Quantum Computing

Fabio Ortolani

19 marzo 2018

Indice

1 Elementi storici introduttivi

L'inizio del XX secolo è stato caratterizzato da una profonda crisi della fisica conosciuta in tale epoca (ora la indichiamo con il termine "fisica classica") in quanto portava a predizioni assurde a livello microscopico per atomi e molecole. Però, assumendo dapprima ipotesi ad hoc, la crisi terminò all'inizio degli anni '20 (dopo una trentina d'anni di turbolenze) con lo sviluppo della moderna teoria della **meccanica quantistica**. Da allora è una parte indispensabile della scienza (non solo della fisica) ed è stata applicata praticamente a tutto ciò che avviene in natura. Possiamo dire che la meccanica quantistica è un insieme di regole matematiche per la costruzione di teorie fisiche.

Le regole della meccanica quantistica sono semplici, ma anche gli esperti in materia le trovano controintuitive e gli inizi della teoria dell'informazione e computazione quantistica si possono trovare nella continua ricerca di una maggiore comprensione della meccanica quantistica. Lo stesso Einstein manifestò sempre un disagio (che si portò fino alla tomba) con la teoria che aveva contribuito a costruire. Generazioni di fisici hanno continuato a lottare con la meccanica quantistica nello sforzo di rendere le sue previsioni più appetibili. Uno degli scopi della computazione quantistica è anche quello di sviluppare strumenti che possano raffinare la nostra intuizione e rendere le previsioni quantistiche più trasparenti per la mente umana. Siamo ancora agli inizi, speriamo bene!

Passiamo ora ad un altro trionfale successo intellettuale del XX secolo, la scienza del computer (**computer science**). Le origini si perdono nella storia. Ad esempio le tavole cuneiformi suggeriscono che ai tempi di Hammurabi ($\approx$ 1750 BC) i babilonesi avevanoo svilupato una idea abbastanza sofisticata di algoritmo, e forse alcune di tali idee risalgono a tempi antecedenti.

La versione attuale, moderna, di computer science nasce per opera del matematico **Alan Turing** nel 1936. Turing sviluppò in dettaglio una nozione astratta di ciò che ora chiamiamo un computer programmabile, un modello puramente matematico noto come **macchina di Turing**. Possiamo dire che Turing sviluppò con precisione il concetto di **algoritmo**. Manca attualmente una definizione formale di algoritmo, possiamo solo dire (informalmente) che un algoritmo è *una sequenza ordinata e finita di passi (operazioni o istruzioni) elementari che risolve un determinato problema in un tempo finito.* La macchina di Turing è un modello computazionale costituito da un insieme finito di *stati*, un *nastro* infinito contenente simboli da un alfabeto finito che possiamo leggere e scrivere con una testina mobile, e una funzione di transizione che in base allo stato e simbolo attuali, specifica l'azione (scrittura, lettura,

movimento) e lo stato successivo. Essa spiegava perfettamente il significato di risolvere un problema matematico tramite un algoritmo. Cioè, *un problema computazionale può essere risolto con un qualsiasi pezzo di hardware (leggi un moderno computer), se e solo se può essere risolto con una macchina di Turing.* Questa affermazione, nota come **tesi di Church-Turing**, in onore di Turing e di un altro pioniere della computer science, **Alonzo Church**, costituisce una assunzione (cioè una congettura) finora non contraddetta dalla pratica, che è alla base della attuale teoria computazionale.

Se crediamo all tesi di Church-Turing, allora una funzione è calcolabile mediante una macchina di Turing se e solo se è calcolabile mediante un qualche device realistico di calcolo. In effetti, il termine tecnico *calcolabile* corrisponde a ciò che può essere calcolato su una macchina di Turing. Per capire meglio l'intuizione dietro la tesi, consideriamo un qualche altro dispositivo di calcolo, A, che abbia una descrizione in termini finiti, accetti una stringa di input x, ed abbia accesso ad uno spazio di lavoro arbitrario. Possiamo scrivere un codice per la macchina di Turing che simuli l'evoluzione di A in conseguenza dell'input x. In alternativa, possiamo simulare l'evoluzione logica di A (come quando un sistema operativo di un computer simula un altro), oppure più semplicemente, data una descrizione fisica del sistema finito A e delle leggi che lo governano, la macchina di Turing potrebbe calcolare, cioè simulare, il suo comportamento.

Poco tempo dopo la pubblicazione del lavoro di Turing furono costruiti i primi computer con componenti elettronici. **John von Neumann** ideò (assieme ai suoi collaboratori) come mettere assieme in maniera pratica tutti i componenti elettrici per avere un computer equivalente alla macchina di Turing. La sua architettura (un'unità di controllo, un'unità aritmetica e logica, memoria e una unità di input/output, tutte collegata da linee di trasporto dei dati, il BUS) è ancora sostanzialmente alla base dei computer attuali. Lo sviluppo dell'hardware ebbe una accelerazione con la scoperta di **John Bardeen**, **Walter Brattain**, **Will Shockly**, del **transistor** nel 1947. L'hardware è cresciuto come potenza ad un ritmo impresssionante e la crescita fu codificata nel 1965 da **Gordon Moore**: *la potenza di calcolo raddoppia ogni due anni a parità di costo.* Sorprendentemente la legge di Moore si è mantenuta valida per decadi a partire dal 1960. Però molti osservatori si aspettano che questo sogno cessi nei primi due decenni del XXI secolo. L'approccio convenzionale nella attuale fabbricazione di componenti comincia ad avere difficoltà con le dimensioni spaziali sempre più piccole. Gli effetti quantistici iniziano ad interferire col funzionamento dei dispositivi elettronici.

Una soluzione al problema posto dal fallimento finale della legge di Moore

potrebbe essere quello di cambiare paradigma di calcolo. Un paradigma alternativo potrebbe essere dato dalla computazione quantistica, usare la meccanica quantistica invece della meccanica classica (come implicitamente supposto per i device di calcolo realistici pensati da Turing), per eseguire i calcoli. Un computer classico può essere usato per simulare un fenomeno quantistico (e quindi anche un computer quantistico), ma sembra che non sia possibile farlo in maniera efficiente. Ciò fa ritenere che i computer quantistici possano offrire un vantaggio essenziale di velocità rispetto ai loro analoghi classici. Alcuni ricercatori pensano che nessun progresso nella computazione classica riesca a superare il gap di potenza computazionale di un computer quantistico.

2 Efficienza

La versione originale della tesi di Church-Turing non dice nulla riguardo all'efficienza di un calcolo o di una simulazione. Spesso siamo interessati o preoccupati riguardo all'ammontare delle risorse usate da un computer per risolvere un problema e ci riferiamo a questo con il termine di **complessità** del calcolo. Una risorsa importante è il **tempo** necessario per il calcolo. Un'altra risorsa è lo **spazio**, cioè l'ammontare di memoria usata. Si esprime di solito l'ammontare di una risorsa necessaria tramite una funzione f della lunghezza n dell'imput di una istanza del problema. Ad esempio, se abbiamo il problema di moltiplicare due numeri con n bits, un dispositivo potrebbe usare fino a $f(n) = 2n^2+n+5$ unità di tempo (dove l'unità di tempo potrebbe essere un secondo o il tempo necessario al dispositivo per compiere un passo elementare, il "ciclo di clock").

Certamente l'ammontare delle risorse necessarie dipende dall'architettura della macchina usata e con un atro dispositivo lo stesso problema potrebbe richiedere $f(n) = 5n^3+n$ unità di tempo per essere eseguito. In genere le operazioni più semplici sono ottimizzate in maniera simile su tutti i dispositivi, per cui si esegue una stima generale valida per la maggioranza dei dispositivi. Inoltre si usa una misura più grossolana, specificando solo i termini di ordine più alto scrivendo $O(n^2)$ (oppure $O(n^3)$ nel secondo caso).

Notiamo che la scrittura $O(f(n))$ indica un limite superiore della risorsa (il tempo o lo spazio). Se si vuole indicare un limite inferiore usiamo la notazione $\Omega(f(n))$.

Possiamo essere ancora più grossolani e vale la convenzione che un algoritmo o una simulazione è **efficiente** inei riguardi di una particolare risorsa se l'ammontare è valutato in $O(n^k)$ per qualche potenxa k. Diciamo anche in questo caso che l'algoritmo è **polinomiale**. Algoritmi che invece sono $\Omega(c^n)$,

con c una costante, è detto **esponenziale**, ed è considerato **non efficiente**. Se la risorsa non è limitabile da sopra con nessun polinomio, diremo che è **sovrapolinomiale**. Il termine esponenziale è generalmente usato al posto di sovrapolinomiale.

Per poter estendere la tesi di Church-Turing e dire qualcosa di utile riguardo all'efficienza è necessario modificare la macchina di Turing aggiungendo la possibilità di fare una scelta random ad ogni step. Esistono alcuni importanti problemi che possono essere risolti in maniera efficiente usando una **macchina di Turing probabilistica** dove le regole di transizione tra stati tengono conto di una variabile random. Un esempio è quello di trovare una radice quadrata modulo un numero primo. Sembra strano che l'aggiunta di un fattore casuale possa migliorare l'efficienza, ma così è stato comprovato in vari casi. Non ci sono prove effettive che una macchina di Turing probabilistica sia equivalente ad una macchina di Turing deterministica (anche se esiste il sospetto), per cui si è modificata di l'affermazione della tesi di Church-Turing.

(Classical) Strong Church-Turing Thesis: *Una macchina di Turin probabilistica può simulare effficientemente ogni modello realistico di calcolo.*

La versione forte della tesi è sopravissuta a così tanti tentativi di di violarla che è diventata ampiamente accettata. Possiamo chiederci perchè la parola "realistico" appaia nell'affermazione. La parola indica un modello di computazione che sia consistente con le leggi della fisica e in cui teniamo conto esplicitamente di tutte le risorse fisiche usate.

Un problema fondamentale della tesi forte di Church-Turing è dato dal fatto che le leggi della fisica classica non sono abbastana potenti da simulare in maniera efficiente la fisica quantistica. La tendenza di pensiero tra i teorici della computer science è che il principio sia ancora vero, ma sia necessario un modello in grado di simulare qualsiasi device fisico realistico, incluso un device quantistico. La risposta può essere una versione quantistica della tesi forte di Church-Turing, dove una macchina di Turing probabilistica è rimpiazzata da un ragionevole modello di calcolo quantistico, che indichiamo come **macchina di Turing quantistica**. Possiamo congetturare la seguente tesi:

(Quantum) Strong Church-Thesis: *Una macchina di Turing quantistica può simulare efficientemente ogni modello realistico di calcolo.*

Motivato da tale problema **David Deutsch** nel 1985 fu un pioniere nella costruzione di tale modelllo considerando dispositivi di calcolo basati sui principi della meccanica quantistica e formulando la versione quantistica della tesi. Notiamo di nuovo che tutte le versioni della tesi di Church-Turing sono congetture e non sono attualmente dimostrate.

3 Crcuiti

Le macchine di Turing sono modelli molto idealizzati di dispositivi di calcolo. I computer reali sono prima di tutto finiti come dimensioni (di memoria in particolare), mentre Turing ipotizzava una memoria (il nastro) infinita. Conviene utilizzare un modello alternativo di computazione, il **modello circuito**, equivalente al modello di Turing in termini di potenza dicalcolo, ma più conveniente e realistico per molte applicazioni.

Il modello è basato sull'assunzione di particolari dispositivi (gates) che implementano il calcolo di una specifica funzione f che trasforma un dato input x nell'output $y = f(x)$. L'input è rappresentato da un numero in binario espresso da n bits (rappresentato graficamente da linee di ingresso) che possono assumere i valori 0 e 1 (stati **false** e **true**), e viene prodotto in maniera univoca un set di m bit che esprimono l'ouput $y = f(x)$. In pratica un gate implementa l'azione di una funzione:

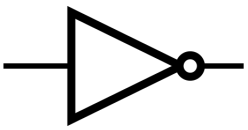
$$\begin{aligned} f \; &: \{0,1\}^n \longrightarrow \{0,1\}^m \\ x = (x_1, \ldots, x_n) &\longmapsto f(x) = (f_1(x), \ldots, f_m(x)) \,. \end{aligned} \tag{1}$$

Chiaramente una funzione con m bit di output è equivalente a m funzioni con 1 bit solo in uscita, per cui potremmo dire che il compito basilare è il calcolo di una **funzione Booleana** (cioè a soli due valori in uscita, falso o vero):

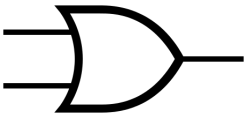
$$f \; : \; \{0,1\}^n \longrightarrow \{0,1\} \,. \tag{2}$$

La cardinalità del dominio è finita: 2^n, e i modi di associare ad ogni input uno dei due valori possibili in uscita sono 2^{2^n}. Il numero di possibili funzioni booleane distinte è effettivamente enorme! Con $n = 5$ abbiamo $2^{32} \simeq 4.3 \cdot 10^9$ - sicuramente nella vostra vita ne avete incontrate solo una piccola parte. Considerazioni pratiche richiedono che un dispositvo di calcolo sia costruito usando gates da un piccolo set di gates elementari. Un insieme di gates elementari è detto **universale** se ogni funzione booleana può essere costruita concatenando in maniera opportuna l'azione dei soli gate dell'insieme. In Fig. 1 sono rappresentati alcuni importanti gates di base.

