III. Quantum computing & algorithms

3.1. Some elements of classical computing

Key elements of modern computers go back to

Charles Babbage (1791 - 1871)

the "founding father" of the formal theory of computation is

Alan Turing (1912 - 1954)

(A) What is information?

> information is notion of a priori ignorance

If a dice player tells you that he has thrown a "5" this is only information for you if you have not watched him throwing!

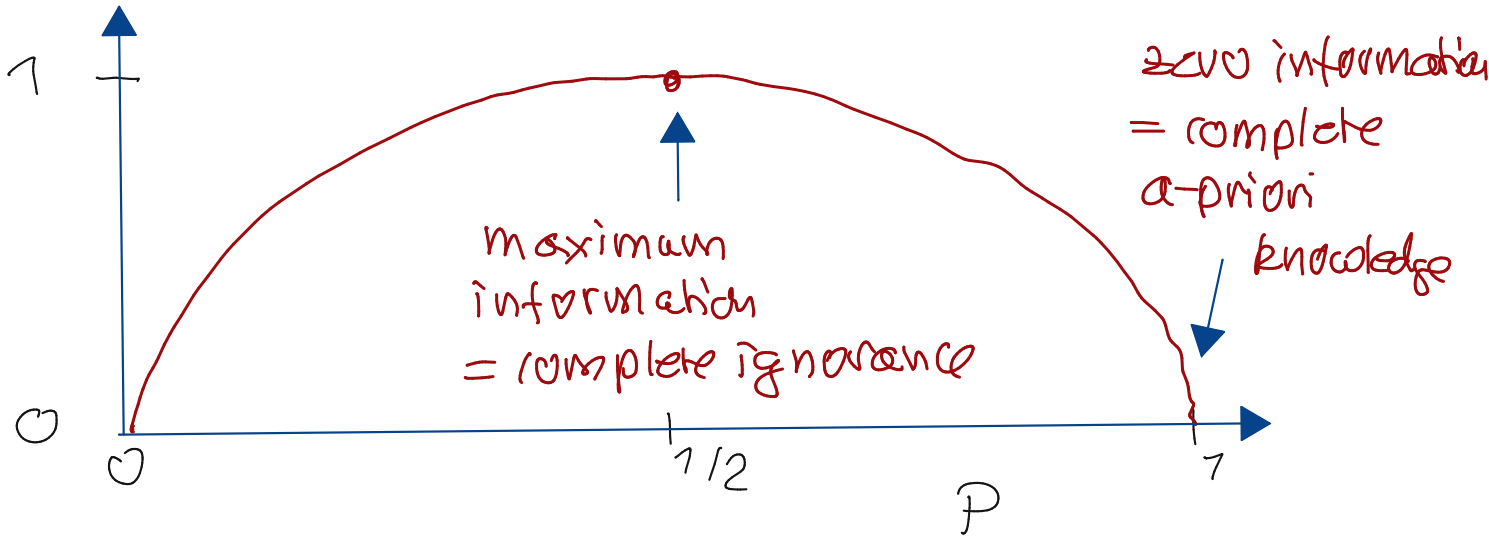
elementary unit of information is a binary decision

(72) c bit or just bit 0 1

a quantitative measure of information is Shannon entropy (see later chapters)

a-priori probability of "0" : p
-"- of "1" : $1-p$

(73) $$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$



(B) Computation & Turing machine

computation: systematic generation of a finite set of (binary) symbols with abstract meaning from a different set of (binary) symbols

this is equivalent to evaluating

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

which can in general be reduced to a decision problem 0 = NO 1 = YES

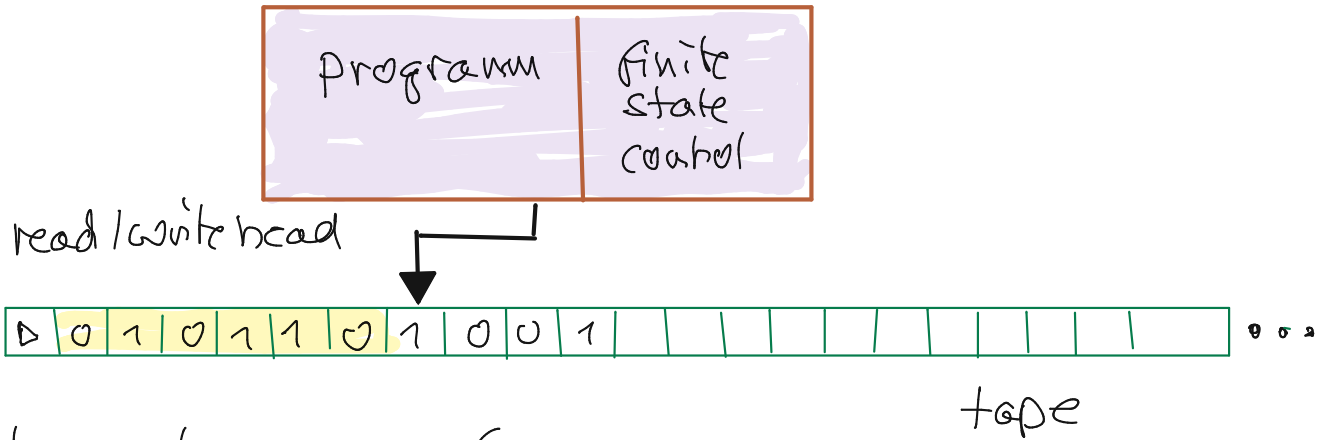
(75) $f : \{0,1\}^n \longrightarrow \{0,1\}$

early 20th century: Hilbert's Entscheidungsproblem

> Does an algorithm exists which in principle can solve all mathematical problems (decision problems)?

