Quantum Computing

Alessandra Di Pierro

Appunti del corso - A.A. 2001-2002

Lezioni 4-5: Introduzione generale

1 Computazioni classiche reversibili

Una differenza fondamentale tra i circuiti classici e quelli quantistici è che le porte logiche classiche potrebbero essere irreversibili (ad esempio `AND`, `XOR`, `NAND`), mentre le porte logiche quantistiche sono sempre unitarie e quindi reversibili. Per poter operare in modo reversibile è necessario che la funzione da valutare sia invertibile, cioè iniettiva e surgettiva. In tal caso possiamo infatti risalire in maniera univoca da ogni output al valore dell'input che l'aveva generato, cioè operare in modo inverso.

Una qualsiasi computazione irreversibile può essere trasformata in una equivalente computazione reversibile rendendo biunivoca la corrispondente funzione da valutare. Ad esempio, data una qualsiasi funzione:

$$f : \{0,1\}^n \mapsto \{0,1\}^m,$$

possiamo costruire $\tilde{f} : \{0,1\}^{n+m} \mapsto \{0,1\}^{n+m}$, tale che:

$$\tilde{f}(x, 0^m) = (x, f(x)),$$

dove 0^m denota m bits inizializzati a 0. La funzione $\tilde{f}$ è biunivoca e permette di valutare $f(x)$ agendo sull'input $(x, 0^m)$.

Anche per le computazioni classiche reversibili esiste un'operazione elementare che è universale, cioè in grado di esprimere tutte le altre computazioni reversibili. Questa operazione è detta la *porta di Toffoli* ed il circuito associato è rappresentato in Figura 1. La porta di Toffoli ha tre bits in input: due sono bits di controllo e il terzo è il bit target che viene scambiato se i bits di controllo sono entrambi 1, come mostrato nella Tabella 1.

IN			OUT		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Table 1: Tabella di verità della porta di Toffoli

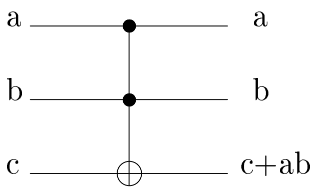

Figure 1: Rappresentazione della porta di Toffoli

Nota che applicando per due volte consecutive la porta di Toffoli si ottiene lo stesso risultato di partenza:

$$(a, b, c) \to (a, b, c \oplus ab) \to (a, b, c),$$

cioè l'operazione è reversibile (in particolare essa coincide con la sua inversa). Applicando l'operazione con $c = 1$, otteniamo $a' = a$, $b' = b$ e $c' = 1 \oplus ab = \neg ab$, cioè la simulazione di `NAND` come operazione reversibile. Anche l'operazione di copia di un bit classico si può rendere reversibile: applicando la porta di Toffoli con $a = 1$ e $c = 0$ si ottiene come risultato la copia del bit b (Ricordiamo che questa operazione di copia non è possibile per un qubit). L'universalità della porta di Toffoli per le computazioni reversibili segue quindi dal risultato di universalità del `NAND` per le computazioni classiche non reversibili.

Una dimostrazione diretta dell'universalità della porta di Toffoli si ottiene osservando che una generica funzione invertibile su n bits si può vedere come una permutazione di stringhe di n bits. Una tale permutazione si può esprimere con una porta di Toffoli con n bits in input, che nel seguito indicheremo con θ^n. Ad esempio, per trasformare la stringa 0100 nella permutazione 0101 possiamo usare il circuito in Figura 2 che fa uso di θ^4 e della porta logica `NOT`.

