Quantum Complexity Theory

Markus Lohrey

Universität Siegen

Summer semester 2024

General information

On the web page

`https://www.eti.uni-siegen.de/ti/lehre/sommer_2024/quantumct/index.html`

you find all informations, in particular:

- most recent version of the slides,
- exercise sheets,
- literature recommendations.

One remark: The lecture slides are not suitable for self studying (and may at some point be incomplete)!

Basic notations

A bit string is a sequence $b_1 b_2 \cdots b_n$ of bits $b_1, b_2, \ldots, b_n \in \{0,1\}$.

The length of the bit string $u = b_1 b_2 \cdots b_n$ is $|u| = n$.

The set of all bit strings is denoted with $\{0,1\}^*$; the set of all bit strings of length n is denoted with $\{0,1\}^n$.

A language L is a subset of $\{0,1\}^*$.

Complexity theory investigates the computational resources needed to check, whether a given $u \in \{0,1\}^*$ belongs to a certain language L.

Often we are actually interested in sets of other finite objects (e.g. numbers, matrices, finite graphs, etc.) instead of bit strings.

In such situations we assume that these objects are suitably encoded by bit strings.

Turing machines

Classical complexity theory is usually formalized using Turing machines.

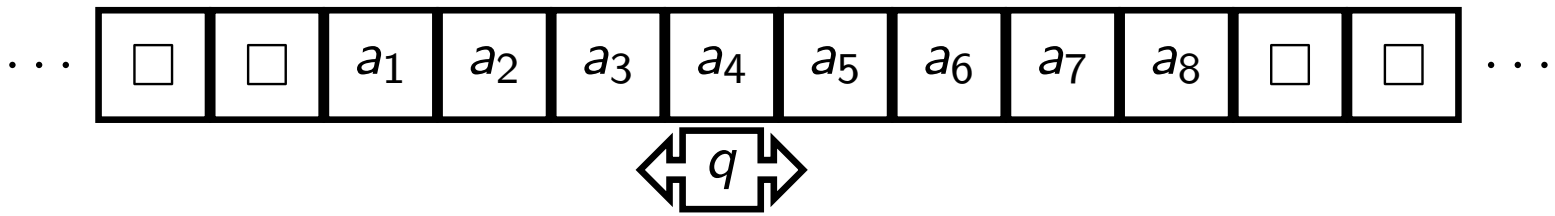
A deterministic Turing machine is a tuple $M = (Q, \Gamma, q_0, q_Y, q_N, \delta)$, where

- Q is the finite set of (control) states,
- Γ is the finite tape alphabet with $0, 1, \square \in \Gamma$,
- $\square$ is the blank symbol,
- 0 and 1 are the input symbols,
- $q_0 \in Q$ is the initial state,
- $q_Y \in Q$ is the accepting state,
- $q_N \in Q$ is the rejecting state, and
- $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ is the transition function.

We only consider deterministic Turing machines in this lecture and will mostly omit "deterministic" in the following.

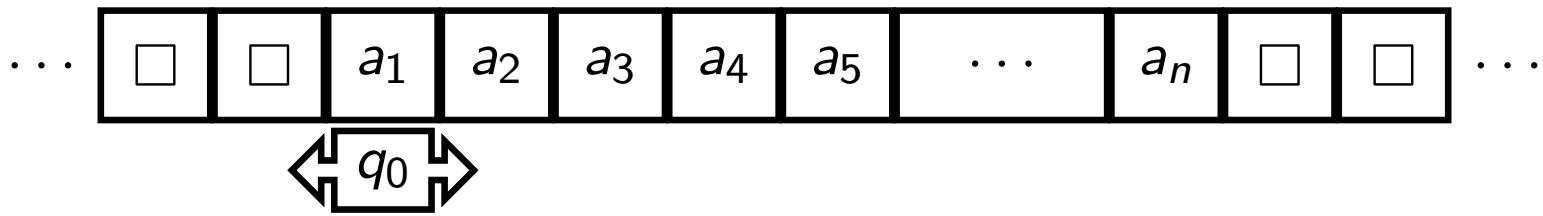
Intuition of Turing machines

- The Turing machine works on an 2-sided infinite tape of cells that contain tape symbols from Γ.
- Only finitely many cells contain a tape symbol from $\Gamma \setminus \{\square\}$.
- There is a read-write head that scans at each time instant a certain cell of the tape.
- Moreover, at each time instant the machine is in a certain control state $q \in Q$.