Hilberts conjecture: yes
Church & Turing: no!

Turing machine (invented to solve Hilberts problem)



elements:

(A) program

(B) finite state control

(C) tape

(D) read/write head

- finite state control

  finite set of internal states $q_1, q_2, \ldots, q_m$

  q_s starting state
  q_h halting state (where machine ends up in, iff computation finishes)

- tape:

  half infinite one-dimensional object with infinite sequence of tape squares labelled 0, 1, 2, 3, 4 ... containing each one symbol of a finite alphabet Γ

  for simplicity $\Gamma : \{0, 1, b\}$ b : "blank" and "▷" to mark start at left end

- read-write head

  points to one tape square

- program

  finite ordered list of program lines

  (q, X, q', X', S)

q – internal state
x – symbol at tape

→ q' – new internal state
x' – x overwritten by x'

tape head is moving left if $S = -1$
right if $S = +1$
stand still if $S = 0$

(only if head is at left-most position than it stops also if $S = -1$)

- if machine does not find a program line for given q and x:

$q \rightarrow q_h$ machine halts

Church-Turing thesis:

The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm

i.e. Turing machine is a universal computer

there are decision problems (functions) that cannot be solved on a Turing machine

"halting problem" (Turing 1936)

problem of determining from description of any computer program and arbitrary input whether the computation will stop or run forever

- remark: a machine with a finite memory will either halt or fall into periodic pattern

(C) Network model of computation

total number of decision-problem functions, eg (75)
2 outputs ; 2^n inputs $\Rightarrow$ 2^{2^n}

each of these functions can be decomposed into a sequence of elementary logic operations

input : string of bits

(76) $$X = X_1 X_2 X_3 \cdots X_n \qquad X_i \in \{0,1\}$$

Since the #number of decision functions is finite we can write ("$\vee$" logic OR)

(77) $$f(x) = f^{(1)}(x) \vee f^{(2)}(x) \vee f^{(3)}(x) \cdots f^{(m)}(x)$$

where

(78) $$f^{(a)}(x) = \begin{cases} 1 & \text{if } x \in X^{(a)} \\ 0 & \text{else} \end{cases}$$

i.e. $\underline{X}^{(a)}$ is the set of inputs x for which $f^{(a)}$ gives 1.

- assume $x = 1111\ldots1 \in \underline{X}^{(a)}$ then we can write

(79) $$f^{(a)}(x) = X_1 \wedge X_2 \wedge X_3 \cdots \wedge X_m$$

where "$\wedge$" logic AND

- if $x \in \underline{X}^{(a)}$ contains a "0" instead of "1" at some position then we have to write

(80) $$f^{(a)}(x) = X_1 \wedge X_2 \wedge \cdots \wedge (\neg X_j) \wedge \cdots$$

where "$\neg$" logic NOT

In order to evaluate eq. (77) we need an m-fold copy of the input, thus we need the operation

(87) COPY $x \rightarrow xx$

$\Rightarrow$ to evaluate an arbitrary function $f: \{0,1\}^n \rightarrow \{0,1\}$ we need a finite number of four elementary gates

COPY, NOT	AND, OR
1 bit	2 bit

If one is able to prepare a cbit in a certain state ($x=0$ or $x=1$) it is sufficient to have one 2-bit operation U and NOT as 1-bit operation

(82) $$(x,y) \xrightarrow{U} (1-x, 1-xy)$$

is equivalent to NAND (NOT AND) if we ignore the first output. Now for $y=1$ U is equivalent to COPY if we perform additionally a bit-wise NOT. i.e:

(83) $$(x,1) \xrightarrow{U} (1-x, 1-x) \xrightarrow{\text{bitwise NOT}} (x,x) \hat{=} \text{COPY}$$

U is called a universal 2-bit gate

network model of classical computation

- contains a few elementary elements (1-bit, 2-bit gates)
- requires preparation of bits in certain states

(D) reversible computing

Many of the considered elementary gates are not reversible. Their operation is thus fundamentally

connected to entropy production and heat!

Landauer's principle

If in a computation a single bit of information is erased there is a minimum amount of energy dissipated into the environment

$$\Delta Q_{min} = k_B T \ln 2$$

where T is the environment temperature. Equivalently the entropy of the environment is increased by at least

$$\Delta S_{min} = k_B \ln 2$$

Any computation has to increase the entropy of the environment, but can we make individual operations reversible and create entropy only be resetting the computer at the end of the computation?

reversible computation requires evaluation of invertible functions

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

every function can be made reversible by addition of ancillas

$$f(x): \{0,1\}^n \longrightarrow \{0,1\}^m \qquad m \neq n$$

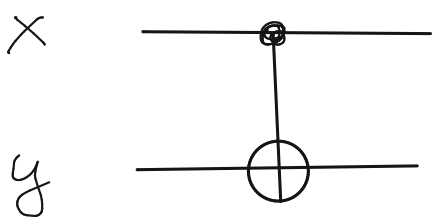
$$\hookrightarrow \tilde{f} : \{0,1\}^{n+m} \longrightarrow \{0,1\}^{m+n}$$

where

$$\tilde{f}(\underbrace{x}_{n}, \underbrace{0,0,..0}_{m}) = (\underbrace{x}_{n}, \underbrace{f(x)}_{m}) \quad (85)$$

unchanged: "control bit"

example: Controlled-NOT CNOT



control bit

target bit

rem: (classical) reversible 1 or 2-bit gates are not universal since their operation can be expressed as linear matrix operation