Analogamente, utilizzando θ^n e `NOT` si può trasporre una qualsiasi stringa di n bits in un'altra che differisce per un solo bit. Per realizzare la trasposizione di

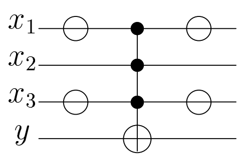

Figure 2: Circuito che permuta la stringa 0100 in 0101

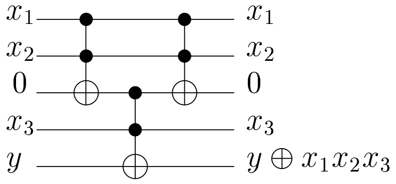

Figure 3: Costruzione di θ^4 per mezzo di tre θ^3

due stringhe che differiscono per $s > 1$ bits, basta comporre le trasposizioni che realizzano lo scambio di ciascuno degli s bits. Il circuito in Figura 2 si può quindi generalizzare per ottenere qualsiasi permutazione di n bits.
L'ultimo passo da fare per dimostrare l'universalità della porta di Toffoli a 3 bits (θ^3) è far vedere come si può costruire θ^n per un generico n usando solo operazioni θ^3. Consideriamo di nuovo il caso $n = 4$. La costruzione richiede l'utilizzo di un bit extra che chiameremo *scratch bit* ed è riportata in Figura 3. Lo scopo dell'ultimo θ^3 è di riportare il valore dello scratch bit al valore originale 0.

Esercizio 1.1 *Modificare il circuito in Figura 3 in modo da permettere che lo scratch bit w abbia un qualsiasi valore iniziale.*

La costruzione di θ^4 usando solo porte logiche θ^3 si generalizza facilmente alla costruzione di θ^n usando θ^{n-1} e θ^3.

Un'ultima osservazione riguarda l'uso di `NOT` per la costruzione dei circuiti che effettuano le permutazioni, come ad esempio quello in Figura 2. Notiamo che questa ulteriore operazione non è indispensabile, dal momento che può essere sostituita da un equivalente θ^3 con opportuni valori per i bits di input a, b, c. In particolare, si realizza $\neg b$ fissando $a = 1$ e $c = 1$.

1.1 Computazioni classiche su circuiti quantistici

La porta di Toffoli può essere implementata come circuito quantistico. In questo caso l'input è dato da tre qubits e la trasformazione, analogamente al caso classico, consiste nello scambio del terzo qubit se i primi due sono 1. Ad esempio la porta di Toffoli quantistica applicata allo stato $|110\rangle$ produce lo stato $|111\rangle$. Un semplice esercizio è scrivere la matrice U corrispondente a questa trasformazione e verificare esplicitamente che U è unitaria ($U^\dagger U = I$).

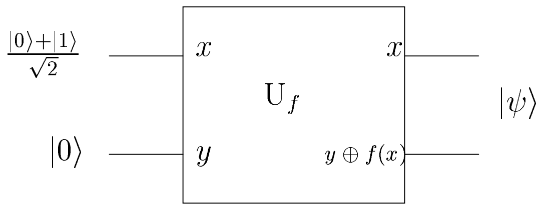

Figure 4: Circuito quantistico per valutare $f(0)$ e $f(1)$ simultaneamente

La porta di Toffoli quantistica si può quindi usare per simulare su un computer quantistico tutte le computazioni classiche irreversibili, assicurando che un computer quantistico è in grado di eseguire una qualsiasi computazione eseguibile su un computer classico.

2 Parallelismo quantistico

Su un computer quantistico si può valutare una funzione $f(x)$ su valori differenti di x contemporaneamente. Questo è noto come *parallelismo quantistico.*

Consideriamo una funzione

$$f(x) : \{0,1\} \mapsto \{0,1\}.$$

Per calcolare $f(x)$ mediante una computazione quantistica si deve definire la trasformazione di x in $f(x)$ come una trasformazione unitaria U_f. Un modo conveniente per farlo è di definire l'input come lo stato $|x, y\rangle$, detto registro dei dati, e applicare poi un'appropriata sequenza di porte logiche (che indicheremo con una scatola nera chiamata U_f) che trasformano $|x, y\rangle$ nello stato $|x, y \oplus f(x)\rangle$, detto registro target. Se $y = 0$ allora lo stato finale del secondo qubit conterrà esattamente il valore di $f(x)$.

