un fisico esperto e per scrivere un programma serviva comunque un matematico o un ingegnere.
Il concetto di memoria volatile era assolutamente nuovo e si basava su tecnologie al limite del realizzabile, per cui era difficile separare l'idea astratta di **automa** dalla sua concreta

realizzazione.
Come risultato di questo scenario la programmazione di un computer appariva possibile solo ad esperti fisici, matematici ed ingegneri.
Anche senza correre troppo indietro negli anni, basta pensare al film "War Games" di John Badham (1983). Il film racconta la storia di un giovane hacker che si trova ad affrontare un'intelligenza artificiale che scambia la guerra mondiale per un gioco di strategia.
È interessante notare che il programma di intelligenza artificiale venga presentato come il frutto del lavoro di un unico scienziato: Stephen Falken.
L'idea dello *scienziato* capace da solo di realizzare *un mostro* come quello del dottor Frankenstein, era un retaggio della prima informatica degli anni '50.

Dalla visione di un'informatica lontana ed *aliena* si è passati in pochi decenni a linguaggi di programmazione come *scratch* pensati per introdurre la programmazione già dalle scuole elementari.
Oggi è possibile formare un programmatore in pochi mesi senza che egli conosca nulla di elettronica.
Per programmare un computer classico è necessario prendere confidenza con la tecnologia: tastiera, monitor, editor ecc., e apprendere il concetto di *linguaggio formale* con il quale è possibile descrivere un algoritmo, anche senza sapere come funziona effettivamente un computer.

Quanto appena detto per la programmazione di computer classici, vale anche per i computer quantistici. Se si conosce la meccanica quantistica si può capire a fondo come funzioni la loro tecnologia, ma la comprensione approfondita di questa non è necessaria per formulare un algoritmo quantistico.
Il messaggio quindi è il seguente: con un certo sforzo, chiunque si applichi, può comprendere e applicare la computazione

quantistica e, se studierà in modo completo la meccanica quantistica, comprenderà anche come funziona un computer quantistico.
Non mi sembra nulla di diverso dallo scenario attuale.

In questo testo svilupperemo diversi esempi di programmi che simuleranno alcuni aspetti di un computer quantistico. Va da sé che la simulazione non potrà riprodurre le prestazioni in termini di velocità.

Domanda n. 2 Quali sono i tre aspetti peculiari della informatica quantistica di cui si è accennato?:

a. Dense coding, quantum teleportation, indeterminazione degli stati
b. Dense coding, quantum teleportation, no cloning principle
c. Dense coding, no cloning principle, sovrapposizione degli stati

1.3 Simulare la fisica sul computer: si può fare Mr. Feynman?

Il titolo di questo paragrafo nasce dalla parafrasi del titolo del libro "Sta scherzando Mr. Feynman?" di Ralph Leighton, amico di Feynman, che raccolse diversi aneddoti sullo scienziato.
Richard Feynman era un fisico con una ricca carriera accademica che aveva lavorato con J. von Neumann a metà degli anni '40 e vide nascere i primi computer *classici*.

L'idea di un modello di computazione basato sulla meccanica quantistica risale almeno al 1980, proprio per opera di Feynman, che presentò la sua idea nel 1981 con un articolo intitolato *Simulating physics with computers*, pubblicato sulla rivista International Journal of Theoretical Physics, nel quale

poneva una domanda precisa:

È possibile simulare la fisica quantistica usando un computer?

Nell'articolo, Feynman spiegava le ragioni per cui i computer classici erano adatti a simulare la fisica classica, ma non la fisica quantistica.

È interessante notare l'influenza che von Neumann deve avere avuto sull'allora giovane Feynman, nell'articolo, infatti, Feynman si riferisce ai computer sempre in termini di automi cellulari, idea nata appunto dalla mente di von Neumann mentre cercava di progettare un modello di automa che potesse *auto riprodursi*.

L'argomentazione usata da Feynman, di seguito esposta, è importante perché ci introduce direttamente ai principi di meccanica quantistica che ci sono necessari per sviluppare il resto del testo, in modo particolare al principio di *località* della fisica classica.

La fisica classica è locale

La fisica classica è una *teoria locale*, cioè: l'interazione tra due *entità fisiche* avviene solo quando sono a diretto contatto.
Questa affermazione potrebbe sembrare in contrasto con i fenomeni elettromagnetici ed elettrostatici: pensiamo per esempio a due punti materiali che esercitano l'uno sull'altro una forza coulombiana.
In realtà anche l'elettromagnetismo è una teoria locale e infatti l'azione a distanza tra due cariche è dovuta al *campo di forze* indotto da ogni carica che agisce sull'altra carica, quindi l'azione su ogni carica è comunque sempre **locale** e quindi dipende solo dal valore del campo nel punto in cui si trova la carica.

Per chiarire bene il proprio punto di vista Feynman fece riferimento al modello computazionale dell'**automa cellulare**.
A differenza della più nota *architettura di von Neumann*, dovuta naturalmente a von Neumann, l'automa cellulare ha la caratteristica che ogni bit cambia il proprio valore in base al valore che hanno i bit adiacenti, quindi l'*interazione è locale*. In realtà anche i computer odierni che si basano sull'architettura di von Neumann sono *locali*, ma nell'automa cellulare questa *località* è resa più evidente, e per questo Feynman preferì riferirsi ad essa.

Box: Modello di calcolo dell'automa cellulare
Questo modello è stato introdotto da von Neumann che, studiando i primi modelli di intelligenza artificiale nel campo informatico, pensò all'automa cellulare per realizzare una forma di automa che potesse auto riprodursi.
La particolarità dell'automa cellulare è che la computazione avviene in modo *parallelo* e puntuale o, appunto, **locale**. L'automa infatti consiste in:

- Una stringa formata di bit che possono essere nello stato 0 oppure 1. Ogni bit è detto cella.
- Una regola che stabilisce come ogni cella cambi di stato in base al valore delle celle vicine.

Il sistema di calcolo è molto semplice. All'istante $t = 0$ l'automa si trova in una certa configurazione iniziale. All'istante $t = 1$, per ogni cella, si valutano i due bit a destra e a sinistra della cella e, in base alla regola stabilita, si assegna il nuovo valore del bit per la configurazione dell'automa a $t = 1$.

Si ripete la procedura per i t successivi.

Per fare un esempio di si consideri la seguente configurazione:

- Configurazione iniziale: 11011011
- Regola:
  - $1\mathbf{1}1 \rightarrow \mathbf{0}$
  - $1\mathbf{1}0 \rightarrow \mathbf{1}$
  - $1\mathbf{0}1 \rightarrow \mathbf{1}$
  - $1\mathbf{0}0 \rightarrow \mathbf{0}$
  - $0\mathbf{1}1 \rightarrow \mathbf{1}$
  - $0\mathbf{1}0 \rightarrow \mathbf{1}$
  - $0\mathbf{0}1 \rightarrow \mathbf{1}$
  - $0\mathbf{0}0 \rightarrow \mathbf{0}$

La computazione dei primi 4 passi è rappresentata nella tabella seguente.

$t = 0$	(1)	1	1	0	1	1	0	1	1	(1)
$t = 1$	(0)	0	1	1	1	1	1	1	0	(0)
$t = 2$	(0)	1	1	0	0	0	0	1	0	(1)
$t = 3$	(1)	1	1	0	0	0	1	1	1	(1)

I bit fra parentesi sono usati per il *padding,* cioè per poter racchiudere tra due celle anche le celle dei bordi. In questo esempio il padding è ottenuto per *circolarità*: in pratica è come se la stringa di bit si chiudesse su sé stessa. Cambiando le regole è possibile definire diversi tipi di automi che possono anche essere definiti su una griglia a due dimensioni. Quello qui mostrato è noto con il nome di **Rule 110**: un automa Turing completo che in

linea di principio può essere usato per eseguire un qualsiasi programma che può essere eseguito su un calcolatore ordinario.

Tornando al punto, Feynman illustra come sfruttando la *località* del computer sia possibile **simulare** la fisica classica ma, al contrario, **non** sia possibile simulare la fisica regolata dalla meccanica quantistica.

Sia chiaro che in un computer classico può essere eseguita l'evoluzione di un sistema quantistico: se si conosce la dinamica di un sistema quantistico, cioè le equazioni che descrivono la sua evoluzione nel tempo (l'analogo delle leggi orarie delle coordinate nella fisica classica), è assolutamente possibile **emularlo**, ma non simularlo, perché la meccanica quantistica non è locale. Questo argomento è molto complesso e può dar luogo a fraintendimenti, infatti la non-località della meccanica quantistica non significa che per essa sia ammessa un'azione a distanza, ma che sia possibile riscontrare una correlazione tra misure sperimentali che non è spiegabile da nessuna teoria locale. La non località della meccanica quantistica è legata al concetto di entanglement che viene discusso nei prossimi capitoli.

È importante avere presente questo argomento perché è stato uno delle motivazioni principali che ha spinto Feynman e altri insieme e dopo di lui ad indagare la possibilità di realizzare una macchina quantistica che permettesse di simulare la fisica quantistica come i computer ordinari permettono di simulare la fisica classica.

Domanda n. 3 In base al principio di *località* della fisica classica, come si spiegano i fenomeni elettrostatici?:

a. Si spiegano con la presenza di fotoni virtuali, quindi con la meccanica quantistica
b. Inserendo il campo elettrico come una entità fisica
c. Sono una eccezione alla teoria

2. Dai bits *classici* ai qubits quantistici

In questo capitolo vengono esposti i principi della meccanica quantistica che devono essere appresi e *digeriti* per continuare con successo la lettura del testo.
Invece di formulare i principi e i postulati con l'ambizione di descrivere l'intero mondo fisico, verranno formulati con l'obiettivo di descrivere ciò che serve a comprendere la computazione quantistica.
In questo capitolo introdurremo questi concetti in modo informale, cercando di abituarci alle nuove idee che devono essere apprese. Nel capitolo successivo daremo una visione più formale e compatta degli stessi concetti.

2.1 I bits nell'informatica classica

Un bit è l'unità di informazione usata in informatica classica. Un bit può assumere solo due valori: l'1 e lo 0. Usando più bit si possono rappresentare delle informazioni complesse, per esempio il numero decimale 11 può essere rappresentato in forma binaria come la sequenza di bit 1011 che sviluppata sulle potenze del due risulta esattamente il numero decimale 11 (i.e. $1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 1 + 2 + 8$).

Un bit può essere *realizzato* con diverse tecnologie. In generale si tratta di creare un **sistema fisico** che ammetta due possibili **stati** di equilibrio. Chiameremo questo sistema: **sistema fisico classico** dove l'aggettivo classico si riferisce alla possibilità di descrivere la fisica del sistema usando le leggi della meccanica classica (Newton, per intenderci).

Per esempio si può pensare ad un bit realizzato da una matita in cui il valore 0 è associato allo **stato** in cui la matita è in posizione verticale, e il valore 1 allo **stato** in cui la matita è in orizzontale.
Lo stato *verticale* della matita può essere espresso

matematicamente indicando con θ l'angolo tra la matita e il piano (per esempio il piano del tavolo). Se θ è uguale a $\frac{\pi}{2}$ allora il bit è 0, se θ è zero allora il bit è 1.
In pratica possiamo associare il valore *binario* del bit allo stato θ del sistema fisico che, nel caso in questione, è una semplice matita.

Ci sono diversi altri modi per realizzare un bit, per esempio usando un interruttore collegato ad una luce led, e associando alla luce accesa il valore 1 e alla luce spenta lo 0.
Ovviamente per condensare molti giga bits in uno spazio limitato non si possono usare matite o led, ma si usano altre tecnologie come quelle presenti nei moderni calcolatori.

2.2 Proprietà dei bits

In questo paragrafo ci soffermiamo su alcune osservazioni che possono sembrare ovvie, ma che acquisiranno interesse nel momento in cui le confronteremo con osservazioni analoghe compiute sui **qubit** cioè i bit quantistici.
Per semplicità continuiamo a considerare i bit identificati dallo stato θ della matita, ma le considerazione che trarremo valgono in generale per qualsiasi tecnologia *classica* che si possa usare per realizzarli.

Relazione tra lo stato di un bit e il suo valore

Lo stato di un *bit-matita* è sempre perfettamente determinato.
La matita può stare solo nello stato $\theta = \frac{\pi}{2}$ o nello stato $\theta = 0$ e, a causa della forza di gravità, non è possibile che si trovi in equilibrio uno stato intermedio.
Per conoscere il valore del bit è sufficiente conoscere il suo angolo rispetto al piano.