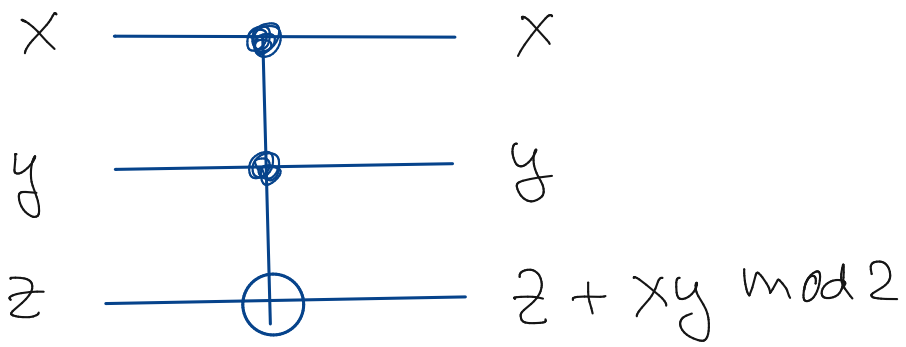
$$\begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = \underline{\underline{M}} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \quad (86)$$

with $\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and

(+) $\underline{\underline{M}} \in \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \right\}$

nonlinear functions $\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x' \\ y' \end{pmatrix}$ cannot be represented! However, reversible n-bit gates with $n \geq 3$ are universal.

example: Toffoli gate controlled-controlled NOT



3.2 Computational complexity*

* neither complete nor fully rigorous

Question: How many elementary operations are needed to evaluate $f(x)$ if the size n (# of bits) of the input is increased?

We consider the evaluation of a function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ (algorithmically) simple if the number of elementary gates needed is a polynomial in n. These functions (problems) are said to belong to the complexity class

> **P** = decision problems with n bit's input that require poly(n) elementary gates to evaluate

All problems not belonging to this class are called (algorithmically) difficult or hard

The class of problems for which the verification of the result $f(x)=1$ is simple in the above sense

> **NP** = decision problems with n bit's of input where the "yes" or "1" output can easily be verified by a witness

The verification of "1" output of a decision problem is equivalent of the verification of the result of a mathematical problem. E.g.

finding the prime factors of an integer is believed to be a hard problem. The verification is however simple as multiplication is simple.

example: travelling salesman problem

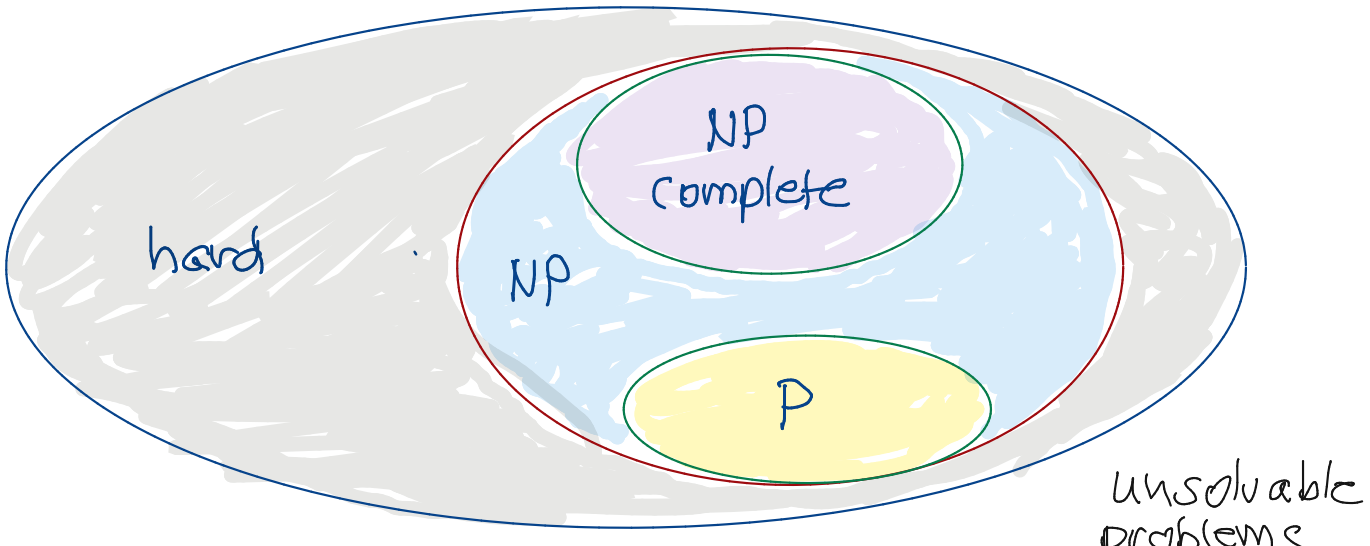
There is a special subset of NP problems, called