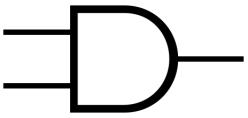
Il modello a circuito modellizza l'implementazione di un algoritmo o una simulazione con un circuito formato da un insieme di linee che contengono e trasportano (spazialmente e temporalmente) i bits e collegano i gates (rappresentati graficamente da box) che eseguono operazioni elementari sui bits. I circuiti che consideriamo sono **aciclici**, cioè i bits fluiscono in maniera lineare in un unico verso e non ritornano in ingresso ad un gate usato in precedenza. Convenzionalmente i bits di input entrano nel circuito dal



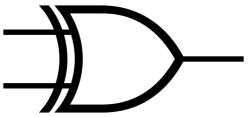
NOT: $x \longmapsto \overline{x} = \neg x = 1 - x$



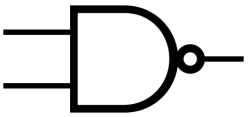
OR: $(x, y) \longmapsto x \vee y = x + y - xy$



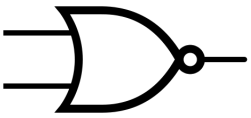
AND: $(x, y) \longmapsto x \wedge y = xy$



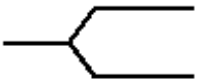
XOR: $(x, y) \mapsto x \oplus y = x + y - 2xy$ (addizione mod 2)



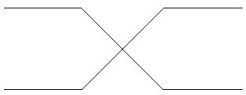
NAND: $(x, y) \longmapsto \neg(x \wedge y) = 1 - xy$



NOR: $(x, y) \longmapsto \neg(x \vee y) = 1 - x - y + 2xy$



FANOUT: $x \longmapsto (x, x)$



SWAP: $(x, y) \longmapsto (y, x)$

Figura 1: Gates elementari comunemente usati. $x, y \in \{0, 1\}$

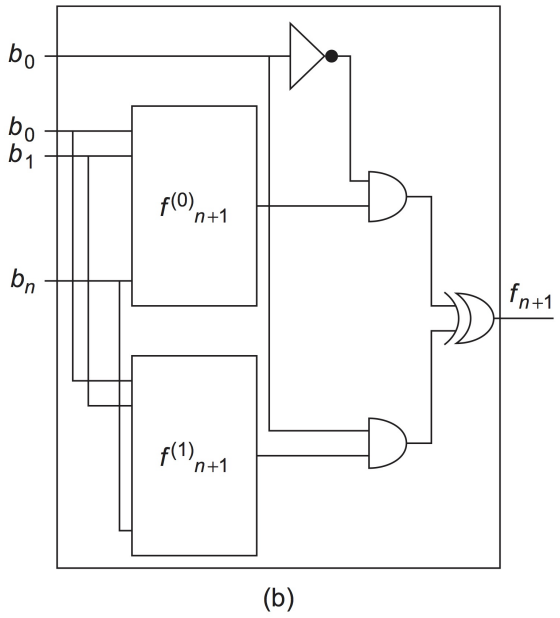
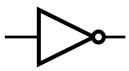


Figura 2: Costruzione di un gate a $n+1$ bits.

lato sinistro del diagramma. Ad ogni step temporale t ogni linea può entrare come input di un solo gate e mai tornare indietro. I bits di ouput che esprimono il risultato dell'algoritmo vengono letti sulle linee che escono dal circuito sul lato destro. Sostanzialmente un problema computazionale viene rappresentato tramite una opportuna funzione (1) e implementata tramite un opportuno circuito che costituisce un gate (in generale non semplice) che fornisce il risultato in un lasso di tempo ragionevole (si spera!).

Possiamo vedere subito che è possibile utilizzare pochi gates. Consideriamo il caso semplificato (2) con $m = 1$. Con $n = 1$ le possibili funzioni $\{0,1\} \longrightarrow \{0,1\}$ sono 4:

$$\begin{aligned} &\text{identità } f(a) = a \\ &\text{NOT } f(a) = \overline{a} \qquad \text{  } \\ &\text{set 0 } f(a) = 0 = a \wedge \overline{a} \\ &\text{set 1 } f(a) = 1 = a \vee \overline{a} \end{aligned} \tag{3}$$

e solo il gate NOT è considerato effetiivamente di base. Per induzione possiamo considerare una funzione di $n+1$ bits assumendo l'esistenza di circuiti

(o gates) per funzioni a n bits. Se

$$f(x_0, x_1, \ldots, x_n) = f(x_0, x) \qquad x = (x_1, \ldots, x_n) \in \{0,1\}^n \,, \tag{4}$$

definiamo:

$$\begin{aligned} f_0(x_1, \ldots, x_n) &= f_0(x) = f(0, x) \\ f_1(x_1, \ldots, x_n) &= f_1(x) = f(1, x) \end{aligned} \tag{5}$$

e associamo ad esse due gates opportuni (per ipotesi essi esistono). Allora essendo tutte le funzioni di tipo Booleano abbiamo:

$$f(x_0, x) = (\overline{x_0} \wedge f_0(x)) \oplus (x_0 \wedge f_1(x)) \,, \tag{6}$$

e abbiamo aggiunto le sole operazioni logiche NOT, AND e XOR (e, anche se di solito non viene considerato, il FANOUT per duplicare su un'altra linea un bit). Vedi Fig. 2.

È stato dimostrato che un tale modello computazionale è equivalente ad una macchina di Turing universale e viene attualmente preferito avendo una corrispondenza più diretta con gli attuali calcolatori. L'hardware di un computer utilizza un numero finito di gates più o meno semplici per eseguire task di base e il programmatore stabilisce come combinare e organizzare tali gates elementari.

4 Formulazione algebrica del modello circuito

Possiamo formulare il modello circuito tramite l'algebra matriciale. Questo è un approccio insolito nella computer science classica ma rende la transizione alla formulazione standard della computazione quantistica più diretta.

Supponiamo di avere una descrizione (come il diagramma nella Fig. 3, e una specifica di alcuni bit di ingresso. Se vogliamo predire l'output del circuito, possiamo seguire il flusso dei dati da sinistra a destra, aggiornando il valore dei bits immagazzinati nelle varie linee all'uscita dei gate. In altre parole seguiamo lo "stato" dei bits mentre progrediscono lungo il circuito. In un punto del circuito possiamo intendere lo stato delle linee come lo "stato del computer".

Lo stato in un circuito *deterministico* può essere specificato da una lista dei valori dei bit su ogni linea, una seguenza di 0 e 1. Se consideriamo un circuito *probabilistico* questa descrizione non è più sufficiente. In un computer probabilistico un singolo bit può assumere il valore 0 con probabilità p_0 o il

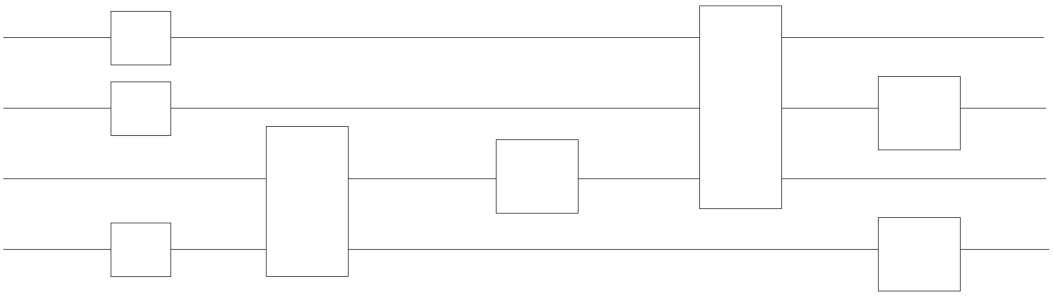


Figura 3: Un circuito classico formato da vari gates.

valore 1 con probabilità $p1$. Possimo indicare questa informazione con un vettore bidimensionale:

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} . \tag{7}$$

Notiamo che la notazione è valida anche per un computer deterministico. Lo stato 0 di un bit si ha quando $p_0 = 1$ e $p_1 = 0$:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} , \tag{8}$$

e analogamente, il valore 1 di un bit è rappresentato dalle probabilità $p_0 = 0$, $p_1 = 1$:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} . \tag{9}$$

Avendo scelto di rappresentare lo stato di una linea (e di una collezione di linee) con un vettore, possiamo esprimere l'azione di un gate come un operatore che agise sui vettori in maniera appropriata. Questi operatori possono essere rappresentati come matrici. Si consideri il gate NOT. Dobbiamo definire una matrice che agisce sui vettori di stato in maniera consistente con l'azione del gate. Sappiamo che il gate NOT trasforma il valore 0 ($p_0 = 1$) in 1 ($p_1 = 1$) e viceversa:

$$NOT \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} , \qquad NOT \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} . \tag{10}$$

Pertanto, se abbiamo familiarità con l'algebra matriciale, il NOT è rappresentato dalla matrice:

$$NOT \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} . \tag{11}$$

Per applicare il gate NOT allo stato di una linea moltiplichiamo la matrice del gate NOT al vettore di stato:

$$NOT \begin{pmatrix} p0 \\ p1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_0 \end{pmatrix} . \tag{12}$$

Supponiamo ora di voler descrivere una coppia di linee in un circuito probabilistico. Il primo bit sia nello stato 0 con probabilità p_0 o nello stato 1 con probabilità p_1. Analogamente al secondo bit siano associate le probabilità q_0 per lo stato 0 e q_1 per lo stato 1. Le quattro possibilità per lo stato combinato delle due linee sono $\{00, 01, 10, 11\}$. Le probabilità associate a tali stati si ottengono moltiplicando le probailità corrispondenti e descritte dal vettore 4-dimensionale:

$$\begin{pmatrix} p_0 q_0 \\ p_0 q_1 \\ p_1 q_0 \\ p_1 q_1 \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \end{pmatrix} \otimes \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} \tag{13}$$

che può essere espresso matematicamente come prodotto tensoriale degli stati delle due linee singole.

Possiamo rappresentare tramite matrici anche i gate agenti sulla coppia di linee. Consideriamo il cosidetto gate **controlled-NOT**, CNOT (vedi Fig. 4). Questo gate agisce su due bit, detti *control* bit e *target* bit. L'azione del gate è quello di applicare l'operazione NOT al bit target se il bit di controllo è 0, e non fare nulla se se il bit di controllo è 0. Equivalentemente se il bit di controllo vale c e il bit target vale t l'azione del CNOT è:

$$NOT \ : \ (c, t) \longmapsto (c, c \oplus t) . \tag{14}$$

Questa azione può essere rappresentata dalla matrice:

$$CNOT \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{15}$$

Ad esempio, se il primo bit è 1 e il secondo è 0:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} , \tag{16}$$

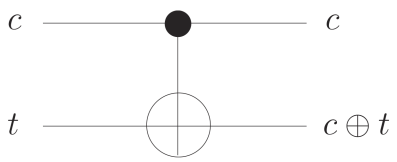


Figura 4: Rappresentazione del gate c-NOT.

allora il risultato del'applicazione CNOT è:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \tag{17}$$

come può essere facilmente verificato.

Notiamo che se il bit di controllo è nello stato probabilistico:

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

e il bit target è settato a 0:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

otteniamo col'azione del CNOT lo stato:

$$\begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

che non può essere fattorizzato come prodotto tensoriale di due stati probabilistici indipendenti. Abbiamo creato una *correlazione* (statistica) tra i due bits (sono entrambi 0 o entrambi 1 con uguale probabilità 50%).