Nel circuito in Figura 4, l'input è $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes |0\rangle$, cioè il valore di x è una sovrapposizione di 0 e 1 che si può ottenere applicando Hadamard a $|0\rangle$. Applicando U_f a questo registro di dati si ottiene

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

Questo stato contiene informazioni sia sul valore $f(0)$ che sul valore $f(1)$. Abbiamo quindi valutato f simultaneamnte su $x = 0$ e $x = 1$. Questo tipo di parallelismo è diverso da quello classico dove più circuiti (ognuno dei quali calcola $f(x)$ per un singolo valore di x) vengono eseguiti contemporaneamente.

Si può generalizzare questa procedura per calcolare funzioni su un numero arbitrario di bits usando una generalizzazione della porta di Hadamard nota come la *trasformata di Walsh-Hadamard.* Questa operazione consiste in n porte di

Hadamard che agiscono in parallelo su n qubits. Per esempio, per $n = 2$, la trasformata di Walsh-Hadamard si indica con $\mathrm{H}^{\otimes 2}$ e applicata a due qubits preparati nello stato $|0\rangle$ dà come risultato

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}.$$

In generale, il risultato di $\mathrm{H}^{\otimes n}$ applicato a n qubits nello stato $|0\rangle$ è:

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \, ,$$

done x è la rappresentazione binaria dei numeri da 0 a $2^n - 1$. Quindi la trasformata di Walsh-Hadamard produce una sovrapposizione equiprobabile di tutti gli stati della base computazionale di n qubits. Osserviamo che per ottenere una sovrapposizione di 2^n stati servono soltanto n porte logiche H.

La valutazione parallela di una funzione, $f(x)$ con input x di n bits e 1 bit come output, può quindi essere eseguita da un circuito simile a quello in Figura 4, con $n+1$ qubits in input preparati nello stato $|0\rangle^{\otimes n} |0\rangle$. Si applica quindi Hadamard ai primi n qubits e successivamente il circuito U_f. Il risultato sarà

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \, |f(x)\rangle \, .$$

Il parallelismo quantistico non è direttamente utilizzabile nel senso che non è possibile ottenere tutti i valori calcolati con una sola misurazione: la misurazione dello stato $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \, |f(x)\rangle$ darà il valore di $f(x)$ per un *singolo* valore di x. Il parallelismo quantistico si può tuttavia sfruttare per estarre informazioni riguardanti più di un valore di $f(x)$.

2.1 Algoritmo di Deutsch

L'algoritmo di Deutsch mostra come attraverso la valutazione parallela di una funzione su tutti i suoi inputs si possano determinare proprietà globali della funzione come, per esempio, quella di essere una funzione *costante* o *bilanciata*.

Considera il circuito in Figura 5.

L'input del circuito che calcola la funzione f sono ora i qubits risultanti dall'applicazione di Hadamard agli stati $|0\rangle$ e $|1\rangle$. Tale input è quindi

$$|\psi_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Applicando U_f allo stato $|x\rangle \, (|0\rangle - |1\rangle)/\sqrt{2}$ si ottiene $(-1)^{f(x)} \, |x\rangle \, (|0\rangle - |1\rangle)/\sqrt{2}$. Infatti, poichè U_f non modifica $|x\rangle$, se $f(x) = 0$ allora il risultato è $|x\rangle \, (|0\rangle - |1\rangle)/\sqrt{2}$, mentre se $f(x) = 1$ il risultato è $|x\rangle \, (|1\rangle - |0\rangle)/\sqrt{2}$.

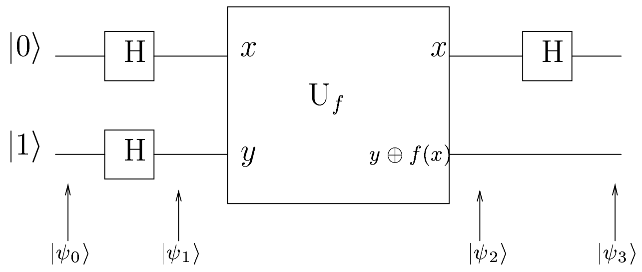

Figure 5: Circuito quantistico che implementa l'algoritmo di Deutsch

Quindi, applicando U_f a $|\psi_1\rangle$ otteniamo un risultato $|\psi_2\rangle$ che dipende da due possibilità:

$$\begin{array}{ll} \text{se } f(0)=f(1) & \pm\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \\ \text{se } f(0)\neq f(1) & \pm\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]. \end{array}$$