NP complete (NPC): problems in NP to each of which any other NP problem can be reduced to with polynomial effort

obviously $P \subseteq NP$

it is however still an open problem if

$$P \neq NP \; ?$$



example 1 for NPC: CIRCUIT-SAT

(881 $C[f(x)] = \begin{cases} 1 & \text{if } x \text{ exists such that } f(x)=1 \\ 0 & \text{if no such } x \text{ exists} \end{cases}$

theorem of Cook (1971): (NP completeness)

every problem in NP can be reduced to CIRCUIT-SAT with polynomial effort

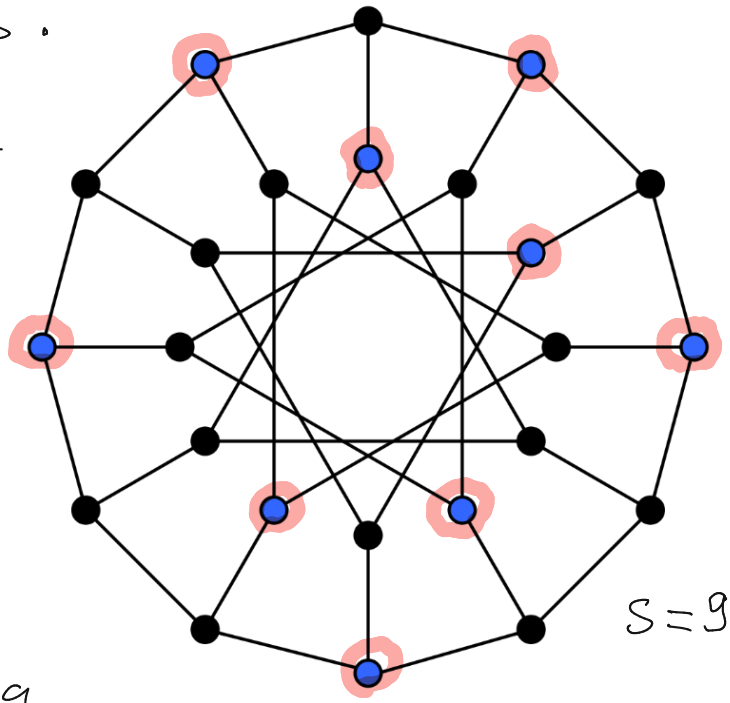
so if CIRCUIT-SAT $\in P \Rightarrow NP = P$
(the same holds for any NP complete problem)

example 2 for NPC: Maximum independent set

Problem in graph theory: independent set is a set of vertices in a graph that are not pairwise directly connected. Size of an independent set = number of vertices.

maximum independent set = maximum size

Strongly believed to be NP but not P



from Wikipedia

- factorization of integers into prime factors is believed to be not NP complete.

So far we have considered deterministic computation. It turns out however that a computation may be more efficient if it uses elements of randomness. Such a probabilistic evaluation of a decision problem gives the correct result only with finite probability

Chernikov bound

Suppose we have an algorithm for a decision problem which gives the correct answer with probability $1/2 + \varepsilon$ and the wrong answer with probability $1/2 - \varepsilon$. Then n-fold repetion yields an exponential improvement

$X_1, \dots, X_n \in [0,1]$ independent random variables

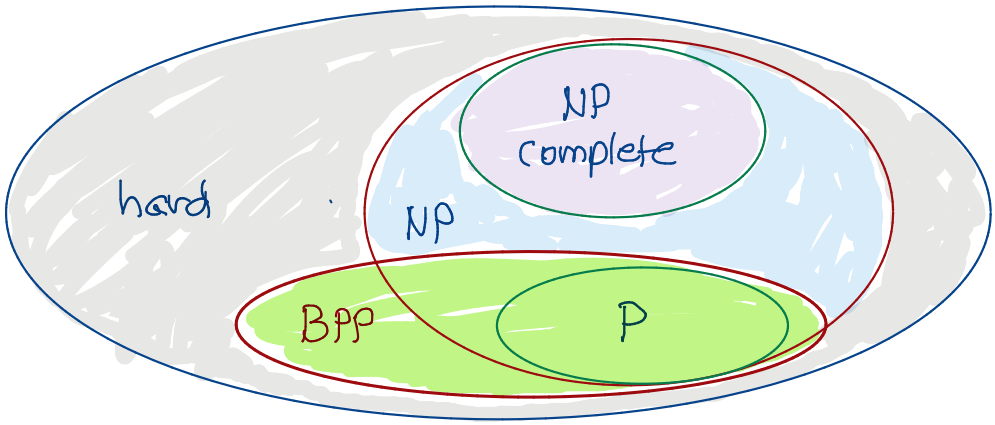
$$p(X_i = 0) = \frac{1}{2} - \varepsilon$$

$$(89) \Rightarrow p\left(\frac{1}{n}\sum_{i=1}^{n} X_i \leq 1\right) \leq e^{-2\varepsilon^2 n}$$

The class of problems that can be solved with probabilistic calculations with polynomial effort and bounded error probability is called