Se un osservatore, sulla Terra, si trovasse ruotato rispetto al piano di un angolo α vedrebbe il *bit-matita* formare un angolo

diverso dai due angoli possibili appena definiti: dal punto di vista dell'osservatore, infatti, la matita si troverebbe ad uno dei due angoli $\frac{\pi}{2} + \alpha$ o $0 + \alpha$.

Questa rotazione non impedirebbe di ricondurre la posizione del *bit-matita* al corretto valore del bit, infatti, l'osservatore saprebbe di essere ruotato perché potrebbe verificarlo con un esperimento legato alla forza gravitazionale: per esempio osservando la direzione in cui gli oggetti cadono a terra. In tal modo potrebbe sempre ricondurre la propria misurazione dell'angolo formato dalla matita nel suo sistema di riferimento, con l'angolo θ usato per definire lo stato della matita.

Quindi, per i bit *classici*, possiamo affermare che:

Lo stato di un bit è associato sempre ad uno ed un solo valore: 0 oppure 1.

Modifica dello stato di un bit

Lo stato di un *bit-matita* può essere modificato solo in seguito ad una azione volontaria che lo porti dallo stato $\theta = 0$ a $\theta = \frac{\pi}{2}$ oppure da $\theta = \frac{\pi}{2}$ a $\theta = 0$.

Domanda n. 4 In linea di principio, per un computer *classico*, il risultato della lettura di un bit dalla memoria è prevedibile?:

a. Si, fatta eccezione per possibili problemi hardware del dispositivo
b. No, è un fenomeno probabilistico
c. Si, ma solo se si è eseguita una copia di back-up

Copia dello stato di un bit

Lo stato di un bit è sempre noto ed è possibile eseguirne una

copia senza modificare l'originale.
Questa caratteristica permette molte operazioni utili come la copia di un documento da condividere o da inviare per email, e altrettante delicate o rischiose dal punto di vista della sicurezza informatica come per esempio un intruso che può leggere un messaggio scambiato tra due soggetti comunicanti senza che loro se ne accorgano intercettando la comunicazione e copiandone il contenuto bit a bit, senza compromettere il messaggio trasmesso.

Stato di più bits

Il bit è l'unità minima di informazione dell'informatica (discreta). Per rappresentare delle grandezze complesse, come ad esempio per codificare un alfabeto o le cifre decimali, un singolo bit non è sufficiente e quindi se ne devono usare un certo insieme in cui ogni bit a seconda della sua posizione abbia un significato preciso, come visto nell'esempio della trasformazione della sequenza binaria 1101 nel numero decimale 11.

Consideriamo per esempio un insieme di otto *bit-matita* (byte) disposti alcuni nello stato $\theta = 0$ e altri nello stato $\theta = \frac{\pi}{2}$.
Per riferirci in modo chiaro allo stato di ognuna delle matite useremo l'indice i. Ad esempio scrivendo θ_0 indicheremo lo stato della prima matita, mentre scrivendo θ_7 quello dell'ottava.
Usando queste otto matite creiamo un nuovo sistema fisico che ammette 2^8 configurazioni diverse e che corrispondono ad una precisa sequenza dei valori θ_i.
Per esempio lo stato $\begin{pmatrix} \pi/2 & 0 & 0 & 0 & 0 & 0 & 0 & \pi/2 \end{pmatrix}$ corrisponde ai bits $\begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$.
La corrispondenza individuata è di natura biunivoca e quindi alle 256 diverse configurazioni corrispondono esattamente 256 diverse informazioni codificate dagli otto bits, non di più e non di meno.

Abbiamo portato volutamente l'attenzione sulla relazione tra l'unità di informazione (1 e 0) e il concetto di sistema fisico classico, perché nel prossimo paragrafo riprendiamo questa relazione, applicandola però al concetto di sistema fisico **quantistico**.

2.3 I qubits

Ci chiediamo ora se esiste una differenza tra sistemi fisici classici e sistemi quantistici.
A rigor di logica dovremmo dire che la teoria quantistica è considerata la meccanica che regola ogni sistema fisico esistente, quindi esistono solo sistemi quantistici.
In realtà è prassi indicare come quantistici solo quei sistemi che non possono essere spiegati con sufficiente accuratezza dalla meccanica classica.
Per esempio, per descrivere il classico moto di una palla di cannone, lanciata ad un angolo θ di inclinazione, non abbiamo bisogno della meccanica quantistica, e chiamiamo un sistema siffatto *classico*.
Chiamiamo quindi sistemi quantistici quei sistemi fisici che richiedono la meccanica quantistica per essere descritti correttamente.
Sistemi come gli atomi e le particelle elementari hanno sempre natura quantistica, ma anche certi sistemi *macroscopici*, normalmente non presenti in natura sulla Terra, ma ottenuti per via artificiale, possono essere sistemi quantistici se il loro comportamento non è descrivibile dalla fisica classica e richiede l'uso della meccanica quantistica.

Realizzare un esperimento per misurare un sistema quantistico non è semplice come realizzarlo per un sistema classico.

Gli esperimenti per acquisire delle misure classiche, come ad esempio la posizione di un oggetto o il tempo di percorrenza di

un carrello lungo un binario, sono facilmente realizzabili, e per questo fanno parte del bagaglio di studio ed esperienza che si acquisisce già frequentando la scuola.
Per questo il concetto di **stato** introdotto con l'esempio della matita è diretto ed intuitivo, in quanto rientra nelle esperienza di tutti.

Il bit è una associazione tra uno stato fisico classico e uno dei due valori numerici 0 e 1.
Analogamente, il qubit è un'associazione tra uno stato fisico quantistico e uno dei due valori numerici 0 e 1. La differenza esistente tra stato fisico classico e quello quantistico è ciò che determina la differenza tra l'informatica classica e quella quantistica.

Il concetto di **stato** che definiremo per i sistemi fisici quantistici non è invece altrettanto intuitivo di quello classico perché, nella vita comune, mancano le occasioni per familiarizzare con esperimenti di misura quantistica.

Per definire i qubits, nel seguito di questo capitolo, introdurremo le particelle chiamate **fotoni** perché useremo la misura del loro stato fisico associandogli il valore binario 0 o 1.

Domanda n. 5 Cosa si intende con sistema quantistico?:

a. Un sistema di dimensioni atomiche, cioè paragonabili a quelle degli atomi
b. Qualsiasi sistema, anche macroscopico, che *richieda* di essere descritto con le leggi della meccanica quantistica, come ad esempio certi sistemi macroscopici a temperature prossime agli 0 gradi Kelvin
c. Un sistema che non può essere misurato con strumenti classici

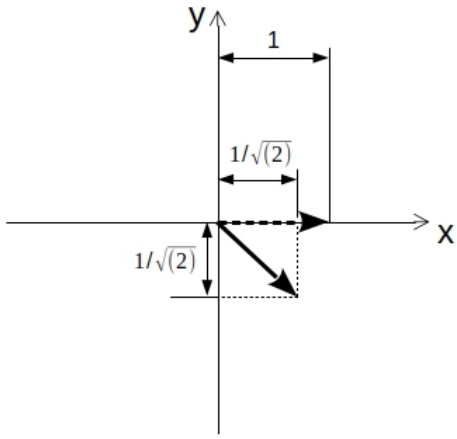

Fig.1 Il vettore ha lunghezza unitaria. Dopo la rotazione di 45 gradi, le sue componenti cartesiane misurano entrambe $\frac{1}{\sqrt{2}}$.

Per scrivere i due stati del fotone rispetto al polaroid ruotato egli si figura di dover ruotare nel piano cartesiano un segmento di lunghezza unitaria con centro nell'origine e associa all'asse x lo stato $|0\rangle$ e all'asse y lo stato $|1\rangle$ e scrive la rotazione di 45 gradi in senso orario dello stato $|0\rangle$ di polarizzazione del fotone come:

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

e quella della rotazione di 45 gradi in senso orario dello stato $|1\rangle$ del fotone come:

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

Il signor Magri postula che la probabilità p che un fotone polarizzato a 45 gradi lungo la diagonale secondaria (come in figura) passi attraverso un polaroid ad asse verticale è data da:

$$\mathbf{p} = \left(\frac{1}{\sqrt{2}} \right)^2 = \frac{1}{2}$$

Il signor Magri non sa spiegarsi il perché, però capisce che c'è una relazione tra la radice di 2 trovata nella rotazione della polarizzazione del fotone e la probabilità del 50% (appunto $\frac{1}{2}$) misurata sperimentalmente, ma essendo un informatico non è interessato alla polarizzazione dei fotoni in sé, ciò che sta catturando la sua attenzione è, invece, la relazione che hanno i qubit con lo stato di polarizzazione dei fotoni ed è questo che lui vuole studiare.
Prima di continuare i suoi studi capisce che le espressioni trovate sono importanti e decide di assegnare dei simboli agli stati da esse individuati. Sceglie i due simboli $|0'\rangle$ e $|1'\rangle$ e li definisce come segue:

$$|\mathbf{0}'\rangle = \left(\frac{1}{\sqrt{2}} \right) (|0\rangle - |1\rangle)$$

e

$$|\mathbf{1}'\rangle = \left(\frac{1}{\sqrt{2}} \right) (|0\rangle + |1\rangle)$$

Si accorge quindi che può riscrivere gli stati |0> e |1> usando gli stati $|0'\rangle$ e $|1'\rangle$. Infatti:

$$|\mathbf{0}\rangle = \left(\frac{1}{\sqrt{2}} \right) (|0'\rangle + |1'\rangle)$$

e

$$|\mathbf{1}\rangle = \left(\frac{1}{\sqrt{2}} \right) (|0'\rangle - |1'\rangle)$$

Il signor Magri capisce che i qubit possono essere rappresentati anche attraverso questi due stati e che quindi questi due stati sono una base per rappresentare i qubit. Il signor Magri ha quindi intuito un concetto importante:

lo stato di polarizzazione di un fotone è esprimibile rispetto ad una base formata da due stati separati e indipendenti
ed esso può anche non coincidere con nessuno dei due stati di base, come gli è successo quando i fotoni con polarizzazione obliqua incidevano sui polaroid ad assi verticale ed orizzontale.
Il signor Magri ha capito un'altra cosa molto importate:
se la polarizzazione del fotone è parallela uno degli assi dei polaroid, allora il risultato della misura è certo, altrimenti, se la polarizzazione del fotone giace su una retta non parallela a nessuno dei due assi, allora il risultato è solo probabilistico.

Differenze tra bits e qubits

La storia del signor Magri e del signor Fabbri, ci ha aperto gli occhi sulle differenze tra bits e qubits.
Entrambi sfruttano lo stato di un sistema fisico per identificare un coppia di valori (1 e 0) e questa è la loro **analogia**. La **differenza** è che per i primi c'è una corrispondenza *biunivoca* tra lo stato e il valore del bit, mentre per i secondi c'è una corrispondenza biunivoca tra la *misura* dello stato e il valore del qubit.

Nel caso classico, lo stato della matita identifica univocamente il valore del bit (verticale 0, orizzontale 1), nel caso dei quantistico stati diversi possono portare probabilisticamente allo stesso valore del bit, quindi non si può stabilire una corrispondenza biunivoca tranne nei casi in cui lo stato di polarizzazione coincide con uno degli assi dei polaroid.
La misura dello stato di polarizzazione, invece, porta sempre ad un risultato che può essere solo uno tra i due possibili e quindi esiste una relazione biunivoca tra la misura dello stato e il valore del qubit.

Questa differenza è molto importante ed è ciò che caratterizza l'informatica quantistica rispetto a quella classica. Quando si sarà finito di leggere il libro, si torni su questo punto fintanto che non risulti completamente chiaro.

Perché questa differenza?

Nella fisica classica, non c'è differenza tra lo stato di un sistema e la sua misura.
Ripensando all'esempio delle matite si può immaginare di adottare la stessa convenzione a bordo di una **stazione spaziale** orbitante. In quel contesto le matite possono assumere angoli arbitrari rispetto al riferimento della stazione quindi, sebbene una coppia di assi rimanga un valido sistema per definirne lo stato, si potrebbe obiettare che analogamente a quanto accade nella meccanica quantistica, se una matita non è allineata lungo uno dei due assi, il suo stato è in una sovrapposizione di stati.
Questo non è falso, ma c'è una differenza importante e sostanziale con il concetto di stato quantistico. Dal punto di vista classico, anche se sulla stazione spaziale la matita può assumere una qualsiasi angolazione rispetto agli assi, rimane una relazione biunivoca tra stato e misura:

la misura dell'angolo che la matita forma con un dato asse può essere eseguita con precisione arbitraria senza modificare lo stato della matita.
Anche su una stazione spaziale si potrebbero quindi usare i *bit-matita* adottando la semplice convenzione che l'angolo da 0 a 45 gradi rappresenti il bit 1 mentre l'angolo da 45 a 90 rappresenti il bit 0.

