IV. Quantum Computation

IV.1 The circuit model

a) Classical computation

Use of class. computers:

Solve problems ≡ compute functions

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

$$\underline{x} = (x_1, \dots, x_n) \longmapsto f(x_1, \dots, x_n) = \underline{y} = (y_1, \dots, y_m)$$

f depends on problem, $\underline{x}$ encodes instance of problem.

E.g.: Multiplication $(a, b) \longmapsto a \cdot b$

$$\underline{x} = (\underbrace{\underline{x}_1, \underline{x}_2}_{\text{enc. in binary}}) \Longrightarrow f(\underline{x}) = \underbrace{\underline{x}_1 \cdot \underline{x}_2}_{\text{enc. in binary}}$$

Factorization: $\underline{x}$: integer; $f(\underline{x})$: list of prime factors (w/ suitable encoding)

Each problem is encoded by a <u>family</u> of functions

$$f \equiv f^{(n)} : \{0,1\}^n \longrightarrow \{0,1\}^m \; ; \; m = \text{poly}(n); \; n \in \mathbb{N}.$$

<u>What ingredients</u> do we need to compute a general function f?

(i) $f : \{0,1\}^n \longrightarrow \{0,1\}^m$

$$f(\underline{x}) = (f_1(\underline{x}), f_2(\underline{x}), \ldots, f_m(\underline{x}))$$

$$f_k : \{0,1\}^n \longrightarrow \{0,1\}$$

$\Rightarrow$ can focus on <u>Boolean functions</u> $f : \{0,1\}^n \longrightarrow \{0,1\}$.

(ii) Let $L = \{\underline{y} \mid f(\underline{y}) = 1\} = \{\underline{y}^1, \underline{y}^2, \ldots, \underline{y}^\ell\}$

Define $g_{\underline{y}}(\underline{x}) = \begin{cases} 0 \; ; \; \underline{x} \neq \underline{y} \\ 1 \; ; \; \underline{x} = \underline{y} \end{cases}$

Then, $f(\underline{x}) = g_{\underline{y}^1}(\underline{x}) \vee g_{\underline{y}^2}(\underline{x}) \vee \ldots \vee g_{\underline{y}^\ell}(\underline{x})$

"$\vee$" $\equiv$ "logical or":

$0 \vee 0 = 0$

$0 \vee 1 = 1$

$1 \vee 0 = 1$

$1 \vee 1 = 1$

<u>associative:</u> $a \vee b \vee c = (a \vee b) \vee c = a \vee (b \vee c)$

(iii) $g_y(x) = \underbrace{(y_1 = x_1)}_{\begin{cases} 1 \text{ if } y_1 = x_1 \\ 0 \text{ if } y_1 \neq x_1 \end{cases}} \wedge (y_2 = x_2) \wedge \dots$

"$\wedge$" = "logical and": $1 \wedge 1 = 1$, else 0.

(iv) $$(y_i = x_i) = \begin{cases} x_i, & \text{if } y_i = 1 \\ \neg x_i, & \text{if } y_i = 0 \end{cases}$$

"$\neg$" = "logical not": $\neg 1 = 0$, $\neg 0 = 1$.

$\Rightarrow$ $f(x)$ can be built from "and", "or", "not" gates, and <u>copy gates</u> $x \mapsto (x, x)$.

<u>"Universal gate set"</u>

(<u>Note</u>: In fact, "nand" $\neg(x \wedge y)$ or "nor" $\neg(x \vee y)$ are by themselves already universal together w/ copy.)

<u>"Circuit model" of computation</u>: $f = f^{(n)}$ can be built from a simple universal gate set w/out loops (i.e., gates are applied sequentially).
(Technical point: The circuit family for $n \in \mathbb{N}$ must be generatable in an efficient way, e.g. a simple Haskell program.)

The hardness of a problem is meas. by number $K(n)$ gates needed to compute $f^{(n)}$. ($\triangleq$ # of time steps).

Can distinguish different regimes:

$K(n) \sim \text{poly}(n)$: efficiently solvable (class P)

$K(n) \gtrsim \text{poly}(n)$, e.g. $K(n) \sim \exp(n)$: hard problem

Are there more hard or easy problems?

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

# of possible f: $2^{(2^n)}$ ← # of inputs; ← 0 or 1 for each input

But there are only $c^{\text{poly}(n)}$ ← # univ. gates circuits of length $\text{poly}(n)$!

$\Rightarrow$ most f cannot be computed efficiently!

Does comp. power (= what is easy) dep. on gate set?

→ No. By def., any univ. gate set can simulate any other gate set of few-qubit gates w/ constant overhead!

Are there other models of computation?