Intuition of Turing machines

For an input string $u = a_1 a_2 \cdots a_n$ with $a_i \in \{0, 1\}$ the machine is started in the following configuration, called the initial configuration for u:



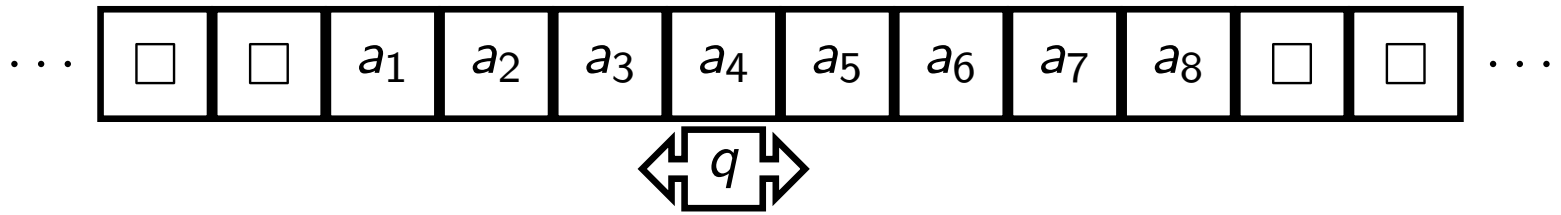
The machine moves on according to the transition function δ:

If the current control state is $q \in Q$, the currently scanned tape cell contains the symbol $a \in \Gamma$, and $\delta(q, a) = (p, b, d)$, then the machine executes the following steps:

- Replace the symbol a in the current cell by b.
- Change into control state p.
- If $d = -1/d = 1$ move the read-write head one cell to the left/right (no movement if $d = 0$).

Intuition of Turing machines

Example: Assume that $\delta(q, a_4) = (p, b, -1)$ and $\delta(p, a_3) = (r, c, 1)$:

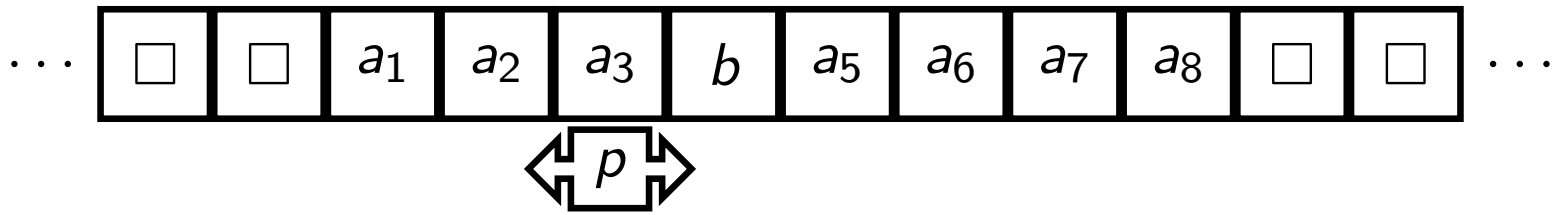


The machine stops when it reaches the accepting state q_Y or the rejecting state q_N.

The language $L(M)$ accepted by the machine M consists of all strings $u \in \{0, 1\}^*$ such that M finally reaches the accepting state q_Y when started in the initial configuration for u.

Intuition of Turing machines

Example: Assume that $\delta(q, a_4) = (p, b, -1)$ and $\delta(p, a_3) = (r, c, 1)$:

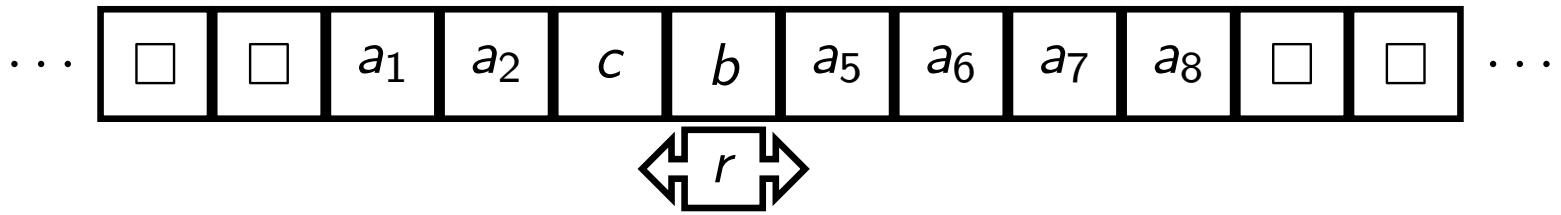


The machine stops when it reaches the accepting state q_Y or the rejecting state q_N.

The language $L(M)$ accepted by the machine M consists of all strings $u \in \{0, 1\}^*$ such that M finally reaches the accepting state q_Y when started in the initial configuration for u.

Intuition of Turing machines

Example: Assume that $\delta(q, a_4) = (p, b, -1)$ and $\delta(p, a_3) = (r, c, 1)$:



The machine stops when it reaches the accepting state q_Y or the rejecting state q_N.

The language $L(M)$ accepted by the machine M consists of all strings $u \in \{0, 1\}^*$ such that M finally reaches the accepting state q_Y when started in the initial configuration for u.

The class **P**

A (deterministic) polynomial time machine (PTM for short) is a Turing machine M for which there is a polynomial $p(n)$ such that:

For every input string $u \in \{0,1\}^n$ the machine M stops after at most $p(n)$ computation steps.

The class **P** (deterministic polynomial time) is the class of all languages $L(M)$ such that M is a deterministic polynomial time machine.

Traditionally **P** is identified with the class of those languages that can be decided in an efficient way.

In the definition of **P** one can replace the Turing machine model by more practical models of computations (e.g. register machines).

The class **P**

Example: An important problem in **P** is the circuit value problem:

- Input: a boolean circuit C.
- Question: Does C evaluate to 1?

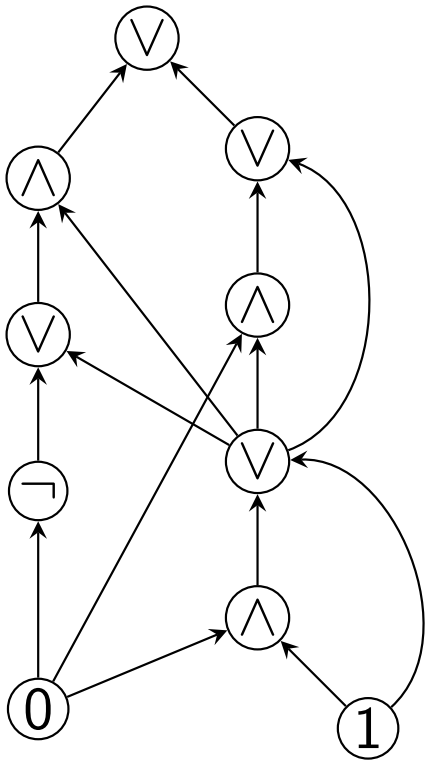
Example of a boolean circuit:

The class **P**

Example: An important problem in **P** is the circuit value problem:

- Input: a boolean circuit C.
- Question: Does C evaluate to 1?

Example of a boolean circuit:

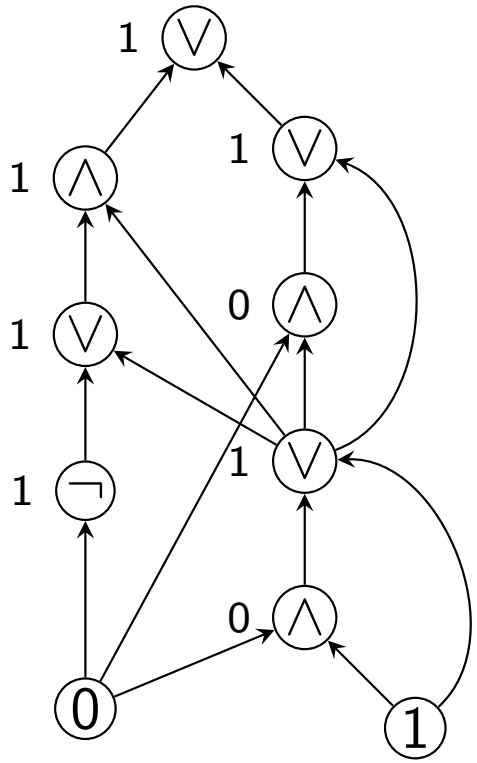


The class **P**

Example: An important problem in **P** is the circuit value problem:

- Input: a boolean circuit C.
- Question: Does C evaluate to 1?