Per i sistemi quantistici invece:

la relazione tra lo stato del sistema e il risultato della misura ha natura probabilistica
Come abbiamo visto, quando si misura lo stato di polarizzazione di un fotone si possono ottenere solo due valori distinti. Prima della misura, il fotone può essere in un infinità di stati diversi tra loro. Il processo di misura *obbliga* il fotone ad assumere uno dei due stati che per questo sono detti *stati di base*.
Questo aspetto della meccanica quantistica è difficile da digerire ed in effetti esistono anche interpretazioni diverse da questa, detta *interpretazione di Copenaghen*, che offrono una visione alternativa. Quella di Copenaghen è comunque l'interpretazione più diffusa e per questo è detta essere *ortodossa*.

Stato fisico del fotone e valore del qubit

Alcuni testi potrebbero tendere ad unificare il concetto di qubit con quello di stato fisico quantistico della polarizzazione del fotone. Indipendentemente dal modo in cui lo si vuole trattare formalmente, è sempre importante tenere a mente la distinzione esistente tra lo stato fisico prima e dopo l'esecuzione di una misura.

Ora che abbiamo introdotto il concetto di stato quantistico dobbiamo chiarire bene qual'è la relazione **tra stato fisico** e

valore del qubit.
Dato un fotone, si è visto che, ad esso è associato un qubit legato al suo stato di polarizzazione che possiamo indicare con la lettera ψ o con il simbolo $|\psi\rangle$ come ha fatto il signor Magri.
Dal concetto di bit, ci aspettiamo che al qubit sia associato un valore binario, o lo zero oppure l'uno.
Il valore binario associato al qubit non è il suo stato $|\psi\rangle$ in sé, quindi ci chiediamo in che forma sia ad esso legato.
Per determinare il valore di un qubit è necessario sempre compiere una **misura**, nel caso in questione una misura della sua polarizzazione.
La misura *obbliga* lo stato (o funzione d'onda) ψ a *collassare* in uno dei due stati che rappresentano la base scelta per eseguire la misura stessa.
Supponendo di aver eseguito la misura rispetto ai due assi dei polaroid, dopo la misura, lo stato $|\psi'\rangle$ del fotone potrà essere:

- $|\psi'\rangle = |0\rangle$ oppure
- $|\psi'\rangle = |1\rangle$

e questo **determina** il valore del qubit ad essere rispettivamente o 0 oppure 1.
Lo stato di un qubit può essere rappresentato graficamente sulla *sfera di Bloch*, un sistema di mappatura per i sistemi quantistici a due livelli (vedi fig. 1 bis).

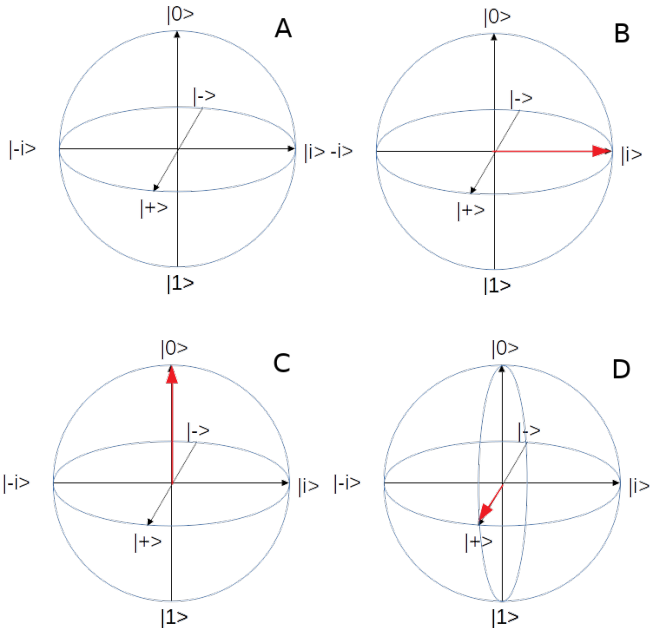

*Fig.1bis La figura mostra la sfera di Bloch, un sistema grafico per rappresentare lo stato di un qubit (**A**). Il vettore che giace lungo il semiasse positivo indica che il qubit è nello stato* $\frac{1}{\sqrt{(2)}}(|0\rangle + i|1\rangle$ *(**B**). Il qubit è nello stato* $|0\rangle$ *(**C**). Il qubit è nello stato* $|+\rangle \frac{1}{\sqrt{(2)}}(|0\rangle + |1\rangle >$ *(**D**).*

Domanda n. 7 La principale differenza tra i *bit-matita* e i *bit-fotoni* è che:

a. Per i primi c'è una relazione deterministica tra il loro stato e la misura del loro valore, mentre per i secondi no
b. I *bit-matita*, sebbene non efficienti, sono implementabili realmente, i secondi sono solo un ipotesi teorica
c. Non mostrano differenze e per questo è possibile realizzare concretamente dispositivi che sfruttano la meccanica quantistica per la computazione

3. Principi della meccanica quantistica

In questo capitolo si pongono le basi canoniche per comprendere i principi della meccanica quantistica. I concetti presentati in modo informale e discorsivo nel capitolo precedente, sono qui ripresi dall'inizio e sviluppati fin dove necessario agli obiettivi del testo.

Per comprendere ed *accettare* la formulazione della meccanica quantistica, così come si presenta oggi, è importante partire dalla visione di chi l'ha concretamente formulata, cioè P.A.M Dirac (Dirac, 1957).
Agli orecchi di chi non è un fisico di professione, questo potrebbe sembrare strano: *in fondo si tratta di formule che o sono giuste o sono sbagliate!*
Non è così perché la fisica è la risposta alle domande che ci si pone e la sua forma dipende dalle domande stesse e dal modo in cui queste sono state poste.
Per questo studiare le risposte della fisica, senza conoscere il pensiero di chi l'ha formulata, può essere molto complesso, specialmente se la formulazione non è intuitiva, come appunto il caso della meccanica quantistica.

Dirac scrive che l'obiettivo principale della fisica non è quello di fornire una raffigurazione della natura, ma di fornire le leggi che governano i fenomeni conosciuti e che possono guidare alla scoperta di fenomeni nuovi.
Si noti che Dirac scrive esattamente:

...the main object of physical science is not the provision of picture,...

evidenziando la **non necessità** di raffigurarsi il mondo quantistico.
Dirac aveva un buon motivo per sostenere questo pensiero visto che si era assunto la responsabilità di stravolgere la visione che

la fisica aveva avuto fino ai primi anni '20.

Il XX secolo era cominciato con diverse *novità* interessanti per la fisica e la formulazione delle leggi dell'elettromagnetismo aveva aperto le porte a nuove frontiere dello studio e della ricerca. All'inizio del secolo Rutherford ipotizza il modello planetario per gli atomi, de Broglie ipotizza la dualità onda corpuscolo, Planck ipotizza il quanto di azione che diverrà la base per la meccanica quantistica ed Einstein spiega l'effetto fotoelettrico basandosi sull'idea di fotone.
I fisici e le scoperte sarebbero molti di più, ma ci accontentiamo di questi appena citati perché ci servono solo per giungere ad altri due: Schrodinger e Heisenberg che furono tra i primi a comprendere che le scoperte e le teorie che avevano salutato il nuovo secolo non potevano trovare una collocazione nel *vecchio* paradigma classico newtoniano e che un nuovo paradigma doveva essere formulato.
Lo fecero entrambi, ognuno a modo suo. Il primo formulò la **meccanica ondulatoria** e il secondo la **meccanica matriciale**.
La prima era più semplice, perché parlava una lingua più conosciuta ai fisici ed ebbe maggior successo iniziale, ma la seconda si rivelò più semplice nel proseguo perché era complessa per trattare i problemi semplici, ma semplice per i problemi difficili. Fu raccolta da Dirac e divenne la **meccanica quantistica**.

Questa breve introduzione ha avuto lo scopo di preparare il lettore digiuno di meccanica quantistica a quanto seguirà, perché la meccanica quantistica può essere apprezzata più facilmente se si comprende a fondo che nasce dal pensiero di altri esseri umani, non è vera o falsa, è una teoria, un modo di pensare e di sperimentare, che si sta rivelando molto utile, tanto quanto lo è stata la meccanica newtoniana prima di lei.

Domanda n. 8 Chi è considerato il *padre* della meccanica ondulatoria?

a. Dirac
b. Schrodinger
c. Heisenberg

3.1 Formulazione matematica del principio di sovrapposizione

Dobbiamo ora addentrarci nei dettagli della teoria quantistica. Nella trattazione che segue si è cercato di estrapolare solo il necessario della teoria ai fini dell'introduzione all'informatica quantistica, attenendosi però rigorosamente al testo di Dirac da cui nasce la principale linea di interpretazione della meccanica quantistica, quella detta della *Scuola di Copenaghen*.

Il lettore sia rassicurato: non è necessario comprendere tutto e perfettamente, ma piuttosto sia determinato a leggere questi capitoli per intero, in modo da non trovarsi sprovveduto nei capitoli dedicati all'informatica dove i concetti e le idee qui esposte troveranno applicazione.
Non si saltino poi le domande e ci si fermi a ragionarci sopra, consultando nel caso le risposte in fondo al testo.

Stato fisico

Consideriamo un sistema atomico composto di corpi (puntiformi) ognuno con una specifica massa, carica elettrica, ecc. I corpi interagiscono tra loro attraendosi o respingendosi secondo certe leggi della fisica.
Ci saranno vari possibili moti per ognuno di questi corpi che saranno in accordo con le leggi della fisica. Seguendo Dirac, chiameremo ognuno di questi possibili moti *stato* del sistema.

Nella meccanica classica lo stato di un sistema può essere completamente definito misurando le tre coordinate x, y, z e le tre componenti v_x, v_y e v_z della velocità di ognuno dei corpi che costituisce il sistema. Questo risulta possibile perché la meccanica classica è stata storicamente applicata a sistemi di *grande massa* che coinvolgono sempre, almeno, decine di migliaia di atomi.
La meccanica quantistica si applica ai sistemi composti anche da pochi o singoli atomi e alle particelle elementari. A tali dimensioni si ha che:

i corpi non posseggono simultaneamente una posizione x ed una quantità di moto p definita con precisione, ma esiste una incertezza minima data da $\Delta x \Delta p \geq \hbar$.
Quello appena enunciato è noto come il principio di indeterminazione di Heisenberg.
A causa di questa incertezza, non è possibile definire il concetto di traiettoria per i corpi come atomi ed elettroni e per questo motivo il concetto di stato fisico **non può essere preso in prestito** dalla meccanica classica, ma:
lo stato di un sistema *piccolo* (quantistico) deve essere specificato usando meno variabili rispetto uno classico o usando dei dati che sono più indefiniti. (Dirac)

Probabilità e misura nella meccanica quantistica

Sia la meccanica ondulatoria che quella matriciale introducono un'ulteriore novità rispetto alla *picture* della natura che era stata indotta dalla meccanica classica.
In meccanica classica conoscere lo stato di un sistema fisico significa poter prevedere il risultato di una misura.
Si consideri ad esempio il sistema fisico costituito dalle sponde di un biliardo e dalle palle libere di rotolare sul tappeto. Se ad un certo istante sono note le coordinate x, y e la velocità v_x, v_y

4. I calcoli dei computer, consumano energia?

L'informatica quantistica è nata dallo sforzo di comprendere quali fossero le limitazioni della scienza informatica, dovute alle leggi della fisica.

Tra i primi ad interessarsi a questo problema ci sono stati i fisici Bennett, Fredkin, Toffoli e Feynman. All'inizio degli anni '80, Bennet suggerì che un computer potesse essere pensato come una macchina che trasforma **energia libera** in calore con lo scopo di compiere un *lavoro matematico*: una definizione alla quale si potrebbe non essere abituati, se non si guarda il computer dal punto di vista *meccanico* (Bennett, 1982).
D'altra parte i computer sono meccanismi costruiti usando componenti elettronici che dissipano calore sia per elaborare i dati sia per mantenerli nella memoria volatile. Quindi, da questa prospettiva, la definizione di Bennet non è poi così peregrina come potrebbe sembrare a prima vista.
Alcuni decenni prima, von Neumann, aveva condotto i primi studi sugli elaboratori elettronici e sugli automi cellulari. Uno dei suoi argomenti di ricerca era stato proprio il calcolo dell'energia associata alla computazione e, con maggior precisione, e si era interessato al calcolo dell'aumento dell'**entropia** durante la computazione.

4.1 Entropia

L'entropia S è una grandezza fisica molto importante nella descrizione dei sistemi fisici che sono coinvolti in processi di trasformazione, come ad esempio una pentola d'acqua che si sta scaldando fino all'ebollizione per cuocere la pasta, o un caminetto in cui la legna arde trasformando l'energia chimica in calore. Questa grandezza è altrettanto importante per descrivere in modo completo anche i sistemi che producono, trasformano e trasmettono *informazione*.