Nota che nella seconda alternativa, si ha che $(-1)^{f(1)} = -(-1)^{f(0)}$. Nota inoltre che $|\psi_1\rangle$ si può scrivere come

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

Pertanto U_f applicato $|\psi_1\rangle$ si può scrivere come

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

o, equivalentemente

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

cioè

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

ovvero

$$\pm\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right].$$

Applichiamo ora Hadamard al primo qubit e otteniamo $|\psi_3\rangle$ che risulta

$$\begin{array}{ll} \text{se } f(0)=f(1) & \pm|0\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \\ \text{se } f(0)\neq f(1) & \pm|1\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]. \end{array}$$

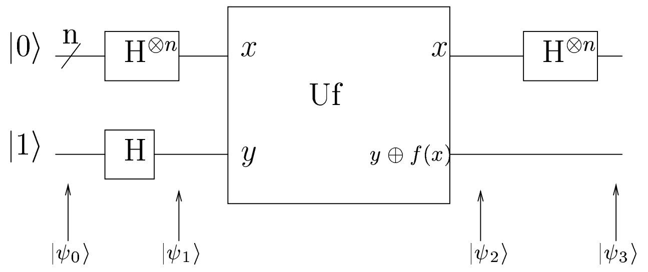

Figure 6: Circuito quantistico che implementa l'algoritmo di Deutsch-Jozsa

A questo punto osserviamo che $f(0) \oplus f(1) = 0$ se $f(0) = f(1)$, altrimenti $f(0) \oplus f(1) = 1$. Possiamo quindi scrivere il risultato in maniera più coincisa come

$$|\psi_3\rangle = \pm\,|f(0) \oplus f(1)\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right].$$

Attraverso una misurazione del primo qubit possiamo quindi determinare con certezza (la probabilità associata al primo qubit è 1) il valore di $f(0) \oplus f(1)$ e quindi se la funzione f è *costante* oppure *bilanciata*. Per far questo abbiamo dovuto valutare f una volta sola.

2.2 Algoritmo di Deutsch-Jozsa

L'algoritmo precedente si può estendere a funzioni booleane su n bits. Consideriamo una funzione $f : \{0,1\}^n \mapsto \{0,1\}$ e supponiamo di sapere che f può essere o costante oppure *bilanciata* (cioè assume valore 0 su esattamente metà degli inputs e valore 1 sulla restante metà). Usando un algoritmo classico, nel caso peggiore abbiamo bisogno di valutare la funzione su almeno $2^{n-1} + 1$ valori per poter essere in grado di stabilire con certezza se f è costante o bilanciata. L'algoritmo quantistico di Deutsch-Jozsa ci permette di stabilirlo in un solo passo. Il circuito quantistico che implementa questo algoritmo è dato in Figura 6. A differenza del circuito in Figura 5, l'input x della funzione è dato da n qubits preparati nello stato $|0\rangle$, che chiameremo il registro dei dati. Il qubit target, destinato a contenere il risultato di $f(x)$, è invece preparato nello stato $|1\rangle$. Il registro iniziale è quindi

$$|\psi_0\rangle = |0\rangle^{\otimes n}\,|1\rangle\,,$$

dove $|0\rangle^{\otimes n}$ indica il prodotto tensore di n qubits $|0\rangle$.

Al registro dei dati $|0\rangle^{\otimes n}$ viene applicata la trasformazione di Walsh-Hadamard, $\mathrm{H}^{\otimes n}$, per produrre una sovrapposizione equiprobabile di tutti i 2^n stati della base computazionale.