Example of a boolean circuit:



The class **NP**

In the following we need to consider PTMs that take two input strings $u, v \in \{0,1\}^*$.

For this, we assume some encoding $\langle u, v \rangle \in \{0,1\}^*$ of u and v into a single bit string.

One possibility: if $u = a_1 a_2 \cdots a_n$ then $\langle u, v \rangle = a_1 0 a_2 0 \cdots a_{n-1} 0 a_n 1 v$.

The class **NP** consists of all languages L, for which there is a PTM M and a polynomial $r(n)$ such that the following hold for every $u \in \{0,1\}^n$:

- If $u \in L$ then there is $v \in \{0,1\}^{r(n)}$ such that $\langle u, v \rangle \in L(M)$.
- If $v \notin L$ then $\langle u, v \rangle \notin L(M)$ for every string $v \in \{0,1\}^{r(n)}$.

NP stands for non-deterministic polynomial time; it is usually defined by non-deterministic polynomial time Turing machines.

The class **NP**

The string $v \in \{0,1\}^{r(n)}$ with $\langle u, v \rangle \in L(M)$ in case $u \in L$ can be seen as a proof for the fact that $u \in L$.

Intuitively, **NP** contains all languages L for which membership in L is equivalent to the existence of an efficiently verifiable short proof.

Example: Consider boolean formulas such as for instance $F(x_1, x_2, x_3, x_4) = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$.

A boolean formula F is satisfiable if one can set the x_i to truth values (0 or 1) such that the formula evaluates to 1.

SAT is the set of all satisfiable boolean formulas.

SAT $\in$ **NP**: the proof for a satisfiable formula $F(x_1, \ldots, x_n)$ is a bit string $a_1 a_2 \cdots a_n \in \{0,1\}^n$ such that $F(a_1, \ldots, a_n)$ is true.

The class **BPP**

The class **BPP** consists of all languages L for which there is a PTM M and a polynomial $r(n)$ such that the following hold for every $u \in \{0,1\}^n$:

- If $u \in L$ then $\langle u, v \rangle \in L(M)$ for $\geq \frac{2}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.
- If $u \notin L$ then $\langle u, v \rangle \in L(M)$ for $\leq \frac{1}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.

BPP stands for bounded-error probabilistic polynomial time.

The string $v \in \{0,1\}^{r(n)}$ is also called the random string.

Intuition: If one randomly sets the bits in $v \in \{0,1\}^{r(n)}$ then with probability $\geq 2/3$ the machine M correctly tells us whether $u \in L$.

Many researchers view **BPP** as the class of those languages that can be decided in an efficient way.

The class **BPP**

Probability amplification:

- Run the machines M k times with independently chosen random strings $v_1, \ldots, v_k \in \{0,1\}^{r(n)}$.
- At the end the input u is accepted if $\langle u, v_i \rangle \in L(M)$ for at least $k/2$ many $i \in [1, k]$.
- **Exercise:** Show that the error probability of this new algorithm is $2^{-\Theta(k)}$.