Definizione in fisica dell'entropia

Il concetto di entropia è stato introdotto da Carnot e Clausius per quantificare la tendenza naturale dei sistemi fisici ad evolvere verso una precisa direzione anziché un'altra. Per esempio lasciando cadere una goccia di inchiostro in un bicchiere d'acqua ci si aspetta che l'inchiostro si diffonda in modo disordinato (direzione attesa) anziché formando delle figure geometriche regolari (direzione alternativa).
Nei termini di temperatura (T) e calore (Q), l'entropia è definita come:

$$\mathbf{S} = \sum \frac{dQ_i}{T_i}$$

dove T_i è la temperatura assoluta e dQ_i è il calore scambiato in una trasformazione composta da un insieme discreto di passaggi, dove ogni passaggio è indicato con l'indice i.
Al fine di comprendere le motivazioni dell'informatica quantistica, è molto istruttiva la sua formulazione data da Boltzmann. Quest'ultima definizione di entropia è basata sulla statistica: molto rigorosa e sopratutto generalizzabile anche ad ambiti diversi dalla fisica.
L'entropia S può essere scritta in termini statistici come:

$$\mathbf{S} = k \sum p_j ln(\frac{1}{p_j})$$

dove p_j è la probabilità che il sistema si trovi nello stato $j - esimo$ e k è la costante di Boltzmann.

Box: entropia nel gioco dei dadi

Si consideri di avere a disposizione sei dadi e di disporli in modo ordinato in modo che la somma dei loro numeri dia 36. Perché si verifichi questa condizione è necessario che tutti i dadi rivolgano al cielo la faccia con il numero 6. Per ogni dado esiste una solo caso su sei per corrispondente alla faccia 6, quindi esiste un'unica disposizione di dadi che porta il sistema dei sei dadi a dare come somma 36.
Si consideri ora la situazione in cui la somma dei dadi dà come risultato il numero 21.
Usando la definizione di entropia data poco sopra, risulta immediato verificare che l'entropia per la disposizione dei sei dadi tutti risultanti nel numero 6 ha una bassa entropia, mentre la disposizione risultante nei dadi che danno come somma 21 ha un'alta entropia.
Questa analisi qualitativa ci è sufficiente per intuire che un sistema che si trovi in uno stato a bassa entropia, scivolerà più facilmente verso uno stato ad entropia più alto, piuttosto che viceversa.

Esperimento mentale
Per comprendere ancora più a fondo, si può fare un semplice esperimento mentale. Si immagini che i dadi siano contenuti in un vaso e che si trovino nello stato iniziale, ordinato, con tutte le facce rivolte verso il 6, in modo che la somma dia 36. Questa situazioni corrisponde ad un minimo di entropia.
Ora si scuota il vaso in modo che i dadi cambino il loro valore casualmente. Quando ogni dado si sarà fermato, si sommino i valori delle sei facce. Con ogni probabilità si troverà un valore diverso da 36. Se si ripete l'esperimento molte volte, la media dei valori dopo lo

scuotimento del vaso, scivolerà un po' alla volta verso il 21.
Questo semplice esperimento ci fornisce una idea intuitiva di un principio molto importante: in natura, i sistemi tendono ad assumere configurazioni con entropia crescente.

Entropia e reversibilità

Sempre continuando con lo stesso esperimento, vediamo che il concetto di entropia è legato anche al concetto di irreversibilità di una trasformazione.

Scelta una faccia da uno a sei, la probabilità che lanciando sei dadi si ottengano sei facce uguali a quella scelta è indipendente dal numero scelto. Per esempio ottenere sei facce con il numero sei ha la stessa probabilità di ottenere sei facce con il cinque, il quattro ecc.

Immaginiamo ora di disporre i dadi allineati in modo che la faccia superiore sia il numero 6 e la faccia frontale sia il 4.
Dallo stato iniziale, in cui tutti i dadi hanno valore 6, incliniamo lentamente il vaso, in maniera che tutti i dadi rotolino nello stesso modo e si posizionino tutti sulla faccia del numero 4. È un esercizio difficile ma, almeno mentalmente, possiamo pensare di riuscire ad eseguirlo.
Ora che tutti i dadi sono sullo stato 4, possiamo eseguire il movimento inverso e riportare i dadi sul valore 6: abbiamo eseguito una trasformazione **reversibile**, caratterizzata dal non aver aumentato l'entropia totale del sistema.

L'argomentazione usata ha solo un valore intuitivo, la relazione esatta tra l'entropia e la reversibilità deve

essere studiata su un testo di termodinamica dedicato, come ad esempio *Termodinamica* di Enrico Fermi.

Ai fini della comprensione del resto del testo, l'idea intuitiva che abbiamo delineato è sufficiente.

Domanda n. 15
Quante sono le possibili combinazioni di sei dadi che corrispondono alla somma 36?

a. Una sola
b. Sei

4.2 Termodinamica della computazione irreversibile

In questo paragrafo vengono introdotti due concetti importantissimi nella *teoria dell'informazione* sia classica che quantistica: l'entropia dell'informazione e l'entropia algoritmica. Concetti necessari per capire l'idea di computazione reversibile ed irreversibile, la prima delle quali è alla base dell'informatica quantistica.

Entropia dell'informazione

Contemporaneo di von Neumann, il matematico Claude Shannon introdusse il concetto di **entropia dell'informazione** applicando la statistica al concetto di informazione (Shannon, 1948).
In modo molto diretto egli definì lo schema della **teoria dell'informazione** in cinque blocchi uniti da un canale, detto canale dell'informazione. Da un lato si ha la sorgente dell'informazione e il sistema di trasmissione, mentre dall'altro si trova il sistema di ricezione e la destinazione. Tra i due blocchi si trova la sorgente di *rumore* che può intervenire nella

comunicazione alterando l'informazione.

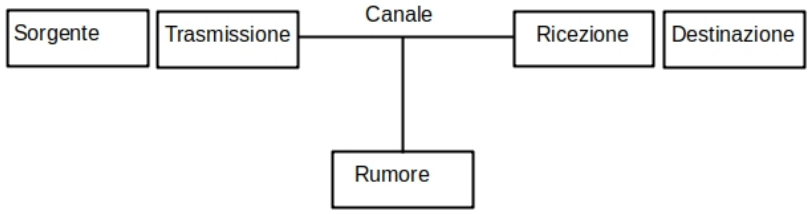

Fig.1a Schema del sistema di comunicazione proposto da C. Shannon.

L'informazione viene scambiata per mezzo di messaggi che sono semplicemente delle sequenze di simboli, per esempio i valori 0 e 1.

Partendo dall'idea che per ogni trasmissione l'informazione è trasportata da un messaggio e che ogni messaggio è una sequenza di simboli, egli definì p_i la frequenza con cui appare il simbolo i-esimo nel messaggio e mostrò che la funzione H definita come segue:

$$\mathbf{H} = \sum p_i log_2(\frac{1}{p_i}) = -\sum p_i log_2(p_i)$$

rappresentava l'**incertezza** contenuta nel messaggio. (Nota che la scelta della base 2 per il logaritmo, non è vincolante nella teoria.)

Un'applicazione della teoria dell'informazione di Shannon

La teoria dell'informazione e il concetto di **incertezza** o **arbitrarietà** contenuta in un messaggio, è piuttosto complessa e merita uno studio dedicato. Ai fini della comprensione di ciò che segue, dobbiamo però notare quanto la definizione di H ricalchi quella data per l'entropia S.
Consideriamo un circuito logico con due variabili di ingresso

a, b e due di uscita a', b' composto da una porta AND e una OR definito dalla seguente tabella di verità:

a	b	a'	b'
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

Consideriamo ora una versione modificata dello schema a cinque blocchi visto sopra, in cui tra **sorgente** e **destinazione** introduciamo un blocco di **elaborazione**.

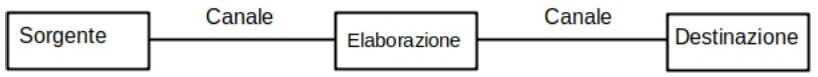

Fig.2 Sistema di comunicazione con un elemento di elaborazione dell'informazione tra sorgente e destinatario.

Per inquadrare la situazione secondo la teoria appena illustrata, dobbiamo considerare come simboli della trasmissione non i singoli bit, ma le coppie di bit. Quindi i simboli possibili per ogni messaggio trasmesso dalla sorgente sono:

$$\{(1,1),(1,0),(0,1),(0,0)\}$$

Nel caso in cui la sorgente abbia arbitrio totale sui simboli da inviare, essi hanno tutti probabilità $p = \frac{1}{4}$, quindi, usando la formulazione vista sopra per l'entropia, calcoliamo che l'**incertezza** dei messaggi trasmessi è 2 .

I possibili messaggi ricevuti, sono invece quelli derivabili dalle due colonne a', b' della tabella e, come si nota, si riducono a soli tre distinti tra loro:

$$\{(1,1),(0,1),(0,1),(0,0)\}$$

Il risultato dell'elaborazione è quindi quello di aver ridotto il

numero di simboli possibili che possono essere ricevuti e di aver aumentato la probabilità (o frequenza) di uno di essi, cioè la coppia $(0, 1)$.

È naturale chiedersi se l'elaborazione abbia avuto un effetto sulla quantità H definita poco prima.
Se calcoliamo H tenendo conto di quanto appena visto, vediamo che essa scende dal valore 2 al valore $\frac{3}{2}$. Inoltre notiamo un'altra cosa: alla variazione della funzione H è associato ad un altro concetto importante: il messaggio elaborato è **irreversibile**, cioè da esso non è più possibile risalire al messaggio trasmesso, perché per il simbolo ricevuto $(1, 0)$ esistono due diversi possibili simboli trasmessi.

La funzione H è quindi un analogo dell'entropia statistica vista sopra, e per questa analogia, pare che von Neumann abbia suggerito a Shannon di chiamarla entropia dell'informazione. Sembra inoltre che von Neumann abbia corroborato il suo suggerimento dicendo che visto che in pochi capivano il significato dell'entropia fisica, Shannon non avrebbe avuto critiche presentando il suo lavoro.

Originariamente Shannon aveva sviluppato e applicato la funzione H sopratutto ai problemi di trasmissione dell'informazione disturbati da sorgenti di rumore piuttosto che a trasmissioni trasformate da unità di elaborazione come nell'esempio presentato sopra. Tale argomento viene invece ripreso e sviluppato da Bennet che propone una sintesi del concetto di entropia algoritmica.

Entropia algoritmica

Vediamo ora di sviluppare il concetto di entropia dell'informazione di Shannon verso quello di entropia algoritmica seguendo la formalizzazione del problema come

proposta da Bennet.
Egli definisce lo scopo di una computazione come quello di produrre una stringa di *output* x per mezzo dell'esecuzione di un programma.
Se questo dovesse sembrare strano, si pensi che qualsiasi risultato otteniamo da uno strumento informatico (digitale) è sempre una stringa di bit o una sua successiva *trasduzione* in un diverso segnale. Per esempio, se ascoltiamo della musica da un dispositivo informatico digitale, il suono che arriva alle nostre orecchie è il risultato della conversione di un segnale digitale in uno elettrico analogico e successivamente in un segale acustico (onda di pressione).
Per continuare il ragionamento, definiamo la lunghezza di una stringa come il numero di bits da cui è composta. Definiamo poi l'**entropia algoritmica** come il numero minimo di bits per configurare un programma che produca l'output x.
Per afferrare il significato della grandezza appena introdotta, si provi ad immaginare di voler produrre delle stringhe casuali di bit. In questo caso, il programma non può sfruttare nessuna regola per produrre l'output x in quanto, se le stringhe sono casuali, la successione di 1 e 0 non ha seguito nessuna logica. Per configurare il programma serviranno quindi tanti bit quanti sono quelli che compongono la stringa x. Questo è il caso di **massima entropia**.
Adesso, si provi invece a considerare un programma che deve produrre la divisione intera per due di un dato input $a : x = \frac{a}{2}$.
Dal punto di vista computazionale, per eseguire questo calcolo basta spostare tutti i bits dell'input a a destra di una unità, quindi, in linea di principio, il programma può essere configurato usando un unico bit di informazione, perché l'operazione di spostamento a destra del bit deve essere applicata a tutti i bit nello stesso modo. Questo è un caso di **minima entropia**.
Per chiarire ancora di più le idee, si consideri di voler ottenere

che la stringa x sia ottenuta dalla divisione per 2 dell'input a nel caso a sia pari, mentre sia il risultato del prodotto per 2 sempre dell'input a nel caso a sia dispari. Semplificando molto il ragionamento, vediamo che sono necessari due bit per configurare il programma che esegue questa elaborazione. Il primo per stabilire se a è pari o dispari (in pratica controllando lo stato del bit 0) e il secondo per traslare a destra o a sinistra la stringa a.
Probabilmente, a qualcuno, non sfuggirà la stretta relazione esistente tra entropia algoritmica e la **complessità di Kolmogorov**.