BPP = bounded error, probabilistic with polynomial effort

The general relation between the classes is not fully known



3.3 Quantum circuit computation

(A) Qubits

quantum system with smallest non-trivial Hilbert space $d = 2$

- spin $\frac{1}{2}$
- two-level atom
- polarization states of a photon

computational basis $\{|0\rangle, |1\rangle\}$

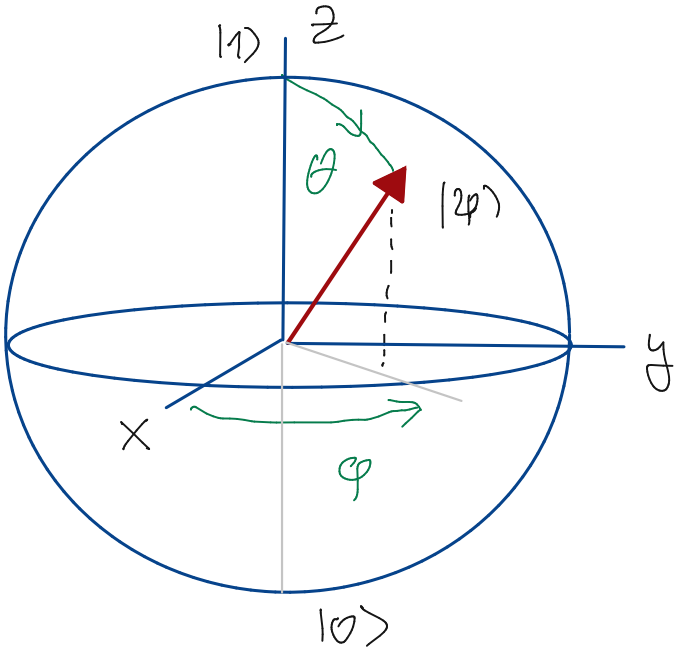
$\langle 0|1\rangle = 0$ e.g. eigenstates of $\hat{\sigma}_z$

general state

(pol) $|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|1\rangle + e^{i\varphi}\sin\frac{\theta}{2}|0\rangle\right)$

irrelevant (θ, φ) characterize qubit

Bloch sphere



perculanties of a qubit vs. bit:

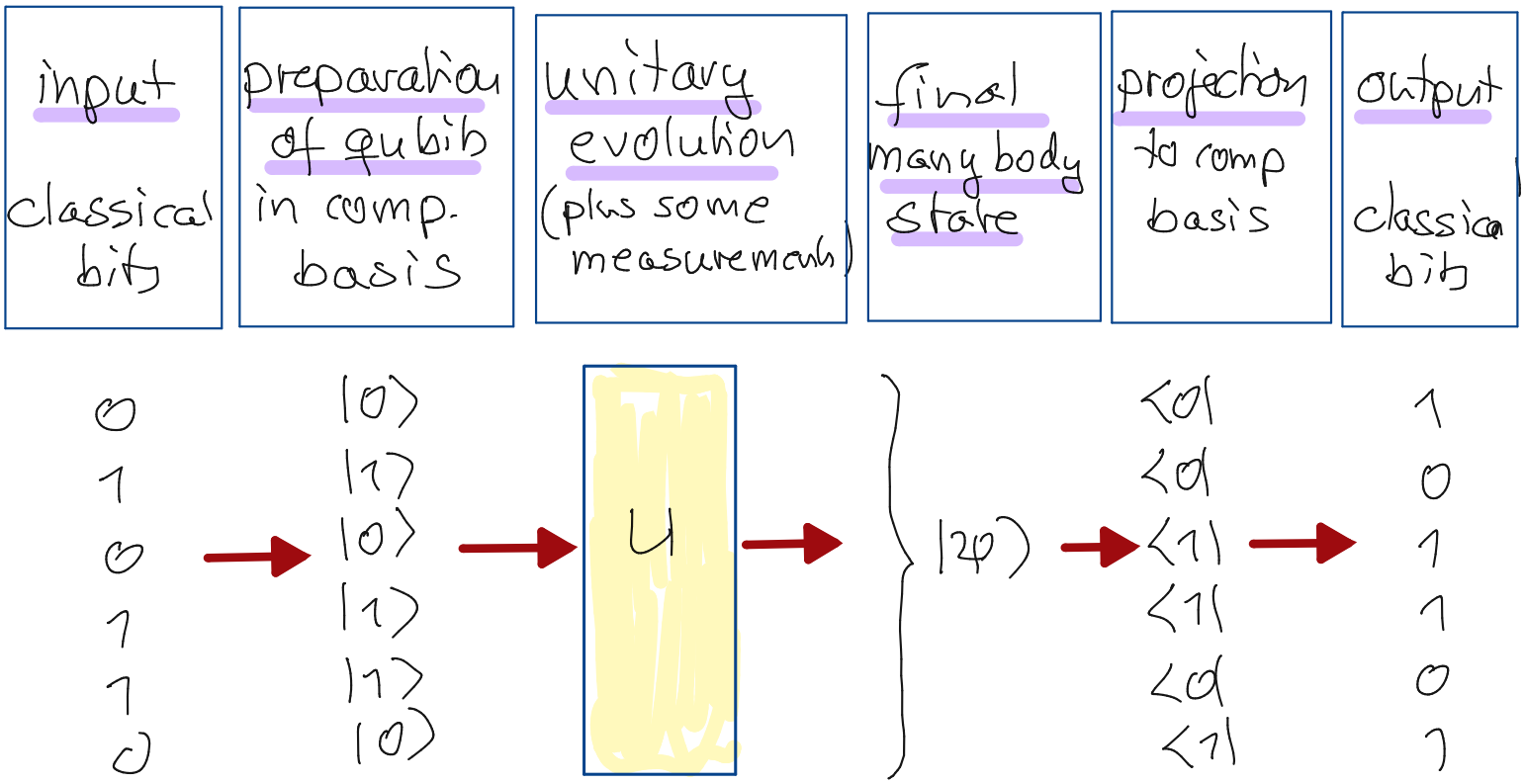
superposition !

⇒ many more possibilities

However: information that can be encoded and/or read out from a single qubit is only one bit of information !