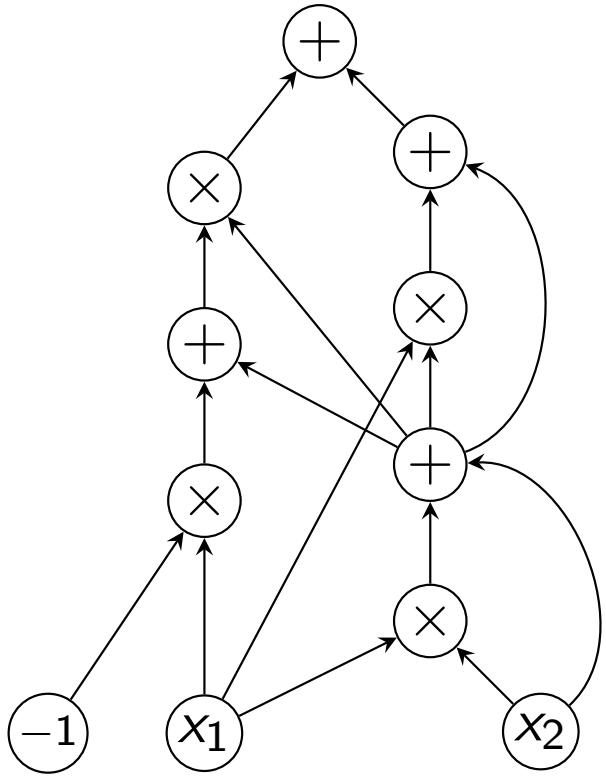
  Hint: use the Chernoff bound.

The class **BPP**

Example: A famous problem in **BPP** that is not known to be **P** is polynomial identity testing (**PIT**):

- Input: an arithmetic circuit C.
- Question: Does C evaluate to the zero polynomial?

An arithmetic circuit:



The class **PSPACE**

A polynomial space machine is a Turing machine M for which there is a polynomial $p(n)$ such that:

For every input string $u \in \{0,1\}^n$ the read-write head never moves more than $p(n)$ cells to the left or right of its initial position.

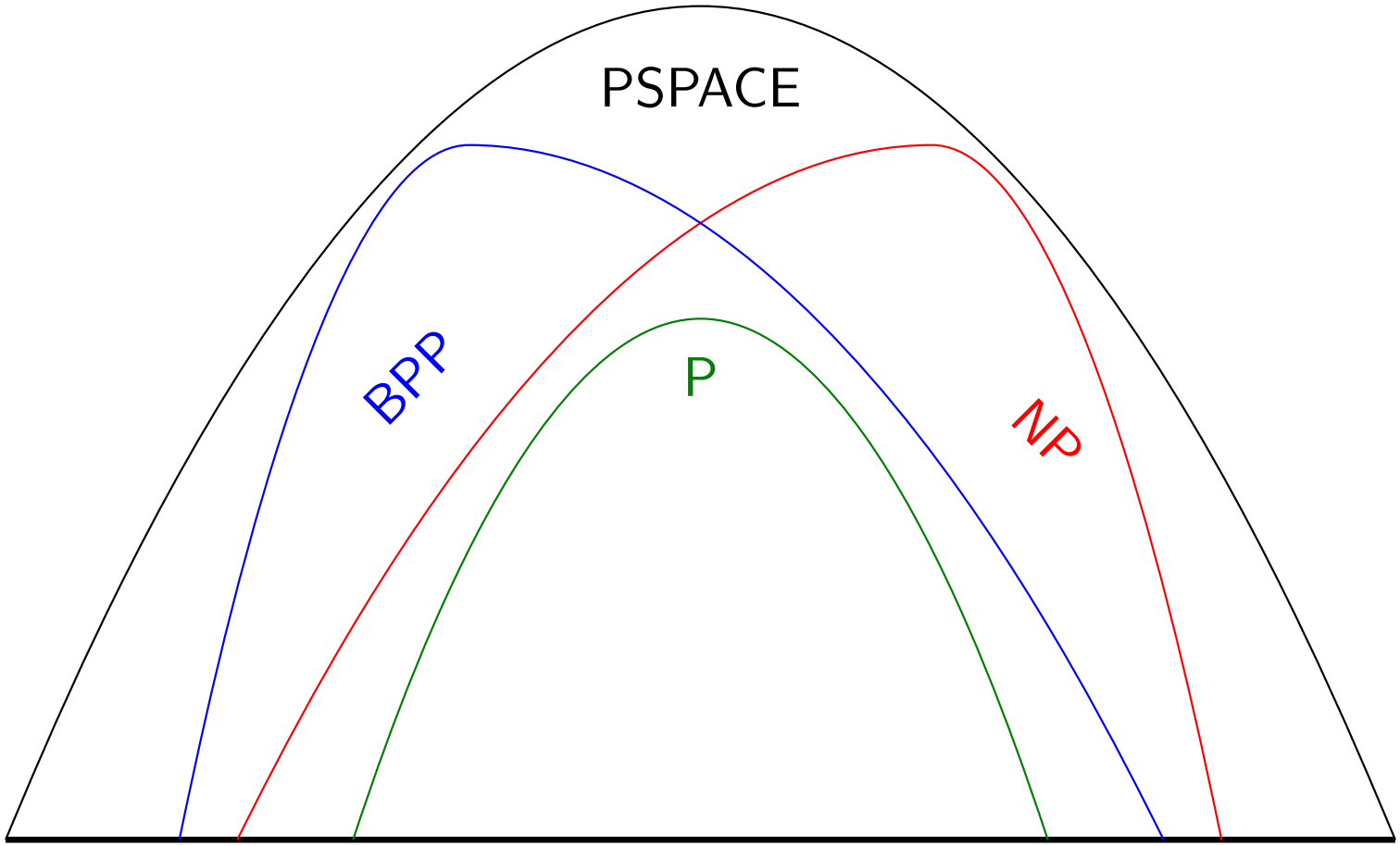
The class **PSPACE** (polynomial space) is the class of all languages $L(M)$ such that M is a polynomial space machine.