Domanda n. 16
Quale affermazione tra quelle che seguono è falsa?

a. L'entropia dell'informazione è massima quando tutti i simboli hanno la stessa probabilità di presentarsi
b. L'entropia dell'informazione non è mai negativa
c. Le due precedenti affermazioni sono false

Entropia di un sistema

Prima di tirare le somme del ragionamento che abbiamo iniziato sull'entropia legata alla computazione, è necessario fare una piccola digressione sull'entropia di un sistema fisico.

La scienza che descrive la trasformazione del calore in lavoro e viceversa, è nota con il nome di termodinamica. La termodinamica si basa su tre principi, oppure può essere vista come una conseguenza statistica della teoria cinetica dei gas.
Nella termodinamica vi sono grandezze macroscopiche, come la temperatura, che sono il risultato del valore medio di grandezze microscopiche come l'energia cinetica delle particelle costituenti il gas. Tali grandezze macroscopiche hanno quindi

una *controparte* microscopica.
L'**entropia** invece, è una grandezza termodinamica macroscopica, presente sia nella descrizione classica che in quella statistica, che non ha una controparte microscopica. Non esiste quindi l'entropia di una particella e neanche una grandezza fisica della particella la cui media fornisca il valore dell'entropia del sistema.

Ora che abbiamo apprezzato questa caratteristica dei sistemi fisici, torniamo a considerare l'**entropia algoritmica** e vediamo che questa è l'analogo *microscopico* della entropia termodinamica *macroscopica*. In pratica si può calcolare l'entropia termodinamica se è nota l'entropia algoritmica di un sistema.
Consideriamo un sistema termodinamico il cui comportamento macroscopico sia descrivibile in *modo conciso* (*Bennet*), cioè attraverso delle equazioni del moto macroscopiche, quindi senza bisogno di seguire le componenti del sistema ad una ad una. A questo proposito, Bennet, dice esplicitamente:

> *A macrostate is concisely describable, if it is determined by equations of motion and boundary conditions, describible in small number of bits...*

Per un tale sistema, Bennet afferma:

> *...its statistical entropy is nearly equal to the ensamble average of microstates' algorithmic entropy*

In altri termini, Bennet ci dice che per un sistema non del tutto *casuale*, quindi che abbia un certo grado di macro-determinismo, il grado di *casualità* è dovuto alla **entropia algoritmica** delle singole componenti del sistema stesso.
Questo implica che per trasformare la quantità $kT\ln(2)$ di **calore** in **lavoro** è necessario aumentare di un bit l'**entropia algoritmica** di un sistema fisico, e corrispondentemente, per diminuire di un bit l'entropia del sistema è necessario fornire un

a	b	c	d	a'	b'	c'	d'
1	1	1	0	1	0	1	1
1	1	0	0	1	0	0	1
1	0	1	0	1	1	0	1
1	0	0	0	1	1	1	0
0	1	1	0	0	1	0	1
0	1	0	0	0	1	1	0
0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0

I valori delle variabili c' e d' che corrispondono alla somma e al resto dell'addizione tra a e b mostrano che il circuito realizzato:

- implementa correttamente la somma binaria completa (*full-adder*)
- è reversibile, perché per ogni configurazione di output esiste una sola configurazione di input.

Spazzatura binaria

Il circuito *full-hadder* appena costruito produce due bits di *spazzatura*, infatti i due output a' e b' non fanno parte delle informazioni ricercate, in pratica sono inutili ai fini informativi.
Come fece notare Feynman, la *spazzatura* è un elemento imprescindibile nella computazione reversibile, ma è proprio ciò che serve per poter tornare indietro, cioè risalire alla configurazione dei bit prima della loro elaborazione. Comunque, i bit inutilizzati, come b' possono essere ulteriormente trasformati per produrre qualcosa di utile. Ad esempio, se nel circuito del full-adder viene aggiunto un CONTROLLED NOT

sulla linea b' come mostrato di seguito:

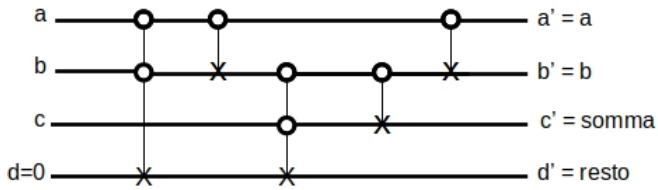

Fig.14 Circuito full-adder reversibile, con recupero del bit spazzatura.

le linee a e b risultano entrambe invariate dopo la computazione, e possono essere usate come ingressi per altri circuiti.

La proprietà appena vista si può generalizzare ad ogni circuito:

Per ogni circuito irreversibile con n bits di input e m di output è sempre possibile realizzare un circuito reversibile con $n+m$ bits di input dei quali m valorizzati a 0 e $n+m$ di output dei quali n valorizzati ai valori di input.

Domanda n. 18
Da chi fu presentato un circuito full-adder nel 1982?

a. Da Feynman
b. Da Shannon
c. Da Dirac

5. Gates quantistici

In questa sezione vediamo gli elementi alla base di un computer quantistico, cioè i sistemi fisici che seguono le regole imposte per i componenti **reversibili** visti nel capitolo precedente.

Le porte reversibili introdotte fin qui possono essere realizzate sfruttando le consuete tecnologie elettroniche. Quello che vedremo ora, invece, è come esse possano essere realizzate sfruttando le proprietà quantistiche della materia, proprietà che *spariscono* nel mondo macroscopico a cui siamo abituati, ma che sono la *vera legge* del mondo microscopico.

Nell'informatica quantistica, ci si riferisce a questi elementi circuitali indicandoli come **quantum gates**, in stretta analogia ai **logic gates** della computazione classica.
Nel seguito faremo uso di questo termine.

NOTA bene: nei capitoli introduttivi abbiamo visto come lo stato di polarizzazione di un fotone sia una buona osservabile (vedi paragrafo 3.5) per rappresentare il valore di un qubit. Anche le particelle *fermioniche* (cioè particelle a spin semi intero scoperte da Enrico Fermi) a *spin* uguale ad un mezzo possono essere usate per rappresentare i quibits e ancora altri sistemi quantomeccanici che posseggono solo due stati di base, ma in questo testo, considereremo sempre che il valore dei qubits sia associato allo stato di polarizzazione dei fotoni. I concetti illustrati sono indipendenti dalla reale tecnologia e usata per ottenere e manipolare lo stato dei qubits.

5.1 Il NOT come gate quantistico

Consideriamo il qubit a che abbia valore 1 rispetto alla base standard. Possiamo scriverlo usando la notazione dei ket come segue:

$$a = |1\rangle$$

o equivalentemente come un vettore colonna:

$$\mathbf{a} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Vogliamo vedere come trasformare il valore del qubit $a = 0$ in $a' = 1$. Questa trasformazione non è un puro problema matematico, ma deve essere realizzabile a livello sperimentale se vogliamo capire come funziona nella realtà un computer quantistico.
La matematica ci guida nella comprensione dei fenomeni fisici e la fisica è la scienza della realtà, le leggi della fisica descrivono processi che avvengono in natura quindi se troviamo una trasformazione quantomeccanica che realizza l'operazione matematica voluta, troviamo anche l'operazione fisica da compiere per realizzarla.
Con questa premessa, vediamo che per trasformare il valore di a da 1 a 0, che è una operazione matematica, è necessario a livello fisico trasformare lo stato di un fotone da $|1\rangle$ ad $|0\rangle$ cambiandogli la polarizzazione agendo con uno strumento ottico che come vediamo qui di seguito dovrà compiere una trasformazione diversa da una semplice rotazione.

Dal punto di vista matematico formale, si tratta di trasformare il vettore

$$\mathbf{a} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

nel vettore

$$\mathbf{a}' = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Detta trasformazione può essere ottenuta operando su a con la matrice

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

Si noti che il simbolo X usato per indicare la matrice è lo stesso usato nel simbolo del CNOT descritto precedentemente.

Dal punto di vista algebrico, la negazione di un qubit è quindi ottenuta moltiplicando la matrice X per il vettore a come segue:

$$a' = Xa$$

Dal punto di vista fisico invece la cosa è più complessa. Non sarà infatti sfuggito che la trasformazione X **non equivale** ad una rotazione nel piano. Le rotazioni nel piano xy attorno all'asse z, di un angolo θ misurato in senso orario, possono essere rappresentate (vedi paragrafo 3.3) con matrici 2×2 del tipo

$$\mathbf{R} = \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix}$$

che, come si deduce, non possono essere ridotte alla matrice X che ha i due elementi sulla diagonale secondaria dello stesso segno mentre per ogni angolo θ gli elementi sulla diagonale secondaria della matrice di rotazione R hanno sempre segno opposto (tranne quando valgono 0).

Per realizzare fisicamente la trasformazione X si usano degli strumenti ottici anche molto semplici come il *beam splitters* che consiste in due prismi di cristallo incollati tra loro.

Il circuito NOT per un singolo qubit è rappresentato graficamente come un quadrato (*box* in inglese) con una X.

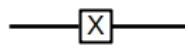

Fig.15 Simbolo dell'X gate.

Il NOT come trasformazione unitaria

Abbiamo visto che l'operazione NOT di un qubit equivale al prodotto della matrice X per il vettore rappresentante il suo stato di polarizzazione.
La matrice X è una trasformazione unitaria nel senso che si ottiene da un operatore unitario (paragrafo 3.3) e può essere rappresentata in forma tensoriale come segue::

$$X = |0\rangle\langle 1| + |1\rangle\langle 0|$$

La sua azione sui kets di base è data dalle trasformazioni:

$$|1\rangle = (|0\rangle\langle 1| + |1\rangle\langle 0|)\,|0\rangle$$

e

$$|0\rangle = (|0\rangle\langle 1| + |1\rangle\langle 0|)\,|1\rangle$$

La negazione di un qubit è quindi una operazione **quantistica** che agisce nello spazio degli stati del fotone, e questo significa che esiste una azione fisica **reversibile** che trasforma un qubit da 0 ad 1 e da 1 a 0, come anticipato nel paragrafo precedente.

Il risultato appena ottenuto ci dice che un'operazione logica come la negazione può essere ottenuta come una trasformazione quantistica reversibile che agisce su un singolo fotone. Quindi possiamo dire che:

per modificare il valore di un qubit è necessario trasformare lo stato del fotone che lo rappresenta.
A ben pensarci una affermazione equivalente è corretta anche per l'informatica classica, ma in quell'ambito la fisica è più nascosta, mentre emergono problemi più ingegneristici che vengono trattati con più *senso comune* e meno formalismo fisico.

Prima di proseguire, notiamo un aspetto della computazione quantistica di straordinaria importanza (Rieffel, 2015 CAP 5),

cioè che:

ogni calcolo può essere decomposto in trasformazioni che agiscono sul singolo qubit e in una trasformazione che agisce su due, come il CNOT.

in pratica, dopo aver definito la trasformazione NOT che agisce sul singolo qubit e la trasformazione CNOT che agisce su due, è possibile implementare qualsiasi operazione logica in termini quantistici.

Questo primo risultato evidenzia come la computazione automatica che attualmente è realizzata da circuiti elettronici irreversibili, può essere implementata nei termini di trasformazioni quantistiche reversibili. Aver dimostrato che è possibile replicare la computazione *classica* in termini quantistici, è ovviamente un passo necessario per dare valore scientifico all'informatica quantistica.

5.2 Matrici di Pauli

La matrice X che abbiamo introdotto nel paragrafo precedente è una delle tre matrici di *Pauli* definite come segue:

$$\sigma_{\mathbf{x}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma_{\mathbf{y}} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$\sigma_{\mathbf{z}} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Accanto alle matrici di Pauli vediamo le loro trasposte coniugate:

$$\sigma_{\mathbf{x}}^{*} = \sigma_{\mathbf{x}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\sigma_{\mathbf{y}}^{*} = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}$$

$$\sigma_{\mathbf{z}}^{*} = \sigma_{\mathbf{z}} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Le matrici appena descritte sono matrici unitarie, cioè (ricordiamo) per loro vale la proprietà seguente:

$$\sigma_x \sigma_x^* = \sigma_y \sigma_y^* = \sigma_z \sigma_z^* = I$$

dove I è la consueta matrice identità definita come:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Da ognuna delle matrici di Pauli, in quanto matrici unitarie, è possibile *teoricamente* ottenere un corrispondente **quantum gate**.