5 Computazione reversibile

Nel formulare un modello di computazione quantistica seguiremo la strada di generalizzare il modello circuito della computazione classica. Un punto

importante è però che le trasformazioni quantistiche (i gates) sono tutte trasformazioni unitarie (lo stato viene fatto evolvere temporalmente in maniera opportuna). I gates quantistici sono quindi invertibili per costruzione.

A parte quello di fornire un ponte tra la computazione classica e quella quantistica, la computazione reversibile ha un interesse fisico in sè. Nel 1961 **Rolf Landauer** enunciò un principio, noto da allora come **principio di Landauer**. Egli osservò che la cancellazione di informazione è necessariamente un processo dissipativo. Il suo ragionamento è che la cancellazione implica sempre una compressione dello spazio delle fasi, anche da un punto di vista termodinamico, e risulta logicamente irreversibile.

Per esempio possiamo immaginare un bit di informazione piazzando una molecola in una scatola divisa in due parti da una barriera, ponendola o a sinistra (stato 1) o a destra (stato 0). La cancellazione significa che muoviamo la molecola nella parte destra indipendemente dalla sua posizione iniziale, ad esempio rimuovendo la barriera e iniziando a comprimere con un pistone il "gas" monomolecolare finchè la molecola è nella parte destra in maniera definitiva. Questa operazione cambia l'entropia del "gas" di $\Delta S = -k \ln 2$ (k costante di Boltzmann) e un associato flusso di calore viene trasferito dalla scatola all'ambiente. Se il processo è quasi-statico ed isotermico ad una temperatura T si deve eseguire un lavoro

$$W = kT \ln 2 \tag{18}$$

sulla scatola. Se distruggo dell'informazione qualcuno deve pagare la tassa!

Landauer formulò quindi il principio che la quantità precedente (18) rappresenta l'ammontare minimo di energia necessaria per "cancellare" un bit di informazione. A temperature ordinaririe ($\sim 300^o K$) questo limite di Landauer corrisponde a circa 0.0178 eV che è molto piccolo rispetto alle attual1 energie utilizzate da una porta logica in ogni operazione (ci sono tre ordini di grandezza di differenza). Estrapolando con la legge di Moore si prevede di raggiungere il limite di Landauer tra il 2030 e il 2035. Se un computer operasse con una logica reversibile allora in linea di principio non ci sarebbe dissipazione e nessuna potenza necessaria. Possiamo fare calcoli gratis!. Secondo **Charles Bennett** l'unico modo di superare il limite di Landauer in futuro è con una computazione reversibile o quantistica.

6 Gates reversibili

Un computer reversibile deve usare gates reversibili che necessariamente devono trasformare n bits di input in n bits di output:

$$f \ : \ \{0,1\}^n \ \longrightarrow \ \{0,1\}^n \tag{19}$$

e l'invertibilità richiede un unico input per ogni output. Possiamo eseguire un "run inverso" per recuperare l'input dall'output. Dovendo avere una funzione che trasformi ogni possibile input in maniera biunivoca e suriettiva nel corrispondente output, possiamo considerare tale funzione come una permutazione delle 2^n stringhe di bits di possibili input. Abbiamo quindi $(2^n)!$ funzioni invertibili possibili. In contrasto, se non richiediamo l'invertibilità avremo $(2^n)^{2^n}$ funzioni possibili che trasformano n bits in n bits in generale. Pensando 2^n grande e ricordando l'approssimazione di Stirling:

$$(2^n)! \sim (2^n)^{2^n} e^{-2^n} ,$$

vediamo che la frazione di funzioni invertibili è molto piccola.

Notiamo però che ogni computazione irreversibile (e non) pùo essere "impacchettata" nel calcolo di una funzione reversibile (aggiungendo eventualmente dei bits all'input). Ad esempio se:

$$f \ : \ \{0,1\}^n \ \longrightarrow \ \{0,1\} \tag{20}$$

è una generica funzione Booleana possiamo costruire una nuova funzione:

$$\begin{aligned} &\tilde{f} \ : \ \{0,1\}^{n+1} \ \longrightarrow \ \{0,1\}^{n+1} \\ &\tilde{f}(x,y) = (x, y \oplus f(x)) , \qquad x \in \{0,1\}^n , \quad y \in \{0,1\} , \end{aligned} \tag{21}$$

L'input x viene preservato e l'ultimo bit y viene invertito se $f(x) = 1$. Applicando la trasformazione $\tilde{f}$ ulteriormente il bit y viene ripristinato (indipendentemente dal valore di $f(x)$), per cui $\tilde{f}$ é invertibile e coincide con la sua inversa. Se poniamo $y = 0$ in in input, l'ultimo bit dell'output contiene esattamente $f(x)$.

Come per le funzioni booleane precedenti, possiamo chiederci se una complicata computazione reversibile può essere costruita con un circuito formato da componenti semplici - cioè esiste una famiglia universale di gate reversibili?

Dei quattro possibili gates da 1 bit a 1 bit, due sono reversibili: l'identità (la linea stessa) e il gate NOT.

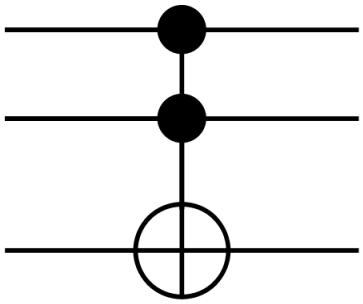


Figura 5: Il gate di Toffoli.

Considerando i possibili gates da 2 bits a 2 bits (in totale $4^4 = 256$) quelli invertibili sono $4! = 24$. Tra questi risulta particolarmente importante il gate CNOT visto in precedenza:

$$CNOT \;:\; (x, y) \longmapsto (x, x \oplus y)\,, \tag{22}$$

che implementa anche un gate XOR (ignorando l'output x). Anticipando la notazione usata per i gates quantistici lo indichiamo anche con $C(X)$ (X è una notazione usata per il NOT quantistico – X è la matrice di Pauli). In generale indichiamo con $C(G)$ un gate che applica G a un "target" condizionato al valore di un bit di controllo: G è applicato se il bit di controllo è 1, altrimenti si applica l'identità.

Nel caso di computazione classica non reversibile è noto che ogni circuito può essere costruito utilizzando in gate NAND, con 2 bit di ingresso e 1 di uscita, non invertibile. Se vogliamo usare solo gates reversibili, si può dimostrare che con i soli gates a 1 e 2 bits non è possibile costruire tutti i gate a n-bits reversibili con $n > 2$. Se consideriamo i gate reversibili a 3-bits o più, un gate particolarmente importante è il **gate di Toffoli** (Fig. 5) $C^2(X)$ detto anche **controlled-controlled-CNOT** (prende il nome dal suo autore, l'italiano **Tommaso Toffoli**):

$$C^2(X) \;:\; (x, y, z) \;\longmapsto\; (x, y, (x\,y) \oplus z)\,. \tag{23}$$

Esso "flippa" il terzo bit solo se i primi due son 1, non fa nulla altrimenti. La notazione $C^2(\cdot)$ indica che l'operazione sul target è triggerata da due bit di controllo che devono essere settati a 1. Come $C(X)$ anche $C^2(X)$ coincide col suo inverso.

Il gate di Toffoli può servire come gate universale per una logica booleana, a patto di poter usare dei bit aggiuntivi di servizio (a volte settati a 0 e 1) e ignorare bits aggiuntivi di output.

Se poniamo $x = y = 1$ il gate esegue un NOT:

$$C^2(X) \ : \ (1,1,z) \ \longmapsto \ (1,1,1 \oplus z) = (0,0,\neg z)\,. \tag{24}$$

Se settiamo $z = 0$ otteniamo un AND:

$$C^2(X) \ : \ (x,y,0) \ \longmapsto \ (x,y,(x\,y))\,. \tag{25}$$

Settando $z = 1$ abbiamo un NAND:

$$C^2(X) \ : \ (x,y,1) \ \longmapsto \ (x,y,(x\,y) \oplus 1) = (x,y,\neg(x\,y))\,. \tag{26}$$

Con $x = 1,\ z = 0$ abbiamo un FANOUT:

$$C^2(X) \ : \ (1,y,0) \ \longmapsto \ (1,y,y)\,. \tag{27}$$

Possiamo quindi affermare che il gate di Toffoli fornisce da solo un gate universale per costruire circuiti classici e circuiti reversibili (e simulare circuiti cassici con circuiti reversibili). Questo però in generale richiede l'uso di bits aggiuntivi (posti su **linee ancella**) e di ignorare bits di output. Ci ritroviamo con dei bits **junk** che occupano spazio in memoria. Questa, col progredire del calcolo, si riempie gradualmente di junk fino ad arrivare ad uno stadio in cui non possiamo continuare il calcolo se non cancelliamo qualche bit e facciamo spazio. In tale stadio della procedura dobbiamo pagare la tassa energetica del principio di Landauer.

Possiamo però fare in modo che il junk addizionale sia cancellato alla fine del calcolo prima copiando l'output (in maniera reversibile) ed poi eseguire il circuito (reversibile) inverso, cancellando i bit di junk al loro stato originario (possiamo supporre che i bit extra siano inizialmente settati a 0. Questo schema è illustrato in Fig. 6.

Abbiamo ottenuto una implementazione reversible che opera nella maniera:

$$(x,c) \ \longmapsto \ (x, c \oplus f(x))\,, \tag{28}$$

che con $c = 0$ fornisce in maniera reversibile $f(x)$ al costo di un overhead di memoria e tempo (che si spera modesto) ma senza un limte inferiore di costo energetico. Questo serve da esempio per mostrare che è possibile operare in maniera totalmente reversibile. Si possono ricercare ovviamente tecniche più sofisticate per risparmiare tempo e memoria.

Nella computazione classica si può scegliere di essere amichevoli con l'ambiente ripristinando l'informazione junk ridondante. In ogni caso il buttare via o cancellare non cambia l'output del calcolo. In una computazione

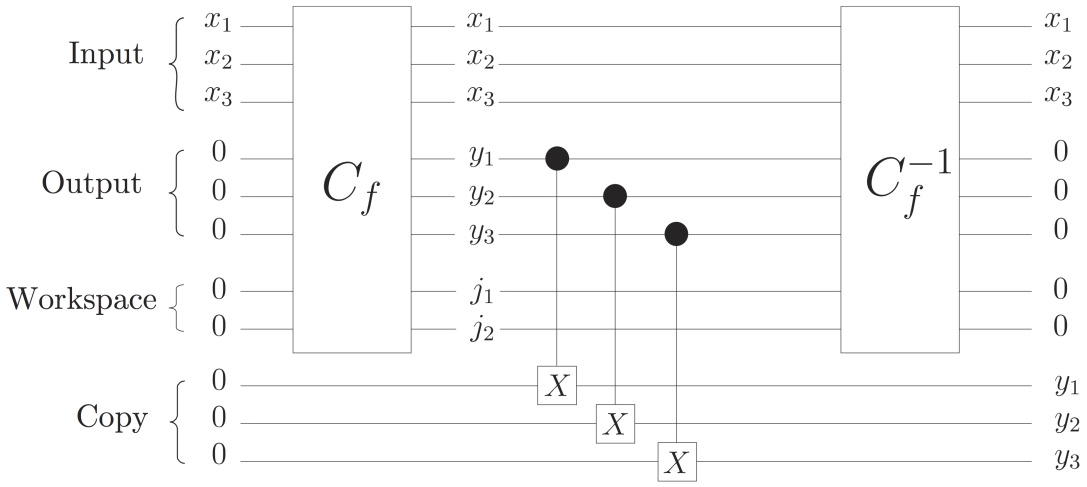


Figura 6: Cancellazione dei bits junk durante il calcolo.

quantistica, "buttare via" delle informazioni che sono strettamente correlate con i bit che manteniamo in vita può drasticamente cambiare l'output. La teoria della computazione reversibile gioca pertanto un ruolo importante nello sviluppo di algoritmi quantistici. In maniera simile a quella classica, le operazioni quantistiche reversibili possono simulare efficacemente operazioni quantistiche non reversibili per cui ci focalizzeremo su quelle reversibili.