Example: QSAT (quantified satisfiability) is the set of all true quantified boolean formulas.

An example of such a formula is $\forall x_1 \exists x_2 : (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$.

QSAT belongs to **PSPACE**.

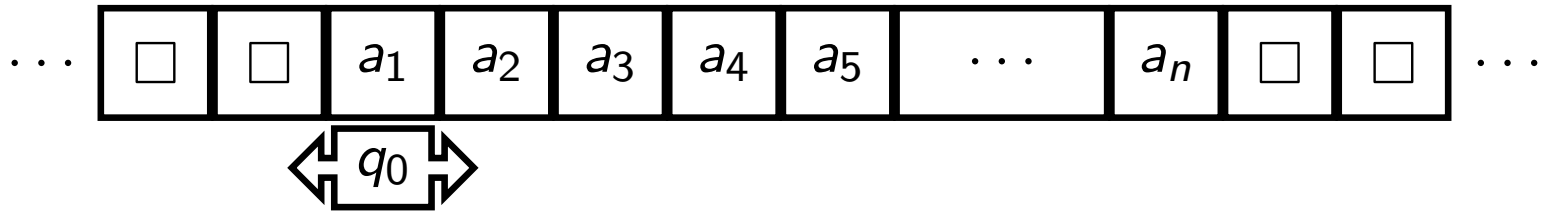
Part of the classical complexity world



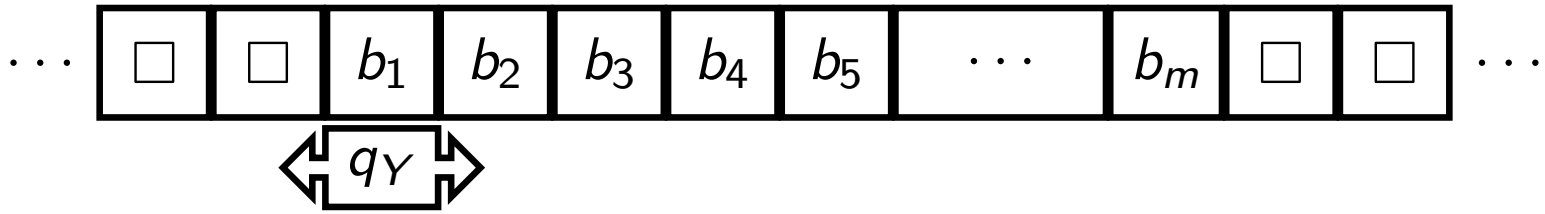
Computing functions with Turing machines

One may also use a Turing machine to M in order to compute an in general partially defined function $f_M : \{0,1\}^* \to \{0,1\}^*$:

$f_M(a_1 a_2 \cdots a_n) = b_1 b_2 \cdots b_m$ iff M reaches from the initial configuration



after a finite number of computations steps the configuration



Karp reductions and completeness

A Karp polynomial time reduction from a language $K \subseteq \{0,1\}^*$ to a language $L \subseteq \{0,1\}^*$ is a totally defined function $f : \{0,1\}^* \to \{0,1\}^*$ such that:

- $f = f_M$ for a PTM M (f can be computed in polynomial time),
- $\forall u \in \{0,1\}^* : u \in K$ if and only if $f(u) \in L$.

We write $K \leq L$ if there is a Karp polynomial time reduction from K to L.

Let **C** be a complexity class (e.g., **NP** or **PSPACE**).

We say that a language $L \subseteq \{0,1\}^*$ is **C**-complete if the following holds:

- $L \in \mathbf{C}$
- $\forall K \in \mathbf{C} : K \leq L$

Karp reductions and completeness

C-complete languages should be seen as the most difficult languages in **C**.

Examples:

- **SAT** is **NP**-complete.
- **QSAT** is **PSPACE**-complete.

Remarks:

- All non-trivial languages in **P** are **P**-complete.

  To get interesting **P**-complete problems, one has to replace Karp polynomial time reductions by Karp logspace reductions.

  Then the circuit value problem is **P**-complete.

- It is open whether **BPP** has complete problems!

Circuits instead of Turing machines

PTMs can be (almost) replaced by boolean circuits.

Let us consider a Boolean circuit $C(x_1, \ldots, x_n)$ with n input gates labelled with the variables $x_1, \ldots, x_n$.

Such a circuit naturally computes a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$.

Example:

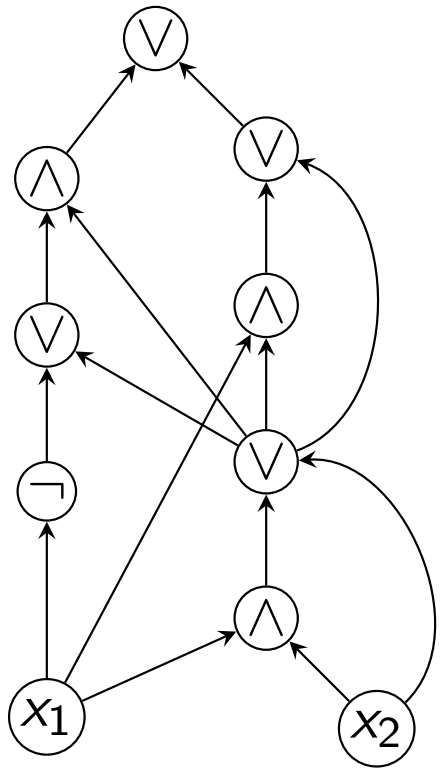
Circuits instead of Turing machines

PTMs can be (almost) replaced by boolean circuits.

Let us consider a Boolean circuit $C(x_1, \ldots, x_n)$ with n input gates labelled with the variables $x_1, \ldots, x_n$.

Such a circuit naturally computes a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$.

Example:



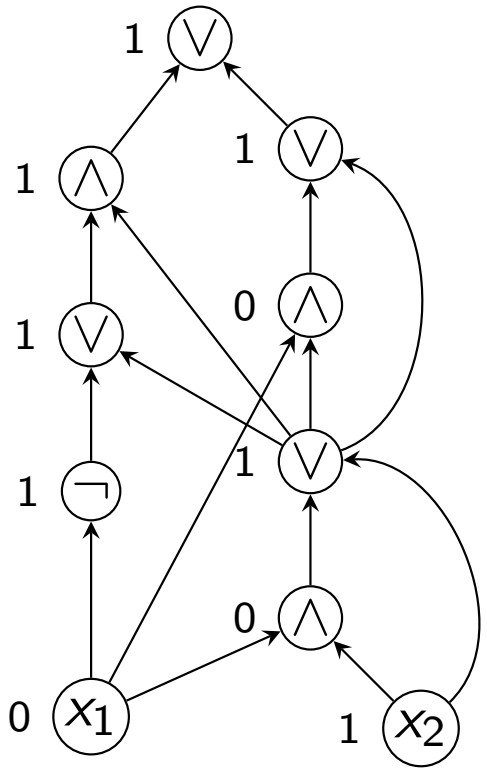
Circuits instead of Turing machines

PTMs can be (almost) replaced