- CPU + RAM
- parallel computer
- "Turing machine" (tape + head ≡ "linear RAM")
- cellular automata
- ⋮

But: All known "reasonable" models of computation can simulate each other w/ poly(n) overhead ⇒ same comput. power!

<u>Church-Turing-Thesis</u>: All reasonable models of computation have the same computational power.

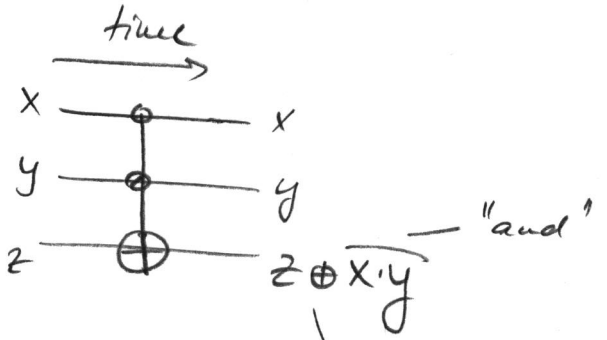
We will use <u>circuit model</u> for quantum comput.: Gates become <u>Unitaries</u>!

<u>But:</u> Unitaries reversible ⟷ class. gates irreversible

So... can we even fit class. computation into that?

Yes! – Class. computation can be turned reversible:

Toffoli gate:

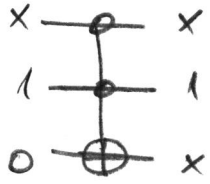


"XOR": $0 \oplus 0 = 0$, $1 \oplus 1 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ (= addition mod 2)

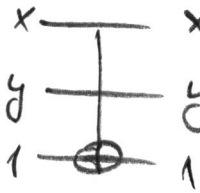
⟶ reversible!

⟶ can simulate and, or, not, copy using ancillas,

e.g.:



COPY



⟹ reversible univ. gate set (using ancillas)

Any $f(\underline{x})$ can be computed reversibly:

$$f^{\times}(\underline{x}, y) = (\underline{x},\ f(\underline{x}) \overset{\text{bitwise XOR?}}{\oplus} y)$$

(Possible w/ ancillas w/ ess. the same circuit.

Idea: Comp. f rev. w/ ancillas, XOR result w/ y,

and "un-compute" everything, i.e. reset ancillas.

Can be optimized to use few ancillas → Preskill's notes.)

⇒ Everything can be computed reversibly.

But 3-bit gate needed!

5) Quantum Circuits

Model of quantum computation – circuit model:

- System consists of qubits: tensor prod. structure
- Universal gate set $S = \{U_1, \ldots, U_k\}$ of "small" (i.e., few-qubit) gates
- build circuits
- Input: classical input $|x_1\rangle|x_2\rangle \ldots |x_n\rangle \equiv |x_1, x_2, \ldots\rangle$ in computational basis $\{|0\rangle, |1\rangle\}$, and possibly ancillas in $|0\rangle$ state.
- Output: Measure some (or all) qubits at end in comp. basis

(Notes: ○ Other meas., e.g. POVM, can be simulated w/n model.
○ Meas. at earlier time can be always postponed.)

Which gate set should we choose?

- Continuum of gates: much more rich!
- Different notions of universality
  - exact universality: any n-qubit gate U can be realized exactly
  - approximate universality: any n-qubit gate U can be approximated by gate set well. (Solovay-Kitaev Thm: ε-approx. of n-qubit gate requires $O(\text{poly}(\log(1/\varepsilon)))$ gates.)
- Turns out: 1 & 2-qubit gates alone univ. ($\leftrightarrow$ classical: $\wedge$)
- Examples of approx. univ. gate sets
  - any random 2-qubit gate (!)
  - more later, after introducing some std. gates.

- Our exact universal gate set:

(i) 1-qubit rotations about X & Z axis:

$$R_x(\phi) = e^{-iX\phi/2} \quad ; \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \; X^2 = I$$

$$R_z(\phi) = e^{-iZ\phi/2} \quad ; \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}; \; Z^2 = I$$

If $\Pi^2 = I$: $e^{-i\Pi\phi/2} = \cos\phi/2\, I - i \sin\phi/2\, \Pi$

$$\Rightarrow \quad R_x(\phi) = \begin{pmatrix} \cos\phi/2 & -i\sin\phi/2 \\ -i\sin\phi/2 & \cos\phi/2 \end{pmatrix}$$

$$R_z(\phi) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix}$$

Can also be interpreted as rotation on <u>Bloch sphere</u>, i.e. in $O(3) \cong SU(2)/\mathbb{Z}_2$, about x & z by ϕ.