Per passare dalla *teoria* alla *pratica*, bisogna verificare se esiste nella realtà un *setup sperimentale*, cioè un dispositivo concreto, che agisca concretamente sulla polarizzazione dei fotoni, così come l'operatore agisce sui kets.

A questa domanda fornirono una risposta di carattere generale Reck, Zeilinger, Bernsetin e Bertani che con una lettera pubblicata sulla rivista *Physical Review Letters*, presentarono un procedimento concreto per realizzare in laboratorio una trasformazione ottica da un sistema di input di N stati in un sistema di output di N stati usando solo *beam splitter*, *phase shifter* e *mirrors*. Testualmente scrivono: (Reck, 1994):

Per quel che ci risulta, questa è la prima volta che viene presentata una prova pratica che ad ogni operatore unitario discreto può essere associato un esperimento nella realtà.

Da questo, sappiamo che la *matematica* che stiamo scrivendo ha una corrispondenza nella realtà e abbiamo quindi la stessa

tranquillità che ha un progettista elettronico nel creare funzioni logiche: sa che esiste un hardware che può implementarle.

Y e Z quantum gates

Il gate Y è costruito sulla matrice σ_y, ma è più pratico definirlo come $iY = \sigma_y$, cioè:

$$\mathbf{Y} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Il gate Z corrisponde invece esattamente alla matrice σ_z:

$$\mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

I due gate vengono rappresentati all'interno di un box quadrato analogamente al X gate.

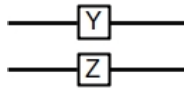

Fig.16 Rappresentazione grafica degli Y e Z gates.

In termini tensoriali, i gates possono essere scritti come:

$$\mathbf{Y} = -|1\rangle\langle 0| + |0\rangle\langle 1|$$

$$\mathbf{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Il gate Z agisce sui kets nella **base standard** cambiandone il segno del rapporto tra a e b:

$$\begin{gathered}(|0\rangle\langle 0| - |0\rangle\langle 1|)\,(a|0\rangle + b|1\rangle) = \\ = a|0\rangle - b|1\rangle\end{gathered}$$

L'azione del gate Y sullo stato $a|0\rangle + b|1\rangle$ è scritta come:

$$\begin{gathered}(-|1\rangle\langle 0| + |0\rangle\langle 1|)\,(a|0\rangle + b|1\rangle) = \\ = -b|0\rangle + a|1\rangle\end{gathered}$$

e può essere vista come l'azione combinata del gate X e del gate Z, infatti si ha $Y = ZX$.

5.3 Il gate H

Con il simbolo di una H dentro un box quadrato si indica la trasformazione di Hadamard, definita dalla matrice:

$$\mathbf{H} = \left(1/\sqrt{2}\right) \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

o in forma tensoriale:

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

Per l'operatore H valgono le stesse considerazioni fatte per le matrici di Pauli quindi può essere realizzato concretamente anche utilizzando degli elementi di ottica lineare.

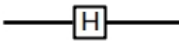

Fig.17 Simbolo dell'Hadamard gate.

Vedremo il suo utilizzo, poco più avanti, per decodificare le informazioni inviate in modo: *dense coding*

5.4 Il gate CNOT

L'azione del CNOT può essere definita solo se si considera un sistema formato da due qubits. Quando questi sono espressi rispetto alla **base standard** allora, come definito nel capitolo precedente, la sua azione è di *negare* il secondo qubit se il primo è 1 o lasciarlo inalterato se il primo è 0.
Prima di analizzare questo gates è necessario formalizzare un insieme di kets di base per rappresentare gli stati di due qubits, ovviamente la formalizzazione che segue poggia su quanto visto nel paragrafo 3.3.

Base standard per due qubits

La base standard per un sistema formato da due qubits è data dal prodotto tensoriale delle due basi singole, quindi dai quattro ket:

$$|0\rangle \otimes |0\rangle,\ |0\rangle \otimes |1\rangle,\ |1\rangle \otimes |0\rangle,\ |1\rangle \otimes |1\rangle$$

che possono essere scritti più concisamente come segue:

$$|00\rangle,\ |01\rangle,\ |10\rangle,\ |11\rangle$$

In forma vettoriale, i quattro ket di base si rappresentano come segue:

$$|\mathbf{00}> = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|\mathbf{01}> = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|\mathbf{10}> = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|\mathbf{11}> = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Per rappresentare uno stato fisico di qubits si usano i tipi `ket1` per i singoli qubit e `ket2` e `ket3` per gli stati di due e tre qubits rispettivamente.
Per creare un ket di un singolo qubit si usa la funzione `crea_ket1` a cui si passano per argomento i coefficienti delle ampiezze degli stati della base `base1` che è un tipo `struct` che ha due campi `e1` ed `e2` di tipo `elemento1`.

In prima istanza è possibile che si faccia fatica a recepire il modello C presentato nella libreria, ma con un po' di pazienza, ci si accorgerà che esso è più intuitivo di quanto appaia all'inizio. Per rompere il ghiaccio vediamo un primo esempio in cui si crea lo stato di un qubit di valore zero espresso rispetto alla base standard.

```
double complex a = 1;
double complex b = 1;
double complex norm = csqrt(1/(a*a+b*b));
ket1 psi = crea_ket1(a*nomr,b*norm,base_std1);
```

Si vede dalla prima riga del codice che la libreria usa le macro definite in `complex.h` per le operazioni sui numeri complessi.
La variabile `base_std1` è predefinita nella libreria e rappresenta un'istanza del tipo `base1`. La chiamata alla funzione `crea_ket1` ritorna une `ket1` di ampiezze `a` e `b` riferite agli elementi `e1` ed `e2` della base standard che è una variabile predefinita nella libreria e può essere usata nel codice utente dichiarandola come `extern`

Kets per due e tre qubits

Se si prova a stampare a video i membri `x` e `y` del ket ψ si vede che essi corrispondono esattamente ai due parametri `a` e `b` passati per argomento alla funzione:

```
printf("%g+i%g,%g+i%g -> ",psi.x, psi.y);
```

Quindi esiste una relazione uno a uno tra i parametri passati e i

coefficienti dei ket di base usati per definire lo stato.
Analogamente, per creare un ket di stato di due qubits si devono passare quattro parametri `a`, `b`, `c` e `d` che corrispondono alle due ampiezze di ognuno dei due ket:

```
double complex a = 1;
double complex b = 0;
double complex c = 1;
double complex d = 0;

double complex norm = csqrt(1/(a*a+b*b+c*c+d*d));

a = a*norm;
b = b*norm;
c = c*norm;
d = d*norm;

ket2 psi2 = crea_ket2(a,b,c,d,base_std2);

printf("(%g+i%g,%g+i%g)x(%g+i%g,%g+i%g) = ",a, b,c,d);

printf("%g+i%g,%g+i%g,%g+i%g,%g+i%g\n\n",
psi2.x11, psi2.x12,psi2.x21, psi2.x22);
```

Si noti che, diversamente dal caso di un solo qubit, i coefficienti `x11`, `x12`, `x21` e `x22` dei quattro ket di base usati per definire lo stato dei due qubits sono diversi dei coefficienti dei due qubit considerati singolarmente (vedi paragrafo 5.4).

Per creare il ket di stato di tre qubits si usa la funzione

```
ket3
crea_ket3(double complex x1, double complex y1,
      double complex x2, double complex y2,
      double complex x3, double complex y3,
      base3 b);
```

il cui prototipo è analogo a quelli visti per creare stati di due o un qubit. Si rammenti che in questo caso i parametri che definiscono gli stati dei singoli qubits sono sei mentre i coefficienti di base sono otto $\left(2^3\right)$.

Operatori

Gli operatori sono definiti in modo simile ai ket di stato.
Per quelli che devono operare sui ket di tipo `ket1` viene definito un tipo di dato per rappresentare gli elementi dello spazio

tensoriale $V \otimes V^*$. Il nuovo tipo è chiamato `elemento1_1` , dove i due indici *1* e *_1* indicano che l'operatore ha un elemento dello spazio V e uno dello spazio V^*:

```
   typedef struct
  {
    double complex c[2][2];
  } elemento1_1;
```

Essi sono rappresentati dai tipi o*n_n* dove l'indice *n* può essere sostituito con il numero 1,2 o 3, quindi gli operatori che agiscono su stati di un singolo qubit sono rappresentati dal tipo o*1_1* e similmente i tipi o*2_2* e o*3_3* rappresentano gli operatori che agiscono su stati di due e tre qubits rispettivamente.
L'indice *n* si riferisce ai ket mentre *_n* si riferisce ai bra.
Ogni operatore *o1_1* è definito rispetto alla base `base1_1` costituita da quattro elementi di tipo `elemento1_1` come segue:

```
   typedef struct
{
  elemento1_1 e1e_1; //00
  elemento1_1 e1e_2; //01
  elemento1_1 e2e_1; //10
  elemento1_1 e2e_2; //11


} base1_1;
```

La libreria libSSQ provvede una istanza già predefinita del tipo `base1_1` attraverso la variabile `base_std1_1`

```
const base1_1 base_std1_1 =
  {
    {
      {1,0,
      0,0}
    }
    ,
    {
      {0,1,
      0,0}
    }
    ,
    {
      {0,0,
      1,0}
    }
    ,
    {
      {0,0,
      0,1}
    }
  };
```

La variabile può essere usata dichiarandola `extern` nel codice.
Le base standard `base_std1_1` è definita in modo che il prodotto dei suoi elementi con quelli della base standard dei `ket1`, `base_std1`, soddisfi le relazioni definite nel paragrafo 3.3 (Prodotto fra duali e vettori).
Per creare un operatore si usa la funzione

```
op1_1 crea_tensore1_1
(double complex a11,double complex
a12,double complex a21,
double complex a22, base1_1 b)
```

Per esempio, l'operatore identità viene creato come segue:

```
 op1_1 o = crea_tensore1_1(1,0,0,1,base_std1_1);
```

Una volta creato un operatore si può usare per trasformare uno ket usando la funzione `ket1 trasforma_ket1(op1_1 o,ket1 ket)` come nell'esempio che segue:

```
  ket1 psip = trasforma_ket1(o,psi);
```

Dopo la breve descrizione della libreria e del suo uso si raccomanda ora di studiare il codice nel dettaglio e poi passare senza indugio agli esercizi del paragrafo seguente.