Still in intermediate operations superpositions and in particular many qubit superpositions i.e. entanglement can improve computations.

(B) Quantum circuit model

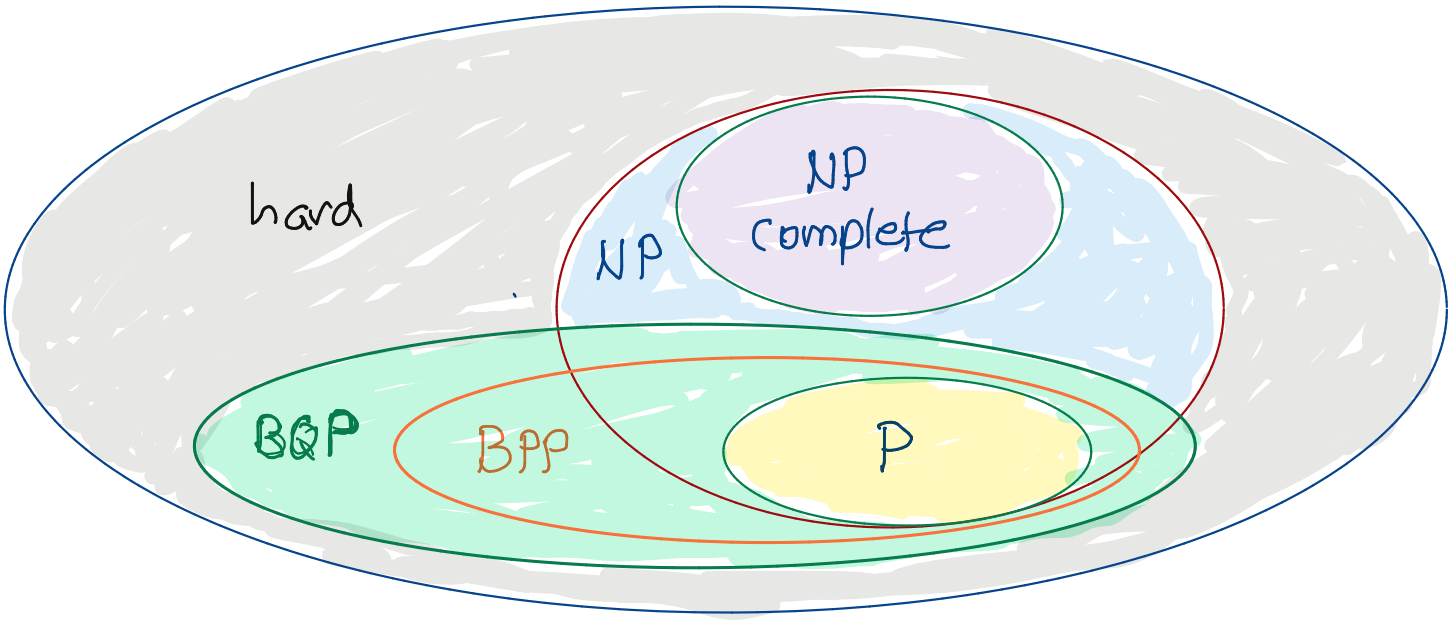


the output is in general not deterministic but probabilistic

It is believed that the class of problems that can be solved probabilistically with bounded error on a quantum computer called

BQP	bounded error probabilistic, quantum with polynomial effort

is larger than BPP !



decomposition of unitary operation into elementary units

Deutsch (1985)

Any d-dimensional unitary matrix can be decomposed into at most $d(d-1)/2$ unitary 2×2 matrices

unitary 2×2 matrices: (also hermitian)

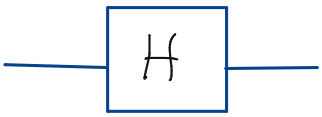
(91) $\mathbb{1}$, $\hat{\sigma}_x$, $\hat{\sigma}_y$, $\hat{\sigma}_z$

Since the 2×2 matrices can also act in the space spanned by states of two different qubits we need

1-qubit and 2-qubit gates

important 1-qubit gates

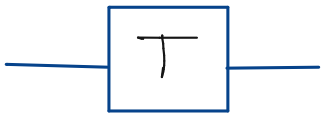
Hadamard gate



$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{matrix} |0\rangle \\ |1\rangle \end{matrix} \qquad \begin{matrix} |0\rangle & |1\rangle \end{matrix}$$

(92) $\{|0\rangle, |1\rangle\} \xrightarrow{H} \left\{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right\}$

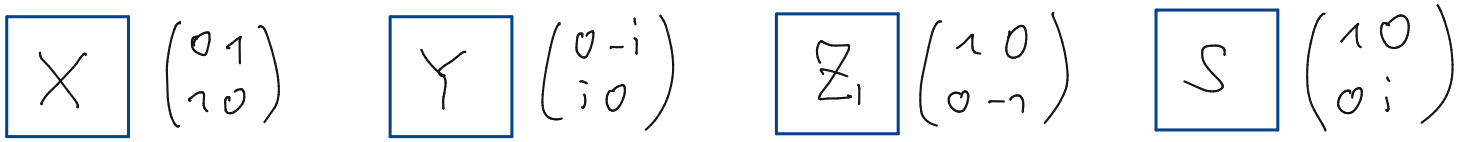
$\pi/8$ gate



$$e^{i\frac{\pi}{8}}\begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}$$

The combination of Hadamard and $\frac{\pi}{8}$ gates allows to approximate any single qubit gate with arbitrary precision