R_x & R_z can generate <u>any rotation</u> in $SU(2)/\mathbb{Z}_2 \cong O(3)$ (→ Euler angles!)

$$U = R_x(\alpha) R_z(\beta) R_x(\gamma) : \text{all of } SU(2).$$

(ii) one 2-qubit gate (almost all would do!)

Typ. we choose "controlled-NOT" ≡ CNOT

$$\text{CNOT} = \quad \begin{array}{ll} x & \longrightarrow\ x \\ y & \longrightarrow\ x \oplus y \end{array} \quad = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

CNOT flips y iff $x = 1$: classical gate!

Can show: This gate set can create any n-qubit U exactly (of course not efficiently, e.g. by parameter counting).

Some important gates + identities (→ HW!)

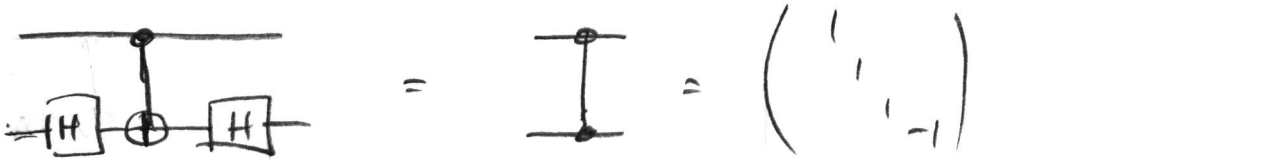
Hadamard gate: $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$; $H = H^\dagger$, $H^2 = I$

$$H R_x(\phi) H = R_z(\phi)$$

$$H R_z(\phi) H = R_x(\phi)$$

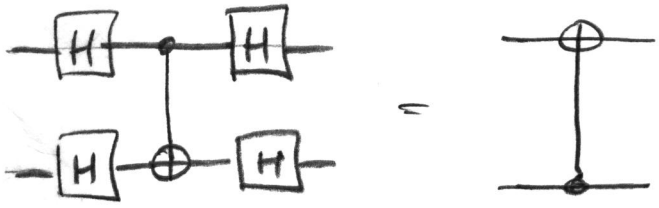
Graphical notation:

$(X = R_x(\pi), Z = R_z(\pi))$

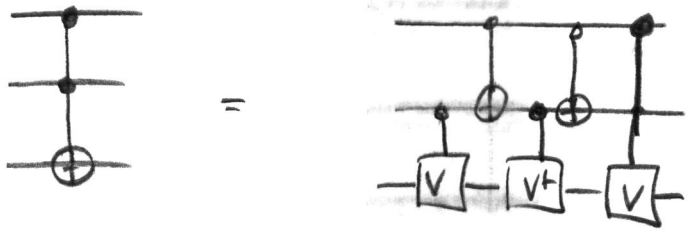


"Controlled-Z", CZ

"Controlled phase", CPHASE



Toffoli:

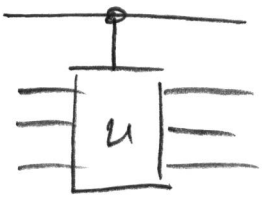


with $V = \frac{1-i}{2}(I + iX)$, and

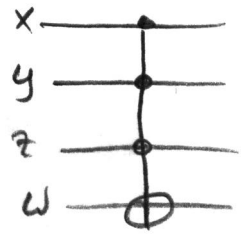
$= \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & V \end{array}\right)$

"controlled-V"

If we can build a classical U, we can also build "controlled-U".



just replace every Toffoli (class. universal!) by Toffoli w/ 3 controls:



(flip w iff $x = y = z = 1$)

Can be built from normal Toffoli (→ DKLW)

Some more approx. univ. gate sets:

- CNOT + 2 random 1-qubit gates
- CNOT + H + $T \equiv R_Z(\pi/4)$ ("$\pi/8$ gate")

IV.2. Oracle-based algorithms

a) The Deutsch algorithm

Consider $f: \{0,1\} \to \{0,1\}$

Let f be "very hard to compute" (esp., long circuit)

Want to know: Is $f(0) \stackrel{?}{=} f(1)$?

How often do we have to evaluate f (= run circuit)?

(We regard f as "black box" = "oracle":

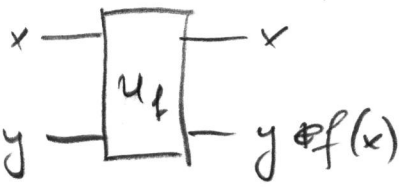
how many queries to oracle?)

Classically: 2 queries: $f(0), f(1)$.

Can quantum mechanics do better?