7 Circuito quantistico

Nelle sezioni precedenti abbiamo introdotto il modello circuito (classico) prestando attenzione ai circuiti reversibili. Il modello a circuito assume una sua valenza anche nell'ambito della computazione quantistica, in quanto anche in questo caso parliamo di un modello a circuito con la modifica che le linee non contengono un bit classico, ma un qubit quantistico, e ogni gate esprime una trasformazione quantistica sugli stati quantistici dei qubits su cui agisce. I "fili" rappresentati da linee non trasportano bits classici ma qubits e i gates sono anch'essi quantistici. Un gate agisce su n qubits di input e fornisce n qubits di output. Un circuito quantistico è rappresentato in maniera analoga ai circuiti classici mediante un diagramma come quello mostrato in Fig. 7. Le linee orizzontali rappresentano qbits che noi immaginiamo fluire nel tempo da sinistra a destra. I gates sono rappresentati generalmente da blocchi rettangolari (consideriamo solo gate unitari). Generalmente si considera uno stato iniziale di n qbits tutti nello stato $|0\rangle$ o equivalentemente lo

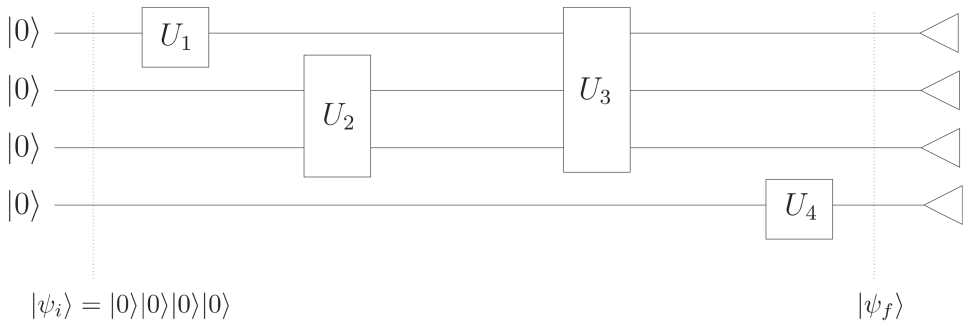


Figura 7: Circuito quantistico.

stato quantistico iniziale è $|\psi_i\rangle = |0\ldots0\rangle = |0\rangle \otimes \ldots \otimes |0\rangle$, e su alcuni qbits alla volta agiscono i gates in sequenza ottenendo alla fine uno stato finale $|\psi_f\rangle$ su cui viene eseguita una misura proiettiva che fornisce un valore con una certa probabilità. In genere tale misura viene fatta sui singoli qubits e lo rappresentiamo con dei triangoli (lo stato viene fatto collassare, cioè proiettato, su uno stato della **base computazionale** $\{|0\rangle, |1\rangle\}^{\otimes n}$). Generalmente lo stato quantistico dopo la misura non interessa (viene scartato) e siamo interessati solo al risultato della misura che può esssere rappresentato con una linea (classica) graficamente diversa.

Notiamo che i processi di computazione quantistica sono molto diversi dalle loro controparti classiche. In un computer classico, noi immettiamo i dati da una tastiera o qualche altro device di input e i segnali vengono inviati ad una porta di I/O del computer e immagazzinati nella memoria, in seguito vengono forniti al microprocessore, i risultati sono di nuovo messi in memoria prima di essere stampati o mostrati sullo schermo. Pertanto l'informazione, generalmente sotto forma di segnali elettrici, viaggia attraverso i circuiti interni del computer. In contrasto, l'informazione in un computer quantistico è immagazzinata in un registro (un sistema fisico opprtuno di qubits) e opportuni campi esterni, elettrici, magnetici o fasci laser vengono applicati per eseguire le operazioni (gates elementari) sul registro. Questi campi esterni sono progettati in modo da operare le trasformazioni desiderate su particolari insiemi di qubits. Pertanto l'informazione risiede stabile nel registro e viene modificata ogni volta che un gate agisce sul registro stesso.

Un'altra distinzione tra una computazione classica e quella quantistica è che mentre la prima è basata completamente su un processo digitale, la seconda su un processo ibrido (digitale + analogico). Un qubit può essere in una sovrapposizione arbitraria degli stati $|0\rangle$ e $|1\rangle$ con coefficienti complessi. La base $\{|0\rangle\ |1\rangle\}$ costituisce l'aspetto digitale, mentre i coefficienti

sono parametri continui, analogici e sono sempre soggetti ad errore. Questi aspetti possono rendere molto difficile una realizzazione pratica di computer quantistico.

Possiamo individuare alcune caratteristiche di un modello di circuito quantistico.

- *Una particolare decomposizione preferita in sottosistemi.* È implicito ma importante che lo spazio di Hilbert del device abbia una decomposizione preferita in un prodotto tensoriale di sottosistemi a bassa dimensionalità, nel nostro caso i qubits, cioè sistemi a due livelli. Potremmo assumere anche sistemi a tre livelli (qutrits) o altro ma gli aspetti teorici sono ancora in larga parte ignoti. In ogni caso assumiamo una decomposizione naturale in sottosistemi che viene rispettata dai gates quantistici. Fisicamente la ragione fondamentale è la località: i gates realizzabili agiscono in una regione spazialmente limitata e il computer si decompone in in sottosistemi che interagiscono solo con i loro vicini.

- *Insieme finito di istruzioni.* Poichè le trasformazioni unitarie formano un continuo, potrebbe sembrare troppo restrittivo assumere che la macchina possa eseguire solo i gates quantistici scelti da un insieme discreo e finito. Accettiamo invece tale assunzione assumendo di poter controllare l'errore provocato da tale scelta. Non vogliamo dover inventare una nuova implementazione fisica ogni volta che affrontiamo un nuovo calcolo.

- *I gates sono trasformazioni unitarie e le misure sono trasformazioni proiettive.* Potremmo ammettere anche gates quantistici non unitari e misure generali, ma queste possono essere realizzate mediante opportune combinazioni di gates unitari emisure proiettive, per cui non perdiamo di generalità.

- *Preparazioni semplici.* Scegliere lo stato iniziale di n qubits come $|0\ 0\ \ldots\ 0\rangle$ è puramente una convenzione. Quello che si assume è che sia semplice preparare uno stato iniziale analogo ad uno stato classico prodotto tensoriale di qubits negli stati $|0\rangle$ o $|1\rangle$. Se ammettesimo come input iniziale uno stato entangled di n qubits nascondiamo le difficoltà di un algoritmo con la difficoltà nella preparazione di uno stato iniziale.

- *Misure semplici.* Possiamo ammettere che la misura finale sia una misura collettiva, cioè una proiezione su una base arbitraria. Ma tale mi-

sura può esseres implementata eseguendo una trasformazione unitaria seguita da una proiezione sulla base "standard":

$$\{|0\rangle\,,\,|1\rangle\}^{\otimes n} \tag{29}$$

(**base computazionale**).

- *Misure fatte solo alla fine.* Potremmo permettere misure intermedie ai vari stadi del calcolo con la possibile scelta di gates condizionati dal risultato di tali misure. Però si può vedere che gli stessi risultati si ottengono con tutte le misure eseguite solo alla fine (rimane valida in pratica l'utilità di poter fare misure intermedie).

Osservazione Un gate quantistico è una trasformazione unitaria, quindi reversibile. Un computer classico reversibile può essere considerato un caso speciale di computer quantistico. Un gate classico reversibile esegue una trasformazione

$$x \;\longmapsto\; f(x)\,, \tag{30}$$

che opera sostanzialmente una permutazione p di stringhe di n bits, che può essere considerata una trasformazione unitaria U, agente su n qubits, che mappa la base computazionale di stati prodotti tensoriali

$$\{|x_j\rangle\,, j = 0, 1, \ldots, 2^n - 1\} \tag{31}$$

ad un'altra base sempre di stati prodotti tensoriali

$$U|x_j\rangle = |y_j\rangle = |x_{p(j)}\rangle\,. \tag{32}$$

permutazione della base di partenza. Una computazione quantistica che utilizzi la versione quantistica di tali gate mapperebbe ad esempio lo stato $|0\ 0\ \ldots\ 0\rangle$ in un singolo stato della base computazionale e una misura finale dei singoli qubits sarebbe deterministica.

8 Gates quantistici

8.1 1-Qubit gates

I gates quantistici 1-qubit esprimono trasformazioni unitarie su uno spazio di Hilbert bidimensionale generato dagli stati $|0\rangle$ e $|1\rangle$:

$$|\psi\rangle \;\longmapsto\; U\,|\psi\rangle \tag{33}$$

e U può essere rappresentato da una matrice 2×2. Una "base" di matrici è data dalle **matrici di Pauli** che definiscono i corrispondenti **gates di Pauli**:

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{34}$$

Tutte le matrici hanno per quadrato l'identità:

$$\mathbb{1}^2 = X^2 = Y^2 = Z^2 = \mathbb{1}\,. \tag{35}$$

Nota Se $A^2 = \mathbb{1}$, allora per ogni ϑ vale:

$$e^{i\,\vartheta\,A} = \cos(\vartheta)\,\mathbb{1} + i\,\sin(\vartheta)\,A\,. \tag{36}$$

Possiamo introdurre le matrici (gates) di rotazione:

$$\begin{aligned} R_x(\theta) &\equiv e^{-i\frac{\theta}{2}X} \\ R_y(\theta) &\equiv e^{-i\frac{\theta}{2}Y} \\ R_z(\theta) &\equiv e^{-i\frac{\theta}{2}Z}\,. \end{aligned} \tag{37}$$

Se scriviamo uno stato generico (normalizzato) di un qubit come:

$$|\psi\rangle = |0\rangle\,\cos(\frac{\alpha}{2}) + |1\rangle\,\sin(\frac{\alpha}{2})\,e^{i\varphi}\,, \tag{38}$$

che identifica un punto di coordinate polari (α, φ) sulla sfera di Bloch, possiamo verificare che $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$, rappresentano (a meno di un fattore di fase globale) rotazioni di un angolo θ attorno agni assi x, y, z rispettivamente. In particolare X, Y, Z sono rotazioni di π attorno ai corrispondenti assi.

Il gate di Pauli X agisce sulla base come:

$$\begin{cases} X\,|0\rangle = |1\rangle \\ X\,|1\rangle = |0\rangle \end{cases} \tag{39}$$

per cui corrisponde all'operazione logica NOT (che viene così indicato con X).

Una qualsiasi trasformazione unitaria sugli stati di 1 qubit può essere identificata (a meno di un fattore di fase) con una rotazione sulla sfera di Bloch e abbiamo diversi modi di esprimere tale rappresentazione. Le principali sono riassunte nei teoremi seguenti di cui omettiamo le dimostrazioni.

Theorem 8.1. *Sia U un gate unitario a 1-qubit. Allora esistono dei numeri reali α, β, γ e δ tali che:*

$$U = e^{i\alpha}\, R_z(\beta)\, R_y(\gamma)\, R_z(\delta)\,. \tag{40}$$

Theorem 8.2. *Sia U un gate unitario a 1-qubit, e siano l e m due assi non paralleli della sfera di Bloch. Allora esistono dei numeri reali α, β, γ e δ tali che:*

$$U = e^{i\alpha}\, R_l(\beta)\, R_m(\gamma)\, R_l(\delta)\,. \tag{41}$$

Theorem 8.3. *Ogni gate unitario a 1-qubit U può essere espresso come:*

$$U = e^{i\alpha}\, A\, X\, B\, X\, C\,, \tag{42}$$

con α reale e A, B, C operatori unitari che soddisfano:

$$A\, B\, C = \mathbb{1}\,. \tag{43}$$

(Ricordiamo che il gate di Pauli X è il gate NOT).

Tra i gates a 1-qubit possiamo menzionarne alcuni che rivestono una particolare importanza.

Il **gate di Hadamard** definito dalle azioni:

$$\begin{cases} H\,|0\rangle = \dfrac{1}{\sqrt{2}}\,(|0\rangle + |1\rangle) \\ H\,|1\rangle = \dfrac{1}{\sqrt{2}}\,(|0\rangle - |1\rangle) \end{cases} \tag{44}$$

e rappresentato dalla matrice:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{45}$$

Un altro gate utile il cosidetto $\dfrac{\pi}{8}$**-gate** T:

$$\begin{cases} T\,|0\rangle = |0\rangle \\ T\,|1\rangle = |1\rangle\, e^{i\frac{\pi}{4}} \end{cases} \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} = e^{i\frac{\pi}{8}} \begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}. \tag{46}$$

L'ulitima espressione ha determinato il nome usato.

In ultimo possiamo ricordare il **gate di fase** definito dalla matrice:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = T^2\,. \tag{47}$$

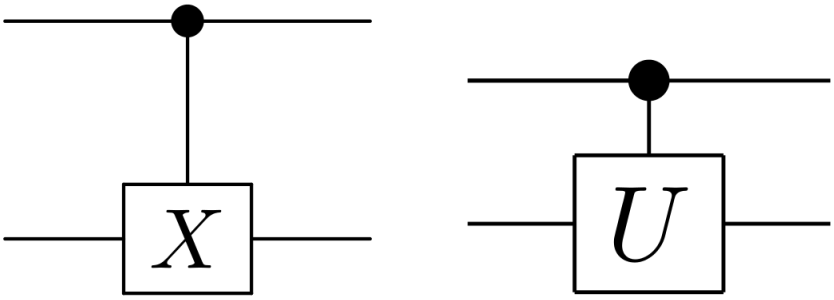


Figura 8:

8.2 Controlled-Gates

Una operazione controllata è del tipo:

"Se A è vero allora esegui B".

Questo tipo di operazione è una delle più utili nella computazione sia classica che quantistica. Il prototipo di operazione controllata è il gate (classico) CNOT visto on precedenza che agisce su una coppia di bits classici come:

$$(x, y) \longmapsto (x,\, x \oplus y) \tag{48}$$

La sua azione può essere interpretata anche in termini di qubits. Se il qubit di controllo $|x\rangle$ è $|1\rangle$ allora il qubit target è flippato, altrimenti è lasciato inalterato. Possiamo scrivere esplicitamente la sua azione come:

$$\begin{aligned} c - NOT\ |0\rangle \otimes |y\rangle &= |0\rangle \otimes |y\rangle \\ c - NOT\ |1\rangle \otimes |y\rangle &= |1\rangle \otimes X\,|y\rangle \end{aligned} \tag{49}$$

con $|y\rangle$ arbitrario stato del secondo qubit. Si usano nche le notazioni $c - X$ (esplicitando l'azione dell'operatore di Pauli X), oppure $C(X)$. Nella base ordinaria dei due qbits $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, l'azione è rappresentata dalla matrice:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{50}$$

(Fig. 8).

Più in generale una **controlled** U trasformazione è definibile come:

$$\begin{aligned} c - U\,|0\rangle \otimes |\psi\rangle &= |0\rangle \otimes |\psi\rangle \\ c - U\,|1\rangle \otimes |\psi\rangle &= |1\rangle \otimes U\,|\psi\rangle \end{aligned} \tag{51}$$

con U trasformazione unitaria. Possiamo dare una rappresentazione matriciale a blocchi:

$$c - U = C(U) = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & U \end{pmatrix} \tag{52}$$

e la definizione posta è valida non solo per uno stato $|\psi\rangle$ ma anche quando $|\psi\rangle$ è uno stato di molti qubits.

Nella logica dei bit classici possiamo scrivere $c - U$:

$$(x, y) \longmapsto (x, U^x\, y)\,, \tag{53}$$

in quanto i valori di x sono 0 e 1 (o comunque interi), ma la notazione é priva di senso con i qubits, che possono trovarsi in una sovrapposizione di stati $|0\rangle$ e $|1\rangle$, come dopo l'applicazione di un gate di Hadamard. L'espressione ha senso solo nella base computazionale $\{|0\,0\,\ldots,0\rangle,\ \ldots\ ,\ |1\,1\,\ldots\,1\rangle\}$ in cui possiamo dare un significato a U^x. In generale possiamo scrivere:

$$c - U = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes U\,, \tag{54}$$

utilizzando i proiettori $|0\rangle\langle 0|$, $|1\rangle\langle 1|$ sugli stati di base del qubit di controllo e il prodotto tensoriale di operatori agenti sui differenti qubits. Con tale espressione risulta più evidente l'unitarietà e quindi la reversibilità del gate controllato.

8.3 No-cloning theorem

Notiamo che con un gate c-NOT classico possiamo simulare un gate FANOUT settando in input il bit target a 0. Consderando l'espressione (48) abbiamo:

$$(x, 0) \longmapsto (x, 0 \oplus x) = (x, x)\,. \tag{55}$$

Questo sembra suggerire la possibilità di poter duplicare lo stato di un qubit $|\psi\rangle$ applicando alla coppia di qubit $|\psi\rangle\otimes|0\rangle$ il gate c-NOT quantistico $C(X)$ ottenendo $|\psi\rangle \otimes |\psi\rangle$. Se:

$$|\psi\rangle = a\,|0\rangle + b|1\rangle\,, \tag{56}$$

abbiamo:

$$|\psi\rangle \otimes |0\rangle = a\,|00\rangle + b|10\rangle\,, \tag{57}$$

e il c-NOT agisce come:

$$C(X)\,|\psi\rangle \otimes |0\rangle = a\,|00\rangle + b|11\rangle\,, \tag{58}$$

(non è un prodotto tensoriale) mentre:

$$\begin{aligned} |\psi\rangle \otimes |\psi\rangle &= (a\,|0\rangle + b|1\rangle) \otimes (a\,|0\rangle + b|1\rangle) \\ &= a^2\,|00\rangle + ab\,|01\rangle + ba\,|10\rangle + b^2\,|11\rangle \end{aligned} \tag{59}$$

e la duplicazione è possibile solo se:

$$a^2 = a\,, \qquad b^2 = b\,, \qquad ab = 0\,, \tag{60}$$

cio'e se e solo se $|\psi\rangle = |0\rangle$, oppure $|\psi\rangle = |1\rangle$. Possiamo duplicare solo gli stati della base computazionale.

In generale possiamo dare la seguente formulazione sulla duplicazione di uno stato quantistico:

Theorem 8.4 (No-cloning)**.** *Dati due sistemi quantistici A e B descritti da un medesimo spazio di Hilbert $\mathcal{H}$,* ***non esiste*** *alcuna trasformazione unitaria U definita su $\mathcal{H} \otimes \mathcal{H}$ che, a meno di un fattore di fase globale, trasformi $|\psi\rangle \otimes |\phi\rangle$ in $|\psi\rangle \otimes |\psi\rangle$* ***per ogni*** *$|\psi\rangle$, $|\phi\rangle$ (normalizzati) $\in \mathcal{H}$, cioè che agisca nella maniera:*

$$U\,|\psi\rangle \otimes |\phi\rangle = e^{i\alpha}\,|\psi\rangle \otimes |\psi\rangle\,, \tag{61}$$

con α dipendente da $|\psi\rangle$ e $|\phi\rangle$, ottenendo due copie esatte di un generico stato quantistico.

Dimostrazione. Infatti se assumiamo che un tale operatore esista possiamo considerare due stati diversi (e arbitrari) $|\psi\rangle$ e $|\psi'\rangle$ (normalizzati) e applicare U ai prodotti $|\psi\rangle \otimes |\phi\rangle$, $|\psi'\rangle \otimes |\phi\rangle$:

$$\begin{aligned} U\,|\psi\rangle \otimes |\phi\rangle &= e^{i\alpha}\,|\psi\rangle \otimes |\psi\rangle \\ U\,|\psi'\rangle \otimes |\phi\rangle &= e^{i\alpha'}\,|\psi'\rangle \otimes |\psi'\rangle\,. \end{aligned} \tag{62}$$

Ma una trasformazione unitaria preseva i prodotti scalari per cui si deve avere:

$$\begin{aligned} \langle\psi'|\psi\rangle &= e^{i(\alpha-\alpha')}\,\langle\psi'|\psi\rangle^2 \\ |\langle\psi'|\psi\rangle| &= |\langle\psi'|\psi\rangle|^2 \end{aligned} \tag{63}$$

cioè $|\langle\psi'|\psi\rangle| = 1$ cioè $|\psi'\rangle$ e $|\psi\rangle$ coincidono a meno di un fattore di fase, oppure $\langle\psi'|\psi\rangle = 0$ e cioè $|\psi'\rangle$ e $|\psi\rangle$ sono ortogonali. Questo contraddice la generalità della trasformazione. □

8.4 Insiemi universali di gates quantistici

I gates quantitici che abbiamo descritto in precedenza agiscono su un qubit (essenzialmente rotazioni sulla sfera di Bloch) o su due qubits (il gate c-NOT o $C(X)$) assumendo che sia possibile implementarle con un device fisico reale. Un algoritmo quantistico richiede, in generale, qualche complicato operatore unitario agente in maniera non banale su n qubits. Nella computazione classsica possiamo implementare complicate operazioni mediante sequenze di operazioni semplici. In pratica siamo in grado di selezionare queste operazioni semplici da qualche insieme finito di gates elementari. In una computazione quantistica vorremmo poter fare lo stesso.

Trovare dei convenienti insiemi universali di gates ha una grande importanza, sia pratica che teorica. Poichè un insieme universale di porte deve essere in grado di implementare un gate non banale come il c-NOT, esso deve contenere almeno un gate non banale a due o più qbits.

Possiamo premettere ai risultati teorici successivi la seguente definizione:

Definition 8.1. *Un gate a* 2*-qubit è detto* ***entangling*** *se per qualche input prodotto (tensoriale)* $|\psi\rangle \otimes |\phi\rangle$ *l'output non è uno stato prodotto tensoriale (cioè i qubit in uscita sono entangled).*

Notiamo che il gate c-NOT stesso è entangling. Infatti consideriamo i seguenti stati:

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right), \\ |\phi\rangle &= |0\rangle, \\ C(X)\, |\psi\rangle \otimes |\phi\rangle &= \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \end{aligned} \tag{64}$$

Il seguente risultato di universalità costituisce un utile punto di partenza.

Theorem 8.5. *Un insieme compsto da un qualsiasi gate a 2-qubit entangling e da tutti i gates a* 1*-qubit, è universale.*

Questo teorema implica, ad esempio, che il gate c-NOT assieme a tutti i gates a 1-qubit, è un insieme universale. Ricordiamo che nella logica di una computazione classica reversibile è necessario avere almeno un gate reversibile non banale a 3-bits, il gate di Toffoli.

Chiaramente l'insieme di tutte le trasformazioni unitarie ha la cardinalità del continuo, per cui sembra impossibile poter costruire tutti i gates a partire da un insieme finito, ma ci accontentiamo di poter implementare un gate unitario entro un qualche specificato livello di precisione. Bisogna chiarire

cosa si intende per *qualità* di una approssimazione. Supponiamo di approssimare una trasformazione desiderata U con un'altra trasformazione unitaria V. L'**errore** nell'approssimazione è definito come:

$$E(U,V) \equiv \max_{|\psi\rangle, \|\psi\|=1} \|(U-V)\,|\psi\rangle\|\,. \tag{65}$$

Diremo allora che un operatore U può essere *aprosssimato con accuratezza arbitraria*, se data una qualsiasi tolleranza $\varepsilon > 0$, siamo in grado di costruire un operatore V tale che $E(U,V) < \varepsilon$.

Notiamo che vale la seguente disuguaglianza:

$$E(U_n \cdots U_1,\, V_n \cdots V_1) \leqslant E(V_n, U_n) + \ \ldots\ + E(U_1, V_1)\,. \tag{66}$$

Possiamo modificare la definizione di insieme universale di gates nel modo seguente:

Definition 8.2. *Un insieme (finito) di gates è universale se per ogni $n \geqslant 1$, ogni n-qubit operatore unitario può essere approssimato con precisione arbitraria da un circuito quantistico usando solo gates di tale insieme.*

Un set di gates a 1-qubit è detto universale se ogni gate a 1-qubit può essere approssimato con gates di tale insieme. Abbiamo visto che tutte le trasformazioni unitarie si possono esprimere (a meno di un fattore di fase) tramite rotazioni, ma il seguente teorema dice che bastano due rotazioni per approssimare tutte le rotazioni (e quindi, considerando banale applicare il fattore di fase, tutte le trasformazioni unitarie).

Theorem 8.6. *Se un insieme di due gates a 1-qubit (rotazioni) $\mathcal{G} = \{R_l(\beta),\, R_m(\gamma)\}$ soddisfa le condizioni:*

(i) l e m sono assi di rotazione non paralleli della sfera di Bloch, e

(ii) β e γ sono numeri reali tali che β/π e γ/π non sono numeri razionali,

allora $\mathcal{G}$ è un insieme universale per gates a 1-qubit.

Il sequente teorema esplicita meglio una scelta minima di rotazioni:

Theorem 8.7. *L'insieme $\mathcal{G} = \{\, H\,T\,H\,T\,,\ T\,H\,T\,H\,\}$ soddisfa le condizioni del precedente teorema 8.6 (H e T sono il gate di Hadamard e il gate-$\pi/8$).*

permettendoci di enunciare il seguente risultato.

Theorem 8.8. *L'insieme $\{H\,,\,T\}$ è un insieme di porte universali per i gates a 1-qubit, e l'insieme $\{$ CNOT, $H\,,\,T\}$ è un insieme universale di gates quantistici.*

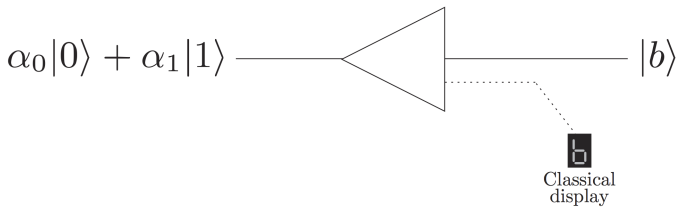


Figura 9: Misura di un qubit.

9 Misure quantistiche

Cerchiamo di chiarire che cosa intendiamo con un processo di misura che può fornire un output classico di un device quantistico. Ricordiamo che un processo di misura non è una trasformazione unitaria dello stato quantistico e questo in realtà viene proiettato (collasso) in un autostato corrispondente alla misura ottenuta. Se misuriamo un osservabile M, possiamo considerare la sua decomposizione spettrale:

$$M = \sum_j m_j \, P_j \,, \tag{67}$$

con P_j proiettore ortogonale sull'autospazio di M con autovalore m_j e uno stato puro $|\psi\rangle$ collassa in un sottospazio:

$$|\psi\rangle \longmapsto \frac{P_j \, |\psi\rangle}{\|P_j \, |\psi\rangle\|} \,, \tag{68}$$

con probabilità:

$$p_j = \|P_j \, |\psi\rangle\|^2 = \langle\psi|P_j|\psi\rangle \,, \tag{69}$$

e segnando su un display il risultato m_j (non sappiamo a priori il risultato della misura, possiamo solo predire le probabilità di uscita se conosciamo lo stato $|\psi\rangle$.

In un device quantistico si assume di avere un apparato che possa eseguire una misura dei singoli qubits (tutti in una volta o solamente alcuni se eseguiamo una misura parziale). Ogni singolo qubit dopo la misura rimane in uno dei due stati di base $|0\rangle$ o $|1\rangle$ (vedi Fig. 9). In output per ogni qubit abbiamo due "canali", uno quantistico (lo stato collassato del qubit), ed uno "classico" (un bit corrispondente allo stato del qubit dopo la misura). In genere dopo la misura uno dei due canali viene trascurato e non viene indicato.

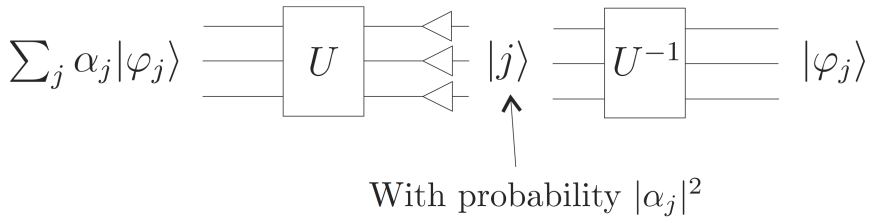


Figura 10: Misura di Von Neumann completa.

Misurando lo stato di tutti i qubit eseguiamo quella che è nota come **misura di Von Neumann completa** rispetto ad una base. Data una base ortonormale completa $\{|\phi_j\rangle\}$, uno stato $|\psi\rangle$ è sviluppato nella base come:

$$|\psi\rangle = \sum_j \alpha_j \, |\phi_j\rangle \, , \tag{70}$$

dopo la misura abbiamo il risultato "j" con probabilità $|\alpha_j|^2$ e lo stato è $|\phi_j\rangle$. Un device quantistico sostanzialmente assume sempre possibile una misura nella base computazionale $\{|\phi_j\rangle\} = \{|0\rangle, |1\rangle\}^{\otimes n}$. Possiamo generalizzare il postulato di misura considerando uno stato $|\psi\rangle$ in uno spazio di Hilbert composto da due sottosistemi A e B, $\mathcal{H}_A \otimes \mathcal{H}_B$:

$$|\psi\rangle = \sum_j \alpha_j \, |\phi_j\rangle \otimes |\gamma_j\rangle \, , \tag{71}$$

con $\{|\phi_j\rangle\}$ base ortonormale (completa) d $\mathcal{H}_A$, e $|\gamma_j\rangle \in \mathcal{H}_B$ normalizzati ma non necessariamente ortogonali e distinti. Allora una misura eseguita sul solo sistema A darà come risultato un valore j con probabilità $|\alpha_j|^2$ e lo stato del sistema bipartito dopo la misura sarà $|\phi_j\rangle \otimes |\gamma_j\rangle$. (Questa ultima variante è utile quando operiamo una misura solo su una parte di qubits).

Assunto che un device possa eseguire una misura dei singoli qubits nella base computazionale (normalmente indicata con $\{|j\rangle\}$ essendo in diretta corrispondenza con i numeri j in forma binaria). possiamo usare un circuito quantistico per eseguire una misura di Von Neumann rispetto a qualsiasi altra base $\{|\phi_j\rangle\}$. Prima costruiamo un circuito quantistico che implementi la trasformazione unitaria che rappresenta il cambiamento di base:

$$U \, |\phi_j\rangle = |j\rangle \, . \tag{72}$$

Operando tale circuito sullo stato:

$$U \;\; : \;\; |\psi\rangle = \sum_j \alpha_j \, |\phi_j\rangle \;\longmapsto\; \sum_j \alpha_j \, |j\rangle \, , \tag{73}$$

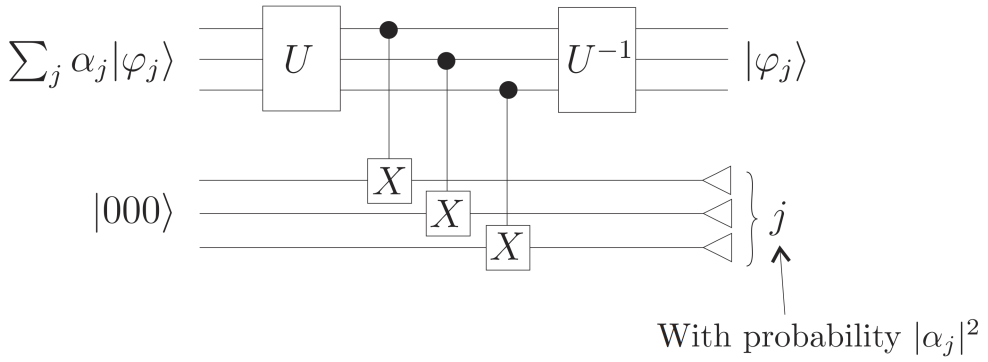


Figura 11: Misura di Von Neumann posposta alla fine.

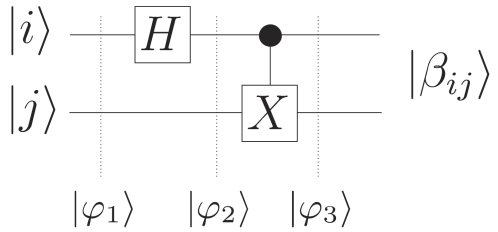


Figura 12: Generazione degli stati di Bell.

ed eseguendo una misura lo proiettiamo sullo stato $|j\rangle$. Operando poi il gate inverso U^{-1} eseguiamo un ritorno alla base voluta con lo stato dei registri nell'elemento di base $|\phi_j\rangle$. Vedi Fig. 10.

Un approccio alternativo è illustrato in Fig. 11. In questo caso non misuriamo direttamente lo stato dopo il cambio di base, ma "duplichiamo" i singoli stati della base computazionale su un registro ancillare di qubits, che poi misuriamo sulla base computazionale.

Come esempio di implementazione di un cambio di base U, consideriamo la base di stati di Bell di 2-qubits:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \qquad |\beta_{01}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \\ |\beta_{10}\rangle &= \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \qquad |\beta_{11}\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle . \end{aligned} \tag{74}$$

Un circuito che esegue la trasformazione dalla base computazionele alla base di Bell è illustrato in Fig. 12. Si può verificare facilmente che se in input

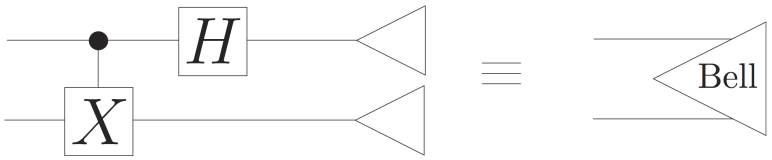


Figura 13: Misura nella base di Bell.

abbiamo lo stato $|\varphi_1\rangle = |00\rangle$ allora, dopo il gate di Hadamard abbiamo lo stato:

$$|\varphi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |10\rangle\right), \tag{75}$$

e dopo il c-NOT:

$$|\varphi_3\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) = |\beta_{00}\rangle\,. \tag{76}$$

Analogamente possiamo verificare la trasformazione per i rimanenti stati computazionali $|01\rangle$, $|10\rangle$ e $|11\rangle$.

Per implementare una **misura di Bell** possiamo implementare il circuito di Fig. 13 (inverso del precedente). Se vogliamo solo il risultato delle misure classiche (00, 01, 10, 11), non dobbiamo inplementare la trasformazione inversa. Si adotta un simblo grafico distinto per tale circuito, essendo una operazione frequente nella computazione quantistica.

10 Protocolli quantistici

Siamo ora in grado di considerare i primi protocolli per l'informazione quantistica, in particolare dei protocolli di comunicazione implementabili con i mezzi descritti in precedenza. Essi descrivono due parti che vogliono eseguire una comunicazione tra loro scambiandosi dei dati. Nella descrizione di tali protocolli è comune adottare la convenzione di chiamare le due parti "Alice" e "Bob", per cui seguiremo tale tradizione. Faremo riferimento a *canali* di comunicazione. Un **canale quantistico** si riferisce ad una linea di comunicazione che contiene dei qubits tra due posizioni remote (ad esempio una fibra ottica). Un **canale classico** contiene invece bits classici (non qubits). In ogni caso i protocolli richiedono che Alice e Bob condividano inizialmente almeno una coppia entangled di qubits in uno stato di Bell del tipo:

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{77}$$

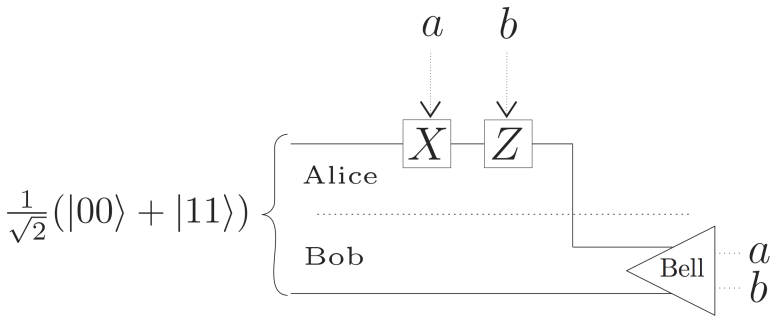


Figura 14: Superdense coding.

A volte tale stato viene detto una *coppia EPR*. Tale coppia deve essere stata creata prima della comunicazione quando i qubits sono assieme in un laboratorio e possono essere fatti interagire e preparare in uno stato entangled. Dopo di chè essi sono separati (senza interagire con l'ambiente per mantenere l'entanglement) e un qubit viene dato ad Alice e l'altro a Bob. L'entanglement tra i due qubits fornisce la risorsa da utilizzare per realizzare il protocollo di comunicazione. Assumeremo che Alice possieda il primo qubit e Bob il secondo della coppia (77).