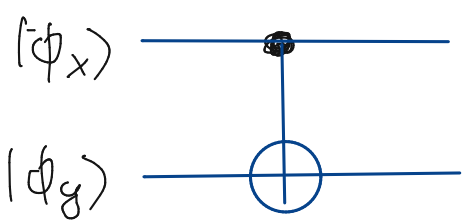
other single qubit gates



Pauli X | Pauli Y | Pauli Z | Phase

important 2-qubit gate

CNOT gate



$$\begin{matrix} |00\rangle & |01\rangle & |10\rangle & |11\rangle \end{matrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

arbitrary single qubit plus CNOT gate are universal

note that in the classical gate we needed 3bit gates (e.g. Toffoli)

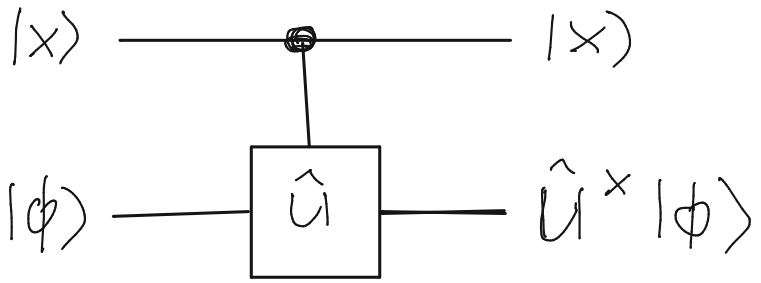
How to evaluate the action of a 2-qubit gate on a general quantum state? Map is linear in Hilbert space $\Rightarrow$ sufficient to know action on basis states, i.e.

$$|\phi_x\rangle \in \{|x=0\rangle, |x=1\rangle\}$$

$$|\phi_y\rangle \in \{|y=0\rangle, |y=1\rangle\}$$

e.g.

(93)



(C) Deutsch and Deutsch-Josza algorithm

We now consider an instructive example for the potential power of quantum computation: A problem that can be solved using a deterministic quantum algorithm more efficiently than with a deterministic classical one is the following

given a function $f: \{0,1\} \to \{0,1\}$
every such function belongs to one of the two classes

(94)

$$\begin{array}{lll} f(0)=0 & \text{or} & f(0)=1 \\ f(1)=0 & & f(1)=1 \end{array} \qquad f: \text{constant}$$

$$\begin{array}{lll} f(0)=0 & \text{or} & f(0)=1 \\ f(1)=1 & & f(1)=0 \end{array} \qquad f: \text{balanced}$$

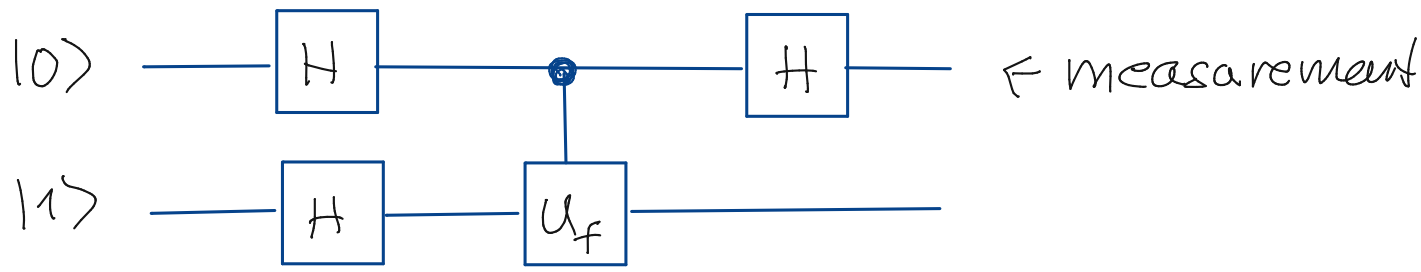
Although the classification of f into "constant" or "balanced" is only one bit of information one has to evaluate f twice to make such a classification classically!

Quantum mechanically we can make use of the possibility of superposition to reduce this to one evaluation

f is encoded in U_f $|x\rangle|y\rangle$ in comp. basis

(95) $$U_f|x\rangle|y\rangle \stackrel{!}{=} |x\rangle|y \oplus f(y)\rangle$$

use circuit (Deutsch)



(96) $|0\rangle|1\rangle \xrightarrow{\text{1st H gates}} \frac{1}{2}\left(|0\rangle+|1\rangle\right)\left(|0\rangle-|1\rangle\right) = \frac{1}{2}\sum_{x=0}^{1}|x\rangle\left(|0\rangle-|1\rangle\right)$

(97) $$U_f|x\rangle(|0\rangle-|1\rangle) = \begin{cases} |x\rangle(|0\rangle-|1\rangle) & \text{iff } f(x)=0 \\ -|x\rangle(|0\rangle-|1\rangle) & \text{iff } f(x)=1 \end{cases}$$

$$= (-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

i.e.

$$\frac{1}{2}\sum_{x=0}^{1}|x\rangle(|0\rangle-|1\rangle) \xrightarrow{U_f} \frac{1}{2}\sum_{x=0}^{1}(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

(98) $$= \pm\frac{1}{2}\begin{cases}(|0\rangle+|1\rangle)(|0\rangle-|1\rangle) & f: \text{constant} \\ (|0\rangle-|1\rangle)(|0\rangle-|1\rangle) & f: \text{balanced}\end{cases}$$