libSSQ.h

```
/*************************************************************************
 *
 * libSSQ @2020 Scuola Sisini
 * Licenza GPL3
 *
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see .
 */

#pragma once
#ifndef LIBSSQ_H
#define LIBSSQ_H

#include <complex.h>

/*____________________SINGOLO QUBIT_________________*/

/*_____________________________________________________
 *
```

```
 * Elemento e in  V
 */
typedef struct
{
  double complex c[2];
} elemento1;

/*_____________________________________________________
 *
 * Base qubit singolo
 */
typedef struct
{
  elemento1 e1;
  elemento1 e2;

} base1;



/*_____________________________________________________
 *
 * Un ket di ampiezze complesse x e y sulla base b
 */
typedef struct
{
  double complex x,y;
  base1 b;
} ket1;


/*___________DOPERATORE TENSORIALE_1X1_____________*/

/*_____________________________________________________
 *
 * Elemento  ee* in V(x)V*
 */
typedef struct
{
  double complex c[2][2];
} elemento1_1;

/*_____________________________________________________
 *
 * Base tensoori V@V*
 */
typedef struct
{
  elemento1_1 e1e_1; //00
  elemento1_1 e1e_2; //01
  elemento1_1 e2e_1; //10
  elemento1_1 e2e_2; //11


} base1_1;

/*_____________________________________________________
 *
 * Un operatore sullo spazio 11
 * di componenti a11, a12, a21, a22 sull base b
 *
 */
typedef struct
{
  double complex a11,a12,a21,a22;
  base1_1 b;
} op1_1;


/*_____________________DOPPIO QUBIT_________________*/
```

```
/*_____________________________________________________
 *
 * Elemento  ee in V(x)V
 */
typedef struct
{
  double complex c[4];
} elemento2;

/*_____________________________________________________
 *
 * Base 2 qubits
 */
typedef struct
{
  elemento2 e1f1; //00
  elemento2 e1f2; //01
  elemento2 e2f1; //10
  elemento2 e2f2; //11


} base2;

/*_____________________________________________________
 *
 * Un ket di 2 qubit di ampiezze complesse x1 e y1
 * e x2, y2  sulla base b
 *
 */
typedef struct
{
  double complex x11,x12,x21,x22;
  base2 b;
} ket2;

/*___________DOPERATORE TENSORIALE_2X2_____________*/

/*_____________________________________________________
 *
 * Elemento  efe*f* in V(x)V(x)V*(x)V*
 */
typedef struct
{
  double complex c[4][4];
} elemento2_2;

/*_____________________________________________________
 *
 * Base tensoori V@V*
 */
typedef struct
{
  //-riga 1
  elemento2_2 e1f1e_1f_1;
  elemento2_2 e1f1e_1f_2;
  elemento2_2 e1f1e_2f_1;
  elemento2_2 e1f1e_2f_2;
  //-riga 2
  elemento2_2 e1f2e_1f_1;
  elemento2_2 e1f2e_1f_2;
  elemento2_2 e1f2e_2f_1;
  elemento2_2 e1f2e_2f_2;
  //-riga 3
  elemento2_2 e2f1e_1f_1;
  elemento2_2 e2f1e_1f_2;
  elemento2_2 e2f1e_2f_1;
  elemento2_2 e2f1e_2f_2;
  //-riga 4
  elemento2_2 e2f2e_1f_1;
```

```
  elemento2_2 e2f2e_1f_2;
  elemento2_2 e2f2e_2f_1;
  elemento2_2 e2f2e_2f_2;

} base2_2;

/*_____________________________________________________
 *
 * Un operatore sullo spazio 11
 * di componenti a11, a12, a21, a22 sull base b
 *
 */
typedef struct
{
  double complex a11,a12,a13,a14,
    a21,a22,a23,a24,
    a31,a32,a33,a34,
    a41,a42,a43,a44;
  base2_2 b;
} op2_2;


/*_____________________TRIPLO QUBIT_____________________*/

/*_____________________________________________________
 *
 * Elemento  eee in V(x)V(x)V
 */
typedef struct
{
  double complex c[8];
} elemento3;

/*_____________________________________________________
 *
 * Base 3 qubits
 */
typedef struct
{
  elemento3 e1f1g1; //000
  elemento3 e1f1g2; //001
  elemento3 e1f2g1; //010
  elemento3 e1f2g2; //011
  elemento3 e2f1g1; //100
  elemento3 e2f1g2; //101
  elemento3 e2f2g1; //110
  elemento3 e2f2g2; //111

} base3;

/*_____________________________________________________
 *
 * Un ket di 3 qubit di ampiezze complesse x1 e y1
 * , x2, y2 e x3, y3  sulla base b
 *
 */
typedef struct
{
  double complex x111,x112,x121,x122,x211,x212,x221,x222;
  base3 b;
} ket3;

/*___________DOPERATORE TENSORIALE_3X3_____________*/

/*_____________________________________________________
 *
 * Elemento  efe*f* in V(x)V(x)V*(x)V*
 */
typedef struct
{
```

```
  double complex c[8][8];
} elemento3_3;

/*_____________________________________________________
 *
 * Base tensoori V@V@V@V*@V*@V*
 */
typedef struct
{
  //-riga 1
  elemento3_3 e1f1g1e_1f_1g_1;
  elemento3_3 e1f1g1e_1f_1g_2;
  elemento3_3 e1f1g1e_1f_2g_1;
  elemento3_3 e1f1g1e_1f_2g_2;
  elemento3_3 e1f1g1e_2f_1g_1;
  elemento3_3 e1f1g1e_2f_1g_2;
  elemento3_3 e1f1g1e_2f_2g_1;
  elemento3_3 e1f1g1e_2f_2g_2;
   //-riga 2
  elemento3_3 e1f1g2e_1f_1g_1;
  elemento3_3 e1f1g2e_1f_1g_2;
  elemento3_3 e1f1g2e_1f_2g_1;
  elemento3_3 e1f1g2e_1f_2g_2;
  elemento3_3 e1f1g2e_2f_1g_1;
  elemento3_3 e1f1g2e_2f_1g_2;
  elemento3_3 e1f1g2e_2f_2g_1;
  elemento3_3 e1f1g2e_2f_2g_2;
   //-riga 3
  elemento3_3 e1f2g1e_1f_1g_1;
  elemento3_3 e1f2g1e_1f_1g_2;
  elemento3_3 e1f2g1e_1f_2g_1;
  elemento3_3 e1f2g1e_1f_2g_2;
  elemento3_3 e1f2g1e_2f_1g_1;
  elemento3_3 e1f2g1e_2f_1g_2;
  elemento3_3 e1f2g1e_2f_2g_1;
  elemento3_3 e1f2g1e_2f_2g_2;
   //-riga 4
  elemento3_3 e1f2g2e_1f_1g_1;
  elemento3_3 e1f2g2e_1f_1g_2;
  elemento3_3 e1f2g2e_1f_2g_1;
  elemento3_3 e1f2g2e_1f_2g_2;
  elemento3_3 e1f2g2e_2f_1g_1;
  elemento3_3 e1f2g2e_2f_1g_2;
  elemento3_3 e1f2g2e_2f_2g_1;
  elemento3_3 e1f2g2e_2f_2g_2;

  //-riga 1
  elemento3_3 e2f1g1e_1f_1g_1;
  elemento3_3 e2f1g1e_1f_1g_2;
  elemento3_3 e2f1g1e_1f_2g_1;
  elemento3_3 e2f1g1e_1f_2g_2;
  elemento3_3 e2f1g1e_2f_1g_1;
  elemento3_3 e2f1g1e_2f_1g_2;
  elemento3_3 e2f1g1e_2f_2g_1;
  elemento3_3 e2f1g1e_2f_2g_2;
   //-riga 2
  elemento3_3 e2f1g2e_1f_1g_1;
  elemento3_3 e2f1g2e_1f_1g_2;
  elemento3_3 e2f1g2e_1f_2g_1;
  elemento3_3 e2f1g2e_1f_2g_2;
  elemento3_3 e2f1g2e_2f_1g_1;
  elemento3_3 e2f1g2e_2f_1g_2;
  elemento3_3 e2f1g2e_2f_2g_1;
  elemento3_3 e2f1g2e_2f_2g_2;
   //-riga 3
  elemento3_3 e2f2g1e_1f_1g_1;
  elemento3_3 e2f2g1e_1f_1g_2;
  elemento3_3 e2f2g1e_1f_2g_1;
  elemento3_3 e2f2g1e_1f_2g_2;
  elemento3_3 e2f2g1e_2f_1g_1;
```

```
  elemento3_3 e2f2g1e_2f_1g_2;
  elemento3_3 e2f2g1e_2f_2g_1;
  elemento3_3 e2f2g1e_2f_2g_2;
   //-riga 4
  elemento3_3 e2f2g2e_1f_1g_1;
  elemento3_3 e2f2g2e_1f_1g_2;
  elemento3_3 e2f2g2e_1f_2g_1;
  elemento3_3 e2f2g2e_1f_2g_2;
  elemento3_3 e2f2g2e_2f_1g_1;
  elemento3_3 e2f2g2e_2f_1g_2;
  elemento3_3 e2f2g2e_2f_2g_1;
  elemento3_3 e2f2g2e_2f_2g_2;

} base3_3;

/*_____________________________________________________
 *
 * Un operatore sullo spazio 111
 * di componenti a11, a12,...,a88 sull base b
 *
 */
typedef struct
{
  double complex
  a11,a12,a13,a14,a15,a16,a17,a18,
    a21,a22,a23,a24,a25,a26,a27,a28,
    a31,a32,a33,a34,a35,a36,a37,a38,
    a41,a42,a43,a44,a45,a46,a47,a48,
    a51,a52,a53,a54,a55,a56,a57,a58,
    a61,a62,a63,a64,a65,a66,a67,a68,
    a71,a72,a73,a74,a75,a76,a77,a78,
    a81,a82,a83,a84,a85,a86,a87,a88;
  base3_3 b;
} op3_3;



/*_____________Funzioni_____________________________*/

/*_____________________________________________________
 *
 * crea un ket di stato di ampiezze complesse
 * x e y sulla base b
 */
ket1
crea_ket1(double complex x, double complex y, base1 b);

/*_____________________________________________________
 *
 * crea un ket di stato di due qubits ampiezze complesse
 * x1, y1, x2 e y2 sulla base b
 * come prodotto dei due key
 * non è possibile creare direttamente ket entangled
 */
ket2
crea_ket2(double complex x1, double complex y1,
          double complex x2, double complex y2,base2 b);

/*_____________________________________________________
 *
 * crea un ket di stato di tre qubits ampiezze complesse
 */
ket3
crea_ket3(double complex x1, double complex y1,
          double complex x2, double complex y2,
          double complex x3, double complex y3,
          base3 b);
```

```
/*_____________________________________________________
 *
 * crea un operatore (tensore 1_1) di coefficienti
 * complessi a11, a12, a21 e a22 sulla base b
 */
op1_1 crea_tensore1_1
(double complex a11,double complex a12,double complex a21,double complex a22,
 base1_1 b);

/*_____________________________________________________
 *
 * crea un operatore (tensore 2_2) di coefficienti
 * complessi a11, a12,...,a44 sulla base b
 */
op2_2 crea_tensore2_2
(double complex a11,double complex a12,double complex a13,double complex a14,
 double complex a21,double complex a22,double complex a23,double complex a24,
 double complex a31,double complex a32,double complex a33,double complex a34,
 double complex a41,double complex a42,double complex a43,double complex a44,
 base2_2 b
 );

/*_____________________________________________________
 *
 * crea un operatore (tensore 3_3) di coefficienti
 * complessi a11, a12,...,a44 sulla base b
 */
op3_3 crea_tensore3_3
(double complex a11,double complex a12,double complex a13,double complex a14,
 double complex a15,double complex a16,double complex a17,double complex a18,

 double complex a21,double complex a22,double complex a23,double complex a24,
 double complex a25,double complex a26,double complex a27,double complex a28,

 double complex a31,double complex a32,double complex a33,double complex a34,
 double complex a35,double complex a36,double complex a37,double complex a38,

 double complex a41,double complex a42,double complex a43,double complex a44,
 double complex a45,double complex a46,double complex a47,double complex a48,

 double complex a51,double complex a52,double complex a53,double complex a54,
 double complex a55,double complex a56,double complex a57,double complex a58,

 double complex a61,double complex a62,double complex a63,double complex a64,
 double complex a65,double complex a66,double complex a67,double complex a68,

 double complex a71,double complex a72,double complex a73,double complex a74,
 double complex a75,double complex a76,double complex a77,double complex a78,

 double complex a81,double complex a82,double complex a83,double complex a84,
 double complex a85,double complex a86,double complex a87,double complex a88,

 base3_3 b
 );



/*_____________________________________________________
 *
 * Trasforma un ket tramite un operatore
 *
 */
ket1 trasforma_ket1(op1_1 o,ket1 ket);

ket2 trasforma_ket2(op2_2 o,ket2 ket);

ket3 trasforma_ket3(op3_3 o,ket3 ket);

#endif
```

libSSQ.c

```
/*****************************************************************************
 *
 * libSSQ @2020 Scuola Sisini
 * Licenza GPL3
 *
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see .
 */

#include <complex.h>
#include <stdio.h>
#include "libSSQ.h"

/*_____________________________________________________
 *
 * Base STANDARD qubit singolo
 */
const base1 base_std1 =
  {
    {{1,0}},//0
    {{0,1}} //1
  };

/*_____________________________________________________
 *
 * Base STANDARD 2qubits
 */
const base2 base_std2=
  {
    {{1,0,0,0}},//00
    {{0,1,0,0}},//01
    {{0,0,1,0}},//10
    {{0,0,0,1}} //11
  };

/*_____________________________________________________
 *
 * Base STANDARD 3qubits
 */
const base3 base_std3=
  {
    {{1,0,0,0,0,0,0,0}},//00
    {{0,1,0,0,0,0,0,0}},//00
    {{0,0,1,0,0,0,0,0}},//00
    {{0,0,0,1,0,0,0,0}},//00

    {{0,0,0,0,1,0,0,0}},//00
    {{0,0,0,0,0,1,0,0}},//00
    {{0,0,0,0,0,0,1,0}},//00
    {{0,0,0,0,0,0,0,1}},//00

  };
```

Compilando ed eseguendo il codice si vede che il ket di stato passa da $|0\rangle$ a $|1\rangle$, quindi il qubit passa da 0 ad 1.