Consider reversible implementation of f:

$$f^R: (x,y) \longmapsto (x, y \oplus f(x))$$



$$|x\rangle|y\rangle \longmapsto |x\rangle|y \oplus f(x)\rangle$$

Try to input superpositions?

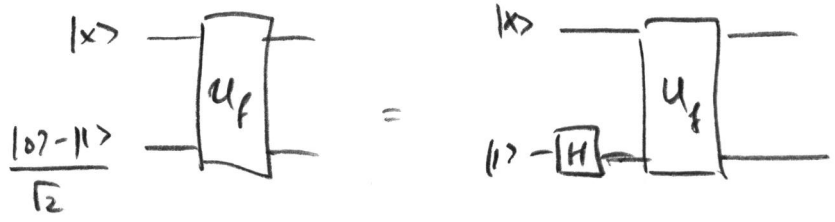
$$\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|0\rangle\right) \xrightarrow{U_f} \frac{1}{\sqrt{2}}\left(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\right)$$

→ Have evaluated f on <u>both inputs</u>!

But: how can we <u>extract</u> relevant <u>information</u>?

- Meas. qubit 1: collapse superposition!
- Meas. qubit 2: ?

Consider instead

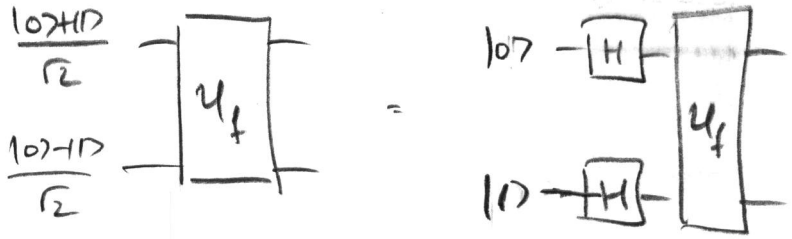


$$|x\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \xrightarrow{U_f} |x\rangle\left(\frac{|f(x)\rangle - |1\oplus f(x)\rangle}{\sqrt{2}}\right) =$$

$$= \begin{cases} f(x)=0: & |x\rangle\,\dfrac{|0\rangle-|1\rangle}{\sqrt{2}} \\ f(x)=1: & |x\rangle\,\dfrac{|1\rangle-|0\rangle}{\sqrt{2}} \end{cases} = |x\rangle\cdot\left[(-1)^{f(x)}\,\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

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

Combine:



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

→ no entanglement created (!)

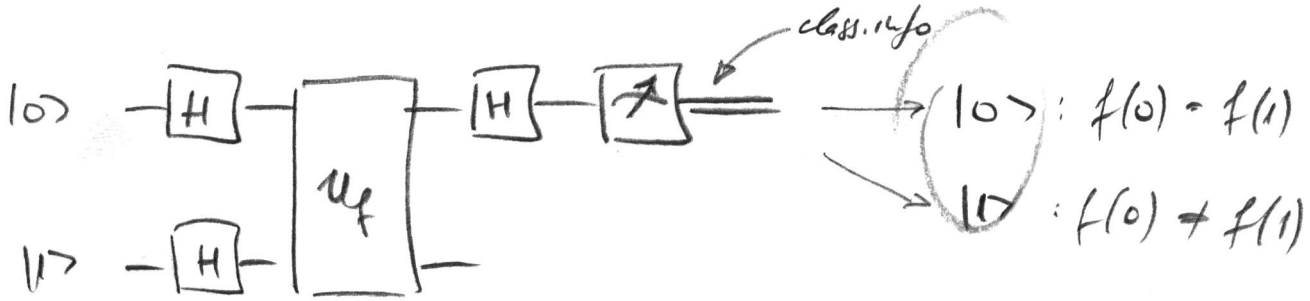
→ 2nd qubit unchanged (!!)

→ 1st qubit gets phase $(-1)^{f(x)}$

⇒ "phase kick-back" technique.

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

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



<u>One application</u> of U_f sufficient ⇒ speed-up w.r.t. classical algorithm!

Note: 2nd qubit never measured (& contains no info!)

Main ideas/points:

- Use input $\sum |x\rangle$ to evaluate f on all inputs simult.
- Need way to read out relevant info!

6) The Deutsch-Jozsa algorithm

Consider $f: \{0,1\}^n \to \{0,1\}$ w/ promise that

either $f(\underline{x}) = c \quad \forall \underline{x}$ ("f constant")

or $\left|\{\underline{x} \mid f(\underline{x})=0\}\right| = \left|\{\underline{x} \mid f(\underline{x})=1\}\right|$ ("f balanced")

Want to know: Is f constant or balanced?

Use same idea: Input $\sum |\underline{x}\rangle$ and $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$: