Quantum Computing

Lecture $|06\rangle$

The Deutsch Algorithm

Paolo Zuliani

SAPIENZA
UNIVERSITÀ DI ROMA

zuliani@di.uniroma1.it

Outline

- Quantum gates
- Reversible computing
- Deutsch’s problem
- Quantum parallelism
- Deutsch’s algorithm

Quantum Gates

- Quantum gates are so-called ***unitary*** transforms that preserve the vector norm. U is unitary if and only if

$$\|Uv\| = \|v\| \qquad \text{for all vectors } v$$

- Equivalently:

$$UU^\dagger = U^\dagger U = I$$

where $U^\dagger$ is the *adjoint* of U, given by $U^\dagger_{i,j} = U^*_{j,i}$ for $i \neq j$ and $U^\dagger_{i,j} = U_{i,j}$ otherwise.

The adjoint can be defined for *any* complex matrix, *e.g.*:

$$A = \begin{pmatrix} 1+2i & 1+i \\ -3i & -1 \end{pmatrix} \qquad A^\dagger = \begin{pmatrix} 1+2i & \mathbf{3i} \\ \mathbf{1-i} & -1 \end{pmatrix}$$

Why Is Quantum Evolution Unitary?

Schrödinger's equation is a *linear* differential equation (*H* is a self-adjoint operator and $v(t)$ is a complex function of time)

$$i\hbar \frac{dv(t)}{dt} = Hv(t)$$

It can be shown that, for $t_2 \geq t_1$:

$$v(t_2) = e^{-\frac{i}{\hbar}H(t_2-t_1)}v(t_1)$$

where the (linear) operator

$$U(t_2, t_1) = e^{-\frac{i}{\hbar}H(t_2-t_1)} = \sum_{j=1}^{m} e^{-\frac{i}{\hbar}\lambda_j(t_2-t_1)} P_j$$

is **unitary**.

H has *m* eigenvalues $\lambda_1 \ldots \lambda_m$

Quantum Gates

- Any quantum gate *G*, being unitary, thus admits an *inverse* gate that 'uncomputes' *G*!

$$GG^\dagger = G^\dagger G = I = \text{“do nothing”}$$

- Therefore, quantum computing (except measurement) is ***reversible***.
- Most classical computations are NOT reversible!
- Does this mean that quantum computers cannot run classical algorithms?

Reversible Classical Computing

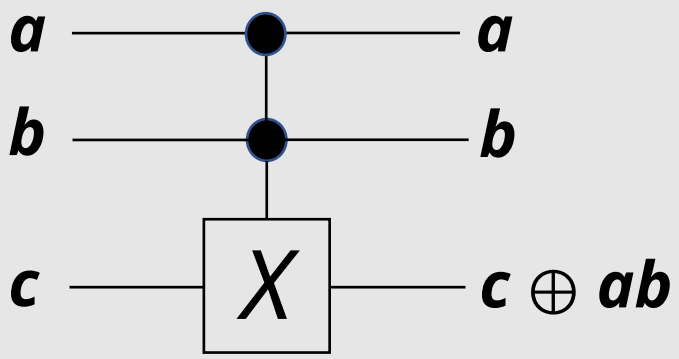
- 1963: Lecerf defined a *reversible*, deterministic Turing machine that is as efficient as a regular Turing machine
- 1973: Bennett independently did the same thing
- 2000: Z showed how to make imperative (*probabilistic* and *nondeterministic*) programs reversible

- Thus, reversibility does not pose an issue for computing!
- Essentially, one only needs more memory/larger circuits to store the 'history' of the computation.