10.1 Superdense coding

Supponiamo che Alice voglia inviare a Bob due bits classici di informazione. Si può ottenere ciò tramite un unico canale quantistico sfruttando la coppia (77). A seconda della coppia di bits che vuole comunicare Alice agisce sul suo qubit con uno tra quattro gates possibili, operando una trasformazione sul suo qubit. Di conseguenza cambia lo stato della coppia dei due qubits:

$$\begin{aligned}
&00 \quad \mathbb{1}\otimes\mathbb{1} \quad : \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \longmapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\beta_{00}\rangle \\
&01 \quad X\otimes\mathbb{1} \quad : \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \longmapsto \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\beta_{01}\rangle \\
&10 \quad Z\otimes\mathbb{1} \quad : \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \longmapsto \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\beta_{10}\rangle \\
&11 \quad ZX\otimes\mathbb{1} \quad : \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \longmapsto \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = |\beta_{11}\rangle
\end{aligned} \tag{78}$$

Dopo aver applicato una delle quattro trasformazioni Alice spedisce (lungo un canale quantistico) il qubit a Bob. Bob si trova in possesso di una coppia

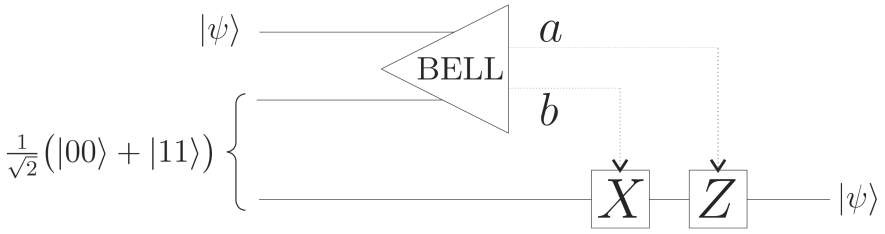


Figura 15: Teletrasporto quantistico.

di qubits in uno stato coincidente con uno specifico stato della base di Bell $\{|\beta_{00}\rangle, |\beta_{01}\rangle, |\beta_{10}\rangle, |\beta_{11}\rangle\}$ e tramite una misura in tale base determina con certezza la coppia di bits che Alice voleva comunicare (vedi Fig. 14).

Questo semplice esempio serve per dimostrare la possibilità di codificare una informazione classica (espressa tramite un certo numero di bits classici) in un canale quantistico contenente un numero inferiore di qubits. Il protocolo prende il nome di **superdence coding**.

10.2 Teletrasporto quantistico

Supponiamo ora che Alice voglia comunicare a Bob lo stato $|\psi\rangle$ di un qubit in suo possesso avendo a sua disposizione un solo canale classico di comunicazione (condivide però una coppia di Bell assieme a Bob). Sembrerebbe che per comunicare lo stato Alice abbia solo due maniere, o inviare fisicamente il qubit a Bob (ma dovrebbe avere un canale quantistico di comunicazione), oppure comunicare i coefficienti complessi con *precisione infinita*. In realtá, sfruttando la coppia di Bell posseduta assieme a Bob, può sfruttare l'entanglement per inviare l'informazione completa mediane un canale classico con soli due bits.

Supponiamo allora che Alice abbia nel suo laboratorio uno stato quantistico

$$|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle \tag{79}$$

assieme al primo qubit della coppia di Bell:

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \tag{80}$$

Lo stato a 3-qubits posseduti assieme da Alice e Bob è inizialmente nello stato $|\psi\rangle| \otimes |\beta_{00}\rangle$ che possiamo espandere come:

$$\begin{aligned} |\psi\rangle\,|\beta_{00}\rangle &= \frac{1}{\sqrt{2}}\left(a_0\,|000\rangle + a_1\,|100\rangle + a_0\,|011\rangle + a_1\,|111\rangle\right) \\ &= \frac{1}{2}\,|\beta_{00}\rangle\,|\psi\rangle + \frac{1}{2}\,|\beta_{01}\rangle\,(X\,|\psi\rangle) + \frac{1}{2}\,|\beta_{10}\rangle\,(Z\,|\psi\rangle) + \frac{1}{2}\,|\beta_{11}\rangle\,(X\,Z\,|\psi\rangle) \end{aligned} \tag{81}$$

sviluppato nella base di Bell dei primi due qubits. Alice può quindi eseguire una misura di Von Neumann nella base di Bell ottenendo uno qualsiasi di quattro risultati (esprimibili tramite due bit classici) con uguale probabilità 1/4. Dopo la misura lo stato dei 3 qubits è collassato in uno degli stati dell'espansione precedente. Alice comunica quindi a Bob (tramite un canale classico a 2-bits) il risultato della misura e Bob può applicare il corrispondente operatore inverso al qubit in suo possesso:

$$\begin{array}{cc} 0,0 & \mathbb{1} \\ 0,1 & X \\ 1,0 & Z \\ 1,1 & Z\,X \end{array} \tag{82}$$

ottenendo il suo qubit (inizialmete entangolato) esattamente nello stato $|\psi\rangle$ (vedi Fig. 15).

Possiamo dire che Alice ha trasmesso un completo stato quantistico a Bob utilizzando solo 2 bits classici di informazione. Notiamo che dopo la misura i due qubits in suo possesso non contengono più alcuna informazione sullo stato $|\psi\rangle$, per cui parliamo di **teletrasporto** nel senso che lo stato viene trasferito dalla sorgente al ricevitore (sembra all'istante, ma occorre il tempo necessario alla comunicazione classica per poter "ricostruire" lo stato vero e prorio da parte di Bob). Il teletrasporto promette di essere un potente mezzo per muovere l'informazione quantistica tra locazioni che possono essere anche molto distanti. Non siamo ancora arrivati al teletrasporto fantascientifico come in Star-Trek!

11 Algoritmo di Deutsch

Vediamo ora il primo algoritmo quantistico, ideato da Deutsch. Esso è un buon punto di partenza perchè, pur essendo molto semplice e facile da capire, illustra le idee di base di **parallelismo quantistico** e **interferenza quantistica** che sono usate in tutti gli algoritmi quntistici più utili.

Il problema risolto dall'algoritmo di Deutsch può essere espresso nei seguenti termini. Supponiamo di aver un circuito per calcolare una funzione incognita a 1-bit (non necessariamente reversibile):

$$f \ : \ \{0,\, 1\} \ \longrightarrow \ \{0,\, 1\}. \tag{83}$$

Trattiamo questo circuito come una "black box" o un "oracolo". Questo significa che possiamo applicare il circuito per per ottenere il valore $f(x)$ per un dato input x, ma non possiamo ottenere nessuna informazione sul funzionamento interno del circuito per conoscere la funzione f. Il problema è solo quello di determinare il valore di $f(0) \oplus f(1)$. Se $f(0) \oplus f(1) = 0$, allora sappiamo che $f(0) = f(1)$ (anche senza sapere i valori singoli), e possiamo dire che f è "costante". Se d'altra parte troviamo che $f(0) \oplus f(1) = 1$, allora sappiamo che $f(0) \neq f(1)$, e diciamo che la funzione è "bilanciata". Pertanto valutare $f(0) \oplus f(1)$ è equivalente a determinare se la funzione f è costante o bilanciata.

Problema di Deutsch

Input: Una scatola nera per calcolare una funzione incognita $f \ : \ \{0,\, 1\} \ \longrightarrow \ \{0,\, 1\}$.
Problema: Determinare il valore di $f(0) \oplus f(1)$ eseguendo valutazioni di f.

Quante valutazioni dell'oracolo per f devono essere eseguite classicamente per determinare $f(0) \oplus f(1)$?. Chiaramente la risposta è 2. L'algoritmo di Deuttsch quantistico permette di determinare il valore di $f(0) \oplus f(1)$ eseguendo una *singola* valutazione di una versione quantistica del'oracolo per f.

Abbiamo visto che indipendentemente dalla invertibilità di f, possiamo costruire un circuito classico reversibile che effettui il calcolo di f e assumiamo che questo sia trasformabile in un circuito quantistico rimpiazzando ogni gate classico reversibile con l'analogo gate unitario quantistico. Questo circuito quantistico può essere sintetizzato come un operatore unitario agente nella base computazionale dei due qubits nella maniera classica:

$$U_f \ : \ |x\rangle|y\rangle \ \longmapsto \ |y\rangle|y \oplus f(x)\rangle\,, \qquad ,x, y \in \{0, 1\}\,. \tag{84}$$

Il primo qubit formalmente rimane inalterato, allora possiamo pensare U_f come un 1-qubit gate $\widehat{U}_{f(x)}$ (che mappa $|b\rangle \ \mapsto \ |b \oplus f(x)\rangle$, controllato dallo stato $|x\rangle$ del primo qubit come mostrato in Fig. 16. In termini operatoriali:

$$U_f = |0\rangle\langle 0| \otimes \widehat{U}_{f(0)} + |1\rangle\langle 1| \otimes \widehat{U}_{f(1)}\,. \tag{85}$$

Figura 16: Il gate a 2-qubit U_f può essere pensato come un gate a 1-qubit $\widehat{U}_{fx)}$ agente sul secondo qubit, controllato dal primo qubit.

Possiamo ora vedere la sua azione con stati in sovrapposizione. Fissiamo il bit controller $|x\rangle$ nella base computazionale e il qubit target nello stato $|0\rangle - |1\rangle$:

$$\begin{aligned} U_f \ : \ |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) &\longmapsto \left(\frac{U_f|x\rangle\|0\rangle - U_f|x\rangle|1\rangle}{\sqrt{2}} \right) \\ &= |x\rangle \left(\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} \right) . \end{aligned} \tag{86}$$

Cosideriamo ora in dettaglio il termine entro parentesi:

$$\begin{aligned} f(x) = 0 \ : \quad & \frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\ f(x) = 1 \ : \quad & \frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned} \tag{87}$$

Cioè:

$$\frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = (-1)^{f(x)} \frac{|0\rangle - |1\rangle}{\sqrt{2}} , \tag{88}$$

$$U_f \ : \ |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \longmapsto (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) . \tag{89}$$

Abbiamo trovato un autovettore di U_f con un autovalore $(-1)^{f(x)}$ che possiamo associare (**kick back**) al primo qubit. Se questo viene posto in uno stato sovrapposizione:

$$\begin{aligned} U_f \ : \ (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) &\longmapsto \\ &\left((-1)^{f(0)} \alpha_0 |1\rangle + (-1)^{f(1)} \alpha_1 |1\rangle \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) . \end{aligned} \tag{90}$$

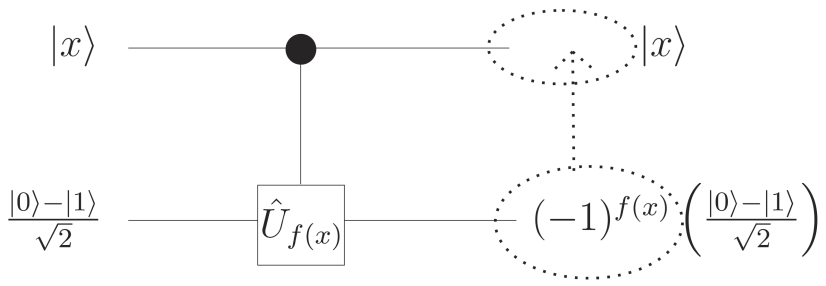


Figura 17: L'autovalore $(-1)^{f(x)}$ è riassegnato al primo qubit.

Il gate U_f è tale che se il secondo qubit è nello stato $|y\rangle = |0\rangle$, allora in output abbiamo $|x\rangle|f(x)\rangle$ (è il funzionamento classico). Possiamo vedere cosa succede se il primo qubit è in una sovrapposizione quantistica:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \tag{91}$$

mantenendo il secondo qubit di input nello stato $|y\rangle = |0\rangle$. Allora la coppia di qubits si trova in input nello stato:

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle, \tag{92}$$

e dopo l'applicazione di U_f:

$$\frac{1}{\sqrt{2}}|0, f(0)\rangle + \frac{1}{\sqrt{2}}|1, f(1)\rangle. \tag{93}$$

In un certo senso, U_f ha valutato *simultaneamente* f con i due possibili inputs 0 e 1 posti in sovrapposizione. Però se ora misuriamo l'output nella base computazionale abbiamo due possibili risultati: $|0, f(0)\rangle$ oppure $|1, f(1)\rangle$ (con probabilità entrambe 1/2). Otteniamo cioè uno solo dei due valori di $f(0)$ e $f(1)$ è disponibile nella base computazionale. Fortunatamente questo non è la fine della storia.

Ricordiamo che nel problema di Deutsch non siamo in realtà interessati ai valori individuali di $f(x)$ ma solo al valore di $f(0) \oplus f(1)$. Vediamo ora come sfruttare l'*interferenza quantistica* per avere questa informazione globale sulla funzione f, ed in maniera più efficiente rispetto al calcolo classico. Possiamo combinare le analisi precedenti implementando un circuito per l'algoritmo di Deutsch (Fig. 18).