$\xrightarrow{\text{2nd H}}$

(99) $$= \pm\frac{1}{2}\begin{cases}|0\rangle(|0\rangle-|1\rangle) & f: \text{constant} \\ |1\rangle(|0\rangle-|1\rangle) & f: \text{balanced}\end{cases}$$

one measurement of 1st qubit sufficient

Speedup by superposition: quantum parallelism

first experiment: Gulde et al Nature 421, 48 (2003)

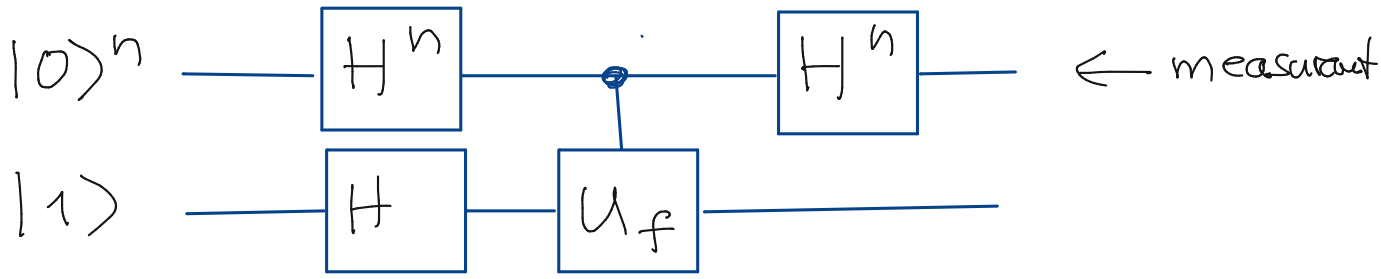
The effect is even more dramatic if we consider

(100) $f: \{0,1\}^n \longrightarrow \{0,1\}$ (Deutsch-Jozsa)

Let f be either constant, i.e. have the same output (0 or 1) for all inputs or be balanced, i.e. have the same number of outputs "0" and "1" in the output. Here again the classification of f has only one bit of information. (Of course very many possible functions of type (100) can be neither constant or balanced but we here assume that f is of either the one or the other type)

Classically we have to evaluate f $2^{n-1}+1$ times!

Quantum mechanically a single evaluation is sufficient!!



H^n: $\underbrace{H \otimes H \otimes H \otimes \cdots \otimes H}_{n}$ bitwise Hadamard

$$H^n|\underline{x}\rangle = \prod_{j=1}^{n} \frac{1}{\sqrt{2}} \sum_{y_j=0}^{1} (-1)^{x_j y_j} |y_j\rangle$$

(101)

$$= \frac{1}{2^{n/2}} \sum_{\underline{y}=0}^{2^n-1} (-1)^{\underline{x}\cdot\underline{y}} |\underline{y}\rangle$$

if $|\underline{x}\rangle = |x_1\rangle|x_2\rangle \cdots |x_n\rangle \quad x_i \in \{0,1\}$

and $|\underline{y}\rangle = \underbrace{|y_1\rangle|y_2\rangle \cdots |y_n\rangle}_{2^n \text{ possibilities}} \quad y_j \in \{0,1\}$

Here $\underline{x} \cdot \underline{y} = x_1 y_1 + x_2 y_2 + \cdots$
is a bitwise addition modulo 2

gate operation:

$$|0\rangle^n |1\rangle \xrightarrow{H^n, H} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} |\underline{x}\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

(102)
$$\xrightarrow[\text{(see (97))}]{U_f} \frac{1}{2^{n/2}} \sum_{\underline{x}=0}^{2^n-1} (-1)^{f(\underline{x})} |\underline{x}\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

$$\xrightarrow{H^n} \frac{1}{2^n} \sum_{\underline{y}=0}^{2^n-1} \sum_{\underline{x}=0}^{2^n-1} \underbrace{(-1)^{f(\underline{x})} (-1)^{\underline{x}\cdot\underline{y}}}_{(-1)^{(\underline{x}\cdot\underline{y} + f(\underline{x}))}} |\underline{y}\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

Now the first register consisting of n qubit is projected to the state

(103) $\underbrace{|0,0,0,\ldots 0\rangle}_{n}$

yes → f: constant

no → f: balanced

If f is balanced the amplitude of $|\underline{y}=0,0,\ldots 0\rangle$ is zero since then

$$\sum_{\underline{x}=0}^{2^n-1}(-1)^{(\underline{x}\cdot\underline{y}+f(\underline{x}))}=\sum_{\underline{x}=0}^{2^n-1}(-1)^{f(\underline{x})}=0 \tag{104}$$

If f is constant $f(\underline{x})=f\in\{0,1\}$

$$\sum_{\underline{x}=0}^{2^n-1}(-1)^{(\underline{x}\cdot\underline{y}+f(\underline{x}))}=(-1)^f\sum_{\underline{x}=0}^{2^n-1}(-1)^{\underline{x}\cdot\underline{y}} \tag{105}$$

$$=0 \quad \text{for all } \underline{y} \text{ except } \underline{y}=(0,0,0,\ldots 0)$$

□

The Deutch-Josza problem is however a problem within BPP, i.e. it can be efficiently solved probabilistically with a classical computer.

3.4 BQP ≠ BPP : the Simon problem

We will now discuss a problem which is believed to be outside of BPP can be solved with polynomial effort using a quantum algorithm:

Simon problem