Reversible Classical Computing

- The Toffoli gate (1980) is ***reversible***

Inputs			Outputs		
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



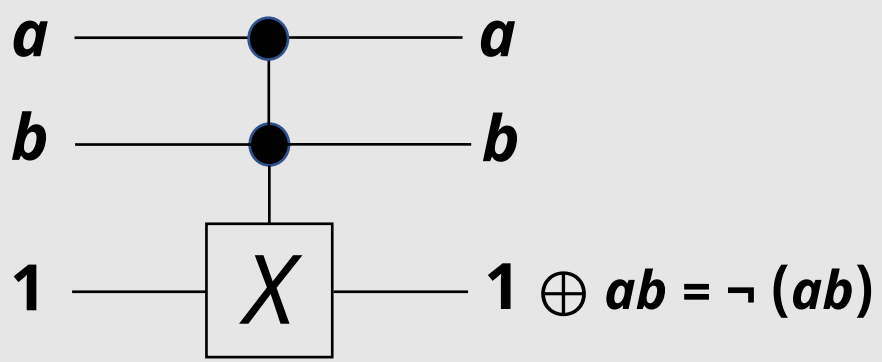
Recall the XOR

a	***b***	***a⊕b***
0	0	0
0	1	1
1	0	1
1	1	0

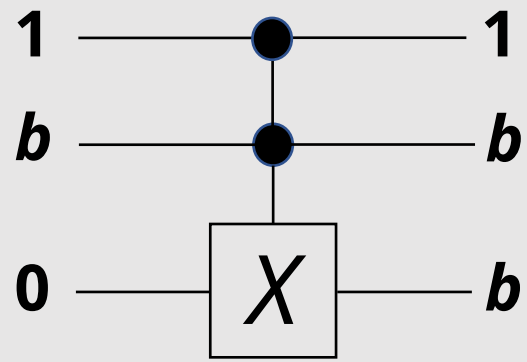
The Toffoli gate T is self-inverse: $T \cdot T = I$

Reversible Classical Computing

The NAND gate



The FANOUT gate



Any Boolean circuit can then be implemented using only Toffoli gates. Hence, the Toffoli gate is ***universal*** for classical computing.

Reversible Classical Computing

Reversibility in programs (Z, 2000):

$pGCL$ atomic statement S	reversible statement S_r	inverse statement S_i
$v := e$	**push** v ; $v := e$	**pop** v
skip	**skip**	**skip**
$pGCL$ constructor C	reversible constructor C_r	inverse constructor C_i
R ; S	R_r ; S_r	S_i ; R_i
while c **do** S **od**	**push** b ; **push** F ; **while** c **do** S_r ; **push** T **od**	**pop** b ; **while** b **do** S_i ; **pop** b **od** ; **pop** b
$R \triangleleft c \triangleright S$	**push** b ; $(R_r$; **push** T $) \triangleleft c \triangleright (S_r$; **push** $F)$	**pop** b ; $(R_i \triangleleft b \triangleright S_i)$; **pop** b
$R \square S$	**push** b ; $(R_r$; **push** $T) \square (S_r$; **push** $F)$	**pop** b ; $(R_i \triangleleft b \triangleright S_i)$; **pop** b
$R \ {}_p\!\oplus S$	**push** b ; $(R_r$; **push** $T) \ {}_p\!\oplus (S_r$; **push** $F)$	**pop** b ; $(R_i \triangleleft b \triangleright S_i)$; **pop** b
proc $Q(param) := body$	**proc** $Q_r(param) := body_r$	**proc** $Q_i(param) := body_i$

Formally, for any *pGCL* program *P*:

$$(P_r; P_c; P_i) \equiv (P_c; P)$$

reversible

copy input

inverse

Quantum Parallelism (Linearity)

Given any Boolean function f, one can show that the function

$$U_f : (x, y) \rightarrow (x, y \oplus f(x))$$

is ***reversible*** ⇒ ***unitary*** (⇒ a valid quantum gate!)

Example: 1-bit function:

Inputs

x	y
0	0
0	1
1	0
1	1

Outputs

x	$y \oplus f(x)$ $[f(x) = 0]$	$y \oplus f(x)$ $[f(x) = 1]$
0	0	1
0	1	0
1	0	1
1	1	0

Thus: $U_f(x, 0) = (x, f(x))$ and $U_f(x, 1) = (x, \neg f(x))$

Quantum Parallelism (Linearity)

$U_f : (x, y) \rightarrow (x, y \oplus f(x))$ can be implemented unitarily, thus can be applied to qubits:

$$U_f : |x \otimes y\rangle \rightarrow |x \otimes (y \oplus f(x))\rangle$$

$$U_f|x \otimes 0\rangle = |x \otimes f(x)\rangle \quad U_f|x \otimes 1\rangle = |x \otimes \neg f(x)\rangle$$

Thus:

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

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

$f(0)$ and $f(1)$ are **computed in "parallel"**!!