Il primo qubit, inizializzato a $|0\rangle$, viene trasformato nella sovrapposizione $|0\rangle + |1\rangle$ dal gate di Hadamard. Il secondo qubit è assunto invece nella

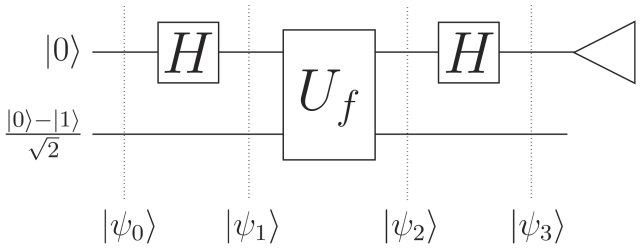


Figura 18: Circuito per l'algoritmo di Deutsch.

sovrapposizione $|0\rangle - |1\rangle$ (nella figura non mostriamo i gates X e H per ottenerlo dallo stato $|0\rangle$).

Seguendo lo stato nella sua evoluzione:

$$\begin{aligned}
|\psi_0\rangle &= |0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \longmapsto \\
|\psi_1\rangle &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\
&= \frac{1}{\sqrt{2}}|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + \frac{1}{\sqrt{2}}|1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \longmapsto \\
|\psi_2\rangle &= \frac{(-1)^{f(0)}}{\sqrt{2}}|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + \frac{(-1)^{f(1)}}{\sqrt{2}}|1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\
&= (-1)^{f(0)} \left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}\,|1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).
\end{aligned} \tag{94}$$

L'azione dell'ultimo gate di Hadamard permette di risolvere il problema di Deutsch. Se f è una funzione costante ($f(0) \oplus f(1) = 0$), allora

$$\begin{aligned}
|\psi_2\rangle &= (-1)^{f(0)} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \longmapsto \\
|\psi_3\rangle &= (-1)^{f(0)}\,|0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right),
\end{aligned} \tag{95}$$

e una successiva misurazione del primo qubit fornisce con certezza (probabilità 1) il risultato $0 = f(0) \oplus f(1)$.

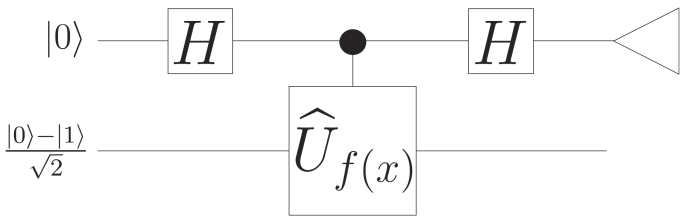


Figura 19: Circuito equivalente per l'algoritmo di Deutsch.

Se f è invece una funzione bilanciata ($f(0) \oplus f(1) = 1$) abbiamo:

$$\begin{aligned} |\psi_2\rangle &= (-1)^{f(0)} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \longmapsto \\ |\psi_3\rangle &= (-1)^{f(0)} |1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right), \end{aligned} \tag{96}$$

e una successiva misurazione del primo qubit fornisce con certezza (probabilità 1) il risultato $1 = f(0) \oplus f(1)$. Pertanto alla fine del circuito determiniamo con certezza se la funzione è costante o bilanciata.

Per capire meglio come verrà generalizzato l'algoritmo di Deutsch è utile ricordare che l'azione dell'operatore $U_f : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$ può essere visto come l'aione di un operatore a 1-qubit $\widehat{U}_{f(x)}$, la cui azione è controllata dallo stato del qubit di controllo (vedi Fig. 19). Lo stato $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ è un autostato di $\widehat{U}_{f(x)}$ con autovalore $(-1)^{f(x)}$. Codificando tali autovalori nei fattori di fase del qubit di controllo, siamo stati in grado di determinare $f(0) \oplus f(1)$ individuando il fattore di fase relativo tra $|0\rangle$ e $|1\rangle$. La distinzione tra $|0\rangle + |1\rangle$ e $|0\rangle - |1\rangle$ è fatta tramite il gate di Hadamard.

12 Algoritmo di Deutsch-Jozsa

L'algoritmo di Deutsch-Jozsa risolve un problema che è una diretta generalizzazione di quello di Deutsch visto in precedenza ed ha la stessa struttura. Come prima, abbiamo un circuito reversibile che implementa una funzione incognita f, ma questa volta è una funzione con un input a n bits:

$$f : \{0,1\}^n \longrightarrow \{0,1\}. \tag{97}$$

Inoltre sappiamo per certo (una assuzione di ipotesi) che f è o *costante* (intendendo che $f(x)$ ha lo stesso valore per ogni x), oppure f è bilanciata

(intendendo che $f(x) = 0$ per esattamente metà dei possibili input x, e $f(x) = 1$ per l'altra metà degli input). Il problema è quello di determinare se f è costante o bilanciata, eseguendo delle opportune valutazioni di f.

Problema di Deutsch-Josza

Input: Una scatola nera per calcolare una funzione incognita $f : \{0, 1\}^n \longrightarrow \{0, 1\}$.
Assunzione: f è costante oppure una funzione bilanciata.
Problema: Determinare se f è costante o bilanciata eseguendo valutazioni di f.

Consideriamo come risolvere questo problema con un algoritmo classico. Supponiamo di avere usato l'oracolo per determinare $f(x)$ per esattamente la metà dei possibili inputs x (eseguendo 2^{n-1} valutazioni di f), e tutte le valutazioni hanno dato $f(x) = 0$. A questo punto, sopettiamo fortemente che f sia costante. Comunque, è possibile che le valutazioni dei rimanenti 2^{n-1} inputs forniscano $f(x) = 1$ ogni volta e f è in tal caso bilanciata. Pertando, ipotizzando il caso peggiore, con un algoritmo classico sono necessarie $2^{n-1} + 1$ valutazioni. La proprietà di essere costante o bilanciata è una proprietà globale che riguarda l'insieme dei possibili risultati. Come nell'algoritmo di Deutsch, un algoritmo quantistico può avvantaggiarsi della sovrapposizione e interferenza quantistica per determinare questa proprietà globale fi f. L'algoritmo di Deutsch-Josza determinerà se f è costante o bilanciata con una sola valutazione da parte di una versione quantistica del circuito reversibile per f.

In analogia con quello fatto in precedenza, possiamo definire l'azione del circuito quantistico che valuta f come:

$$U_f \ : \ |x\rangle|y\rangle \ \longmapsto \ |x\rangle|y \oplus f(x)\rangle\,, \qquad x \in \{0,1\}^n, y \in \{0,1\}\,. \tag{98}$$

Questa definizione è sostanzialmente l'azione classica trasferita nella base computazionale, x è la stringa di n bits classici che individua l'elemento $|x\rangle$ di base per i primi n qubits. Come in precedenza possiamo pensare U_f come l'azione di operatore a 1-qubit $\widehat{U}_{f(x)}$, la cui azione sull'ultimo qubit è controllata dal registro di qubits nello stato $|x\rangle$:

$$\begin{aligned} U_f &= \sum_{x\in\{0,1\}^n} |x\rangle\langle x| \otimes \widehat{U}_{f(x)}\,, \\ \widehat{U}_{f(x)} \ &: |y\rangle \ \longmapsto \ |y \otimes f(x)\rangle\,, \qquad y \in \{0,1\}\,. \end{aligned} \tag{99}$$

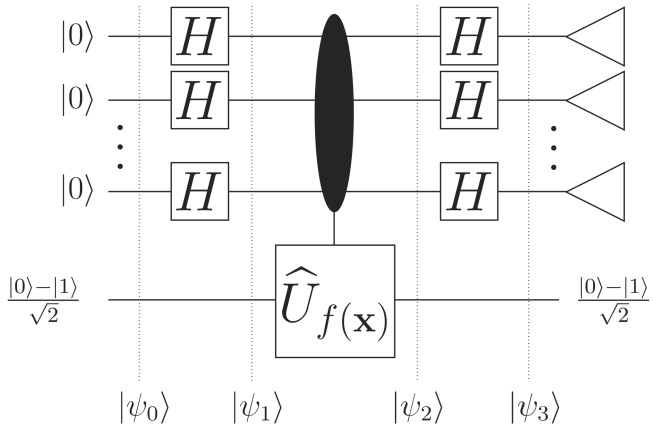


Figura 20: Algoritmo di Deutsch-Josza.

Possiamo vedere che $|0\rangle - |1\rangle$ è un autostatp di $\widehat{U}_{f(x)}$ con autovalore $(-1)^{f(x)}$.

Il circuito per l'algoritmo di Deutsch-Josza è mostrato in Fig. 20. Notiamo la somiglianza con il caso più semplice. Al posto di un semplice gate di Hadamard a 1-qubit, abbiamo un prodotto tensoriale di n 1-qubit gates di Hadamard, $H^{\otimes n}$ da applicare allo stato $|0\rangle^{\otimes n}$.

Come abbiamo fatto con l'algoritmo di Deutsch, possiamo seguire lo stato attraverso il circuito. Inizialmente abbiamo:

$$|\psi_0\rangle = |0\rangle^{\otimes n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) . \tag{100}$$

Consideriamo l'azione delle trasformazioni di Hadamard:

$$\begin{aligned} H^{\otimes n} |0\rangle^{\otimes n} &= \left(\frac{1}{\sqrt{2}}\right)^n (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle . \end{aligned} \tag{101}$$

Otteniamo una sovrapposizione di tutti gli n elementi di base con la stessa ampiezza. Pertanto dopo l'applicazione dei primi gates di Hadamard lo stato è:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) . \tag{102}$$

L'azione del gate U_f risulta quindi:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) . \tag{103}$$

Vediamo ora l'azione dei gate di Hadamard successivi. Su un singolo qubit in uno stato di base $|x\rangle$ abbiamo:

$$H|x\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^x|1\rangle\right) = \frac{1}{\sqrt{2}}\sum_{z\in\{0,1\}}(-1)^{xz}|z\rangle\,, \qquad x \in \{0,1\}\,. \tag{104}$$

Pertanto l'azione della trasformazione multipla di Hadamard su un singolo elemento di base $|x\rangle = |x_1\rangle \cdots |x_n\rangle$:

$$\begin{aligned} H^{\oplus n}|x\rangle &= H|x_1\rangle \otimes \cdots H|x_n\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{z_j\in\{0,1\}}(-1)^{x_1z_1+\cdots+x_nz_n}|z_1\rangle\cdots|z_n\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{x\cdot z}|z\rangle\,, \end{aligned} \tag{105}$$

dove $x\cdot z$ denota un prodotto scalare con le somme eseguite modulo 2. (cioè le somme sono operazioni XOR). Lo stato del sistema dopo l'applicazione di tali gate di Hadamard risulta quindi:

$$\begin{aligned} |\psi_3\rangle &= \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{x\cdot y}|z\rangle\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \\ &= \frac{1}{2^n}\sum_{z\in\{0,1\}^n}\left(\sum_{x\in\{0,1\}^n}(-1)^{f(x)+x\cdot z}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\,. \end{aligned} \tag{106}$$

Abbiamo espanso nella base computazionale il risultato, per cui una misurazione dei primi n qubits fornirà uno di tali stati con il valore x. Per vedere cosa accade concentriamoci sull'ampiezza totale dello stato $|0\rangle^{\otimes n}$ ($z_j = 0$ per ogni j). L'ampiezza risulta:

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{fx)}\,. \tag{107}$$

Se f è costante l'ampiezza risultante è o $+1$ oppure -1 (dipende dal valore comune di $f(x)$). Questo ci dice che con probabilità 1 lo stato dei primi n qubits è $|0\rangle^{\otimes n}$ (tutti gli altri coefficienti devono essere nulli).

Se f è bilanciata allora i termini positivi e negativi si compensano e il coefficiente di cui sopra è sicuramente nullo, per cui una misura non potr'a **mai** dare tutti i qubits nello stato $|0\rangle$. Abbiamo quindi che una sola valutazione di f e la misura dello stato di output nella base computazionale ci

permette di decidere se f è *costante* (lo stato risultante è $|0\rangle^{\otimes n}$) o *bilanciata* (lo stato risultante *non* è $|0\rangle^{\otimes n}$).

Certamente il problema di Deutsch-Josza è puramente accademico (è difficile vederne una utilità pratica), la sua importanza è quello di dimostrare sostanzialmente la potenza del parallelismo e dell'interferenza quantistica come mezzi per affrontare problemi che classicamente richiedono un uso estensivo di risorse.