3. **Soluzione**: il primo passaggio per risolvere questo esercizio è di creare uno stato entangled. È stato spiegato nella teoria che uno stato entangled è tale perché non può essere scritto come il prodotto di due stati singoli, per tale motivo non è possibile usare la funzione `crea_ket2`, ma il ket deve essere creato specificando manualmente le sue componenti.
   Usando l'operatore σ_x si deve trasformare il qubit del signor Fabbri in modo da codificare il numero 1, cioè il SUD (con riferimento all'esempio della teoria).
   Per decodificare lo stato si deve operare in successione con il gate CNOT e il gate $H \otimes I$, le cui componenti devono essere calcolate sviluppando il prodotto tensoriale dell'operare H e I.

```
#include <complex.h>
#include <math.h>
#include <stdio.h>
#include "libSSQ.h"

extern base2 base_std2;
extern base2_2 base_std2_2;


int main()
{
  printf("_____Dense Coding____\n\n");

  printf("1)Viene creato e condiviso uno stato entangled |00>+|11> tra
 Fabbri e Magri");
  double complex norm = 1/sqrt(2);
  //Non può essere costruito come il prodotto di due ket
  ket2 ent_k = {1*norm,0,0,1*norm,base_std2};

  //Operatore XxI
  op2_2 o2 = crea_tensore2_2(
                       0,1,0,0,
                       1,0,0,0,
                       0,0,0,1,
                       0,0,1,0,
                       base_std2_2);

 ket2  psip2 = trasforma_ket2(o2,ent_k);

 printf("\n\n2)Fabbri codifica il valore 1 (SUD) usando l'operatore
 XxI:\n\n");

 printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g -> ",ent_k.x11, ent_k.x12,
 ent_k.x21, ent_k.x22);
 printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g\n",psip2.x11, psip2.x12,
 psip2.x21, psip2.x22);

 printf("\n3)Fabbri invia il proprio qubit a Magri\n");
```

```
printf("\n4)Magri applica un operatore CNOT e uno HxI:\n\n");
//Passo1: trasforma con Cnot
o2 = crea_tensore2_2(
                      1,0,0,0,
                      0,1,0,0,
                      0,0,0,1,
                      0,0,1,0,
                      base_std2_2);

ket2 psipp2 = trasforma_ket2(o2,psip2);

printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g -> ",psip2.x11, psip2.x12,
psip2.x21, psip2.x22);
printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g\n",psipp2.x11, psipp2.x12,
psipp2.x21, psipp2.x22);

double complex h = norm*1;
o2 = crea_tensore2_2(
                      h,0,h,0,
                      0,h,0,h,
                      h,0,-h,0,
                      0,h,0,-h,
                      base_std2_2);

ket2 psippp2 = trasforma_ket2(o2,psipp2);

printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g -> ",psipp2.x11, psipp2.x12,
psipp2.x21, psipp2.x22);
printf("\t%g+i%g,%g+i%g,%g+i%g,%g+i%g\n",psippp2.x11, psippp2.x12,
psippp2.x21, psippp2.x22);
printf("\n5)Magri ha ricevuto |01> cioè SUD\n\n");


}
```

Compilando ed eseguendo il codice si vede che il ket entangled passa dallo stato $|00\rangle + |11\rangle$ allo stato $|01\rangle$ come atteso.

8. Programmazione di un circuito quantistico

Nel capitolo 5 è stata sviluppata l'idea di *gate quantistici* e si è visto come questi possono essere usati per costruire circuiti reversibili analoghi a quelli classici, come ad esempio il circuito sommatore. Nel capitolo 6 si è utilizzata la meccanica quantistica per ottenere: non clonazione, teletrasporto e codifica densa, che non hanno un analogo classico.

Per apprezzare la differenza tra la computazione classica (o convenzionale) e quella quantistica, vediamo una semplice quanto efficace applicazione del principio di dense-coding.
Il problema che stiamo per affrontare consiste nello stabilire se una certa funzione $f(x)$, di cui non si conosce l'implementazione, sia una funzione costante oppure no.
Per ridurre al minimo la complessità della presentazione facciamo l'ipotesi semplificativa che la funzione accetti un solo bit di informazione. In tal caso possiamo distinguere i due casi, cioè costante o non costante come segue:

Costante:
$f(0) = f(1) = 1$
Non costante:
$f(0) = 0, f(1) = 1$

Se la funzione $f(x)$ è ignota, l'unica strategia con cui si può distinguere tra i due casi è quella di chiamare due volete la stessa funzione passando la prima volta come argomento 0 e la seconda volta 1. Per esempio in linguaggio C avremmo:

```
int f(int x)
{
 return 0;
}
```

per il caso costante e: int f(int x) { return x; } per il caso non

costante. Per testare i due casi si ha:

```
int r[2];
for(int i=0;i<=1;i++)
{
 r[i]=f(i);
}
if(r[0]==r[1])
{
//costante
}
else
{
//non costante
}
```

È chiaro che anche soluzioni del tipo:

```
int r = f(0)+f(1);
```

non cambiano il fatto che il codice della fuzione f venga eseguito per due volte.
Usando il dense coding, possiamo ridurre il numero di chiamate necessarie ad uno, quindi è possibile risolvere il problema posto dimezzando quella che viene anche chiamata *complessità di chiamata o di comunicazione (query)*. Il problema che stiamo per impostare è noto come *Deutsch problem*.

Progettazione del circuito

L'obiettivo è quello di mostrare che la computazione quantistica può risovere il Deutsch problem usando una solo chiamata a funzione. Per dimostrare questa affermazione si deve essere in grado di scrivere un programma (progettare un circuito) che implementi il problema. Al fine di mantenere la reversibiltà della computazione, se una funzione $f(x)$ è rappresentabile

nella computazione classica come:

$$x \to f(x)$$

questa può essere rappresentata

$$x, y \to (x, y \oplus f(x))$$

in quella quantistica.
In pratica si usa più memoria, però dal risultato della computazione è sempre possibile risalire al valore delle variabili di input, quindi la computazione diventa reversibile. Ovviamente affiché la computazione sia reversibile è necessario conoscere la definizione della funzione f, cosa che ci accingiamo a fare.

Il passaggio appena illustrato vale tanto per un circuito quantistico quanto per uno classico, in tutti i casi se si configura la chiamata a funzione come visto sopra, questa risulterà reversibile. Vediamo ora nello specifico come eseguire detta chiamata nel mondo quantistico.
Come abbiamo visto nel capitolo 6, le operazioni che trasformano i qubit sono operatori unitari, perciò è necessario riscrivere l'azione della funzione f nei termini di un operatore unitario che chiameremo U_f.

Per il caso costante, l'operatore U_f può essere scritto come:

$$U_f = I \otimes X$$

mentre per il caso non costante si ha:

$$U_f = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

Come si può constatare facilmente, l'azione di questi due operatori su qubits che si trovino in stati $|0\rangle$ o $|1\rangle$ corrisponde esattamente all'azione della funzione f definita come:

$$x, y \to (x, y \oplus f(x))$$

Per valutare se f è costante possiamo calcolare prima $U_f(|0\rangle|0\rangle)$ dove lo stato $|0\rangle|0\rangle$ rappresenta i valori di x e y entrambi a 0, poi $U_f(|1\rangle|0\rangle)$ dove mantenedo costante y poniamo $x = 1$. Confronatando i valori delle due espressioni si deduce se la funzione f sia o meno costante.
Se operassimo con U_f direttamente sugli stati $|0\rangle|0\rangle$ e $|1\rangle|0\rangle$ non avremmo nessun vantaggio rispetto all'implementazione classica di questo questo problema, in quanto dovremmo comunque seguire due *chiamate* o run del circuito, una per lo stato $|0\rangle|0\rangle$ e una per lo stato $|1\rangle|0\rangle$.
Se invece dallo stato iniziale $x, y = |0\rangle|0\rangle$ costruiamo lo stato $|+\rangle|-\rangle$, dove i due stati $|+\rangle$ e $|-\rangle$ definiti come:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

e

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

vediamo che è sufficiente eseguire una singola trasformazione U_f, cioè l'analogo di una chiamata ad f per stabilire se la funzione f sia costante oppure no. Infatti si ha che:

$$U_f\left(|+\rangle|-\rangle\right) = \frac{1}{\sqrt{2}}\sum_{x=0}^{1}(-1^{f(x)}|x\rangle|-\rangle)$$

e quindi dopo l'azione di U_f si agisce con l'operatore di Hadamard H su x e poi si misura il valore di x che potrà essere o 0 oppure 1. Nel primo caso la funzione è costante, nel secondo caso la funzione non è costante.

Codice QASM

Nel capitolo 5 abbiamo visto come rappresentare graficamente le trasformazioni quantistiche, vediamo ora che esse possono

anche essere codificate in un linguaggio detto *assembly quantistico* o appunto QASM.
Il QASM è il tentativo di realizzare una sorta di standard per i circuiti quantistici, ma i tempi sono ancora giovani, quindi è meglio aspettare e vedere cosa succederà. Intanto, per gli anni in cui viene scritta questa edizione del testo, il codice qui riportato può essere eseguito su un computer quantistico della IBM o anche sul simulatrore Qiskit, entrambi ambienti accessibili gratuitamente.

```
OPENQASM 2.0;
include "qelib1.inc";
gate nG0 ( param ) q1, q2  {
//***f(q1) ccostante***
//id q1;
//x q2;
//***f(q1) NON-ccostante***
cx q1,q2;
}

qreg q[2];
creg c[1];

id q[0];
x q[1];
h q[0];
h q[1];
nG0(0) q[0],q[1];
h q[0];
measure q[0] -> c[0];
```

Analizziamo il codice. La dichiarazione `gate nG0 ( param ) q1, q2` è analoga alla dichiarazione di una fuzione il cui nome sia `nG0`, che operi sull'argomento definito dai due qubit `q0` e `q1` e che sia parametrizzata da `param`. In questo contesto non

analizziamo il significato di questo parametro che riguarda un argomento più avanzato che non viene discusso in questo testo. In realtà `gate nG0 ( param ) q1, q2` non è una funzione ma un operatore, e rappresenta esattamente l'operatore U_f che abbiamo definito nei conti eseguiti poco sopra.
L'operatore (gate) `gate nG0` è quindi la nostra *black-box* di cui dobbiamo stabilire se l'implementazione sia di tipo costante o non costante.
La dichiarazione `qreg q[2]` inizializza un registro quantistico a due qubit nello stato $|0\rangle$, mentre `creg c[1]` indica un registro classico ad un solo bit usato per registrare il risultato di una misura quantistica.

Esecuzione del programma

L'esecuzione del programma termina con la misura del qubit `q[0]` il cui risultato viene memorizzato sul registro classico e risulta quindi essere un consueto bit di un sistema informatico convenzionale (un PC per esempio). Valutando il valore di c si può conoscere il risultato della computazione quantistica.
Se `c` risulta essere uguale a 0 la funzione f è costante, mentre, se risulta uguale ad 1, la funzione non è costante.
Per provare il codice e testare le due eventualità, si commenti l'una o l'altra sezione della definizione del gate nG0.

Conclusioni

Concludiamo questo testo con una domanda:

La computazione quantistica prenderà mai il posto di quella classica?

Probabilmente questa domanda sta spaventando molti, sia tra i programmatori che tra i grandi della produzione industriale. È ovvio che l'idea che una nuova tecnologia possa prendere il posto di quella a cui si è dedicato energie e risorse non sia piacevole.
Penso però che chi si occupa di informatica classica possa stare relativamente tranquillo.
Come scrisse Landau, la formulazione stessa della Meccanica Quantistica è intrinsecamente impossibile senza l'inclusione della meccanica classica, e penso che lo stesso valga per la computazione quantistica.
L'informatica quantistica sfrutta caratteristiche della fisica che non sono apprezzabili nel mondo classico in cui vive l'essere umano, per questo credo sarà necessario anche in futuro disporre di sistemi informatici classici per *trasportare* i risultati quantistici nel mondo classico.

Tirando le somme, è probabile che la computazione quantistica si svilupperà molto in futuro, ma anche che l'informatica classica rimarrà al suo fianco, allo stesso modo in cui la fisica classica continua a servire da supporto a quella quantistica.

Come approfondire gli argomenti

Gli argomenti che sono stati trattati in questo testo sono tutti piuttosto complessi e meritevoli ognuno di uno studio dedicato.
Chi fosse interessato ad approfondirli per completare la propria conoscenza può iniziare leggendo i libri e gli articoli riportati nel capitolo finale della bibliografia. In base alla propria

preparazione di partenza potrà trovare i riferimenti citati come utili o difficoltosi, in ogni caso costituiscono un punto di partenza.
Buon proseguimento.

Risposte alle domande

In alcuni e-reader, la numerazione delle risposte possibili alle domande compare come 1, 2, 3 anziché a), b), c). In quel caso, si interpretino le risposte: a come 1, b come 2 e c come 3.

- **1** - a.
- **2** - b.
- **3** - b.
- **4** - a.
- **5** - b.
- **6** - a.
- **7** - a.
- **8** - b.
- **9** - c.
- **10** - c.
- **11** - c.
- **12** - c.
- **13** - b.
- **14** - a.
- **15** - a.
- **16** - c.
- **17** - a.
- **18** - a.
- **19** - b.
- **20** - a.