Indicando con $x, y \in \{0,1\}^n$ sequenze di bits di lunghezza n, abbiamo dimostrato che

$$H^{\otimes n}\,|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y}\,|y\rangle\,,$$

dove $x \cdot y$ è $\sum_{j=1}^{n} x_i y_i$ modulo 2 (cf. Esercizio4 dell'esercitazione 1). In particolare, risulta

$$H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle .$$

Gli stati $|\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle$ nel circuito in figura risultano quindi:

$$\begin{aligned}
|\psi_1\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \\
|\psi_2\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \\
|\psi_3\rangle &= \sum_{y \in \{0,1\}^n} \sum_{x \in \{0,1\}^n} \frac{(-1)^{x \cdot y + f(x)} |y\rangle}{2^n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] .
\end{aligned}$$

L'ampiezza di $|y\rangle = |0\rangle^{\otimes n}$ in $|\psi_3\rangle$ è $\sum_x (-1)^{f(x)}/2^n$. Consideriamo i due casi. Se f è costante, tale valore risulta 1 o -1. Questo significa (per la condizione di normalizzazione) che le ampiezze associate a tutti gli altri stati sono 0. Una qualsiasi misurazione del registro dei dati produce lo stato $|0\rangle^{\otimes n}$ con probabilità 1. Se invece f è bilanciata, l'ampiezza di $|y\rangle = |0\rangle^{\otimes n}$ risulta 0. Ciò significa che il risultato di una misurazione deve essere diverso da zero su almeno un qubit del registro dei dati. Dopo la misurazione siamo quindi in grado di stabilire con certezza se la funzione è costante o bilanciata.

Abbiamo ottenuto quindi una riduzione esponenziale della complessità rispetto ad un qualsiasi algoritmo classico.

A Richiami di Algebra Lineare

A.1 Prodotto tensore di matrici

Dati due operatori lineari con matrici di rappresentazione

$$M : \mathbb{C}^k \mapsto \mathbb{C}^k \text{ e } N : \mathbb{C}^l \mapsto \mathbb{C}^l ,$$

rispetto alle basi standard di $\mathbb{C}^k$ e $\mathbb{C}^l$, il prodotto tensore di M ed N è l'operatore lineare su $\mathbb{C}^{kl}$ con matrice di rappresentazione

$$M \otimes N : \mathbb{C}^{kl} \mapsto \mathbb{C}^{kl}$$

definita da

$$M \otimes N = \begin{bmatrix} M_{11}N & M_{12}N & \dots & M_{1k}N \\ M_{21}N & M_{22}N & \dots & M_{2k}N \\ \vdots & \vdots & \vdots & \vdots \\ M_{k1}N & M_{k2}N & \dots & M_{kk}N \end{bmatrix} ,$$

dove M_{ij} è l'elemento di indici i, j della matrice M e $M_{ij}N$ è la matrice $l \times l$ ottenuta moltiplicando N per il numero complesso M_{ij}.

Esempio A.1 *Il prodotto tensore delle matrici*

$$M = \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} \ e \ N = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$$

è la matrice

$$M \otimes N = \begin{bmatrix} & M_{12}N & \dots & M_{1k}N \\ M_{21}N & M_{22}N & \dots & M_{2k}N \\ \vdots & \vdots & \vdots & \vdots \\ M_{k1}N & M_{k2}N & \dots & M_{kk}N \end{bmatrix}.$$

Esercizio A.2 *Dimostrare le seguenti proprietà:*

1. $(M \otimes N)(v \otimes w) = (Mv) \otimes (Nw)$.
2. $(\alpha M + \alpha' M') \otimes (\beta N + \beta' N') =$
   $\alpha\beta M \otimes N + \alpha\beta' M \otimes N' + \alpha'\beta M' \otimes N + \alpha'\beta' M' \otimes N'$.
3. $(M \otimes N)(M' \otimes N') = (MM') \otimes (NN')$.
4.
$$\begin{aligned} (M \otimes N)^* &= M^* \otimes N^*, \\ (M \otimes N)^T &= M^T \otimes N^T, \\ (M \otimes N)^\dagger &= M^\dagger \otimes N^\dagger. \end{aligned}$$

5. *Se M ed N sono unitarie (invertibili), allora anche $M \otimes N$ è unitaria (invertibile).*