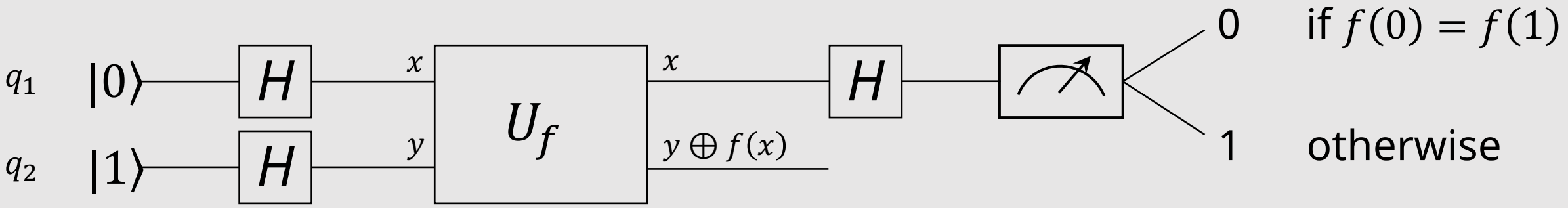
Deutsch's Problem

- A Boolean function $f: \mathcal{B} \to \mathcal{B}$
- We want to compute $f(0) \oplus f(1)$

$f(0)$	$f(1)$	$f(0) \oplus f(1)$
0	0	0
0	1	1
1	0	1
1	1	0

- Classically: we need to evaluate *f* twice
- Quantum: one evaluation of *f* suffices!

Deutsch's Algorithm (1985)



Using a "programming" notation:

$q_1, q_2 = |01\rangle;$
$q_1, q_2 = H \otimes H(q_1, q_2);$
$q_1, q_2 = U_f(q_1, q_2);$
$q_1 = H(q_1);$
$b = Measure(q_1);$

Deutsch's Algorithm

$q_1, q_2 = |01\rangle;$
$q_1, q_2 = H \otimes H(q_1, q_2);$
$q_1, q_2 = U_f(q_1, q_2);$
$q_1 = H(q_1);$
$b = Measure(q_1);$

$|01\rangle$

Apply $q_1, q_2 = H \otimes H(q_1, q_2)$

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

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

For the next step: note (and verify!) that

$$U_f(|a\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}) = (-1)^{f(a)}|a\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$$

Deutsch's Algorithm

$q_1, q_2 = |01\rangle;$
$q_1, q_2 = H \otimes H(q_1, q_2);$
$q_1, q_2 = U_f(q_1, q_2);$
$q_1 = H(q_1);$
$b = Measure(q_1);$

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

Apply $q_1, q_2 = U_f(q_1, q_2)$

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

$$\begin{cases} \pm \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) = f(1) \\ \pm \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) \neq f(1) \end{cases}$$

Apply $q_1 = H(q_1)$

Note that $HH = I$

$$\begin{cases} \pm |0\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) = f(1) \\ \pm |1\rangle \otimes \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) \neq f(1) \end{cases}$$

Deutsch's Algorithm

$q_1, q_2 = |01\rangle;$
$q_1, q_2 = H \otimes H(q_1, q_2);$
$q_1, q_2 = U_f(q_1, q_2);$
$q_1 = H(q_1);$
$b = Measure(q_1);$

$$\begin{cases} \pm|0\rangle \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) = f(1) \\ \pm|1\rangle \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} & \text{if} \quad f(0) \neq f(1) \end{cases}$$

Apply $b = Measure(q_1)$

If we measure 0 on q_1 we know **for sure** that $f(0) = f(1)$ (with only one evaluation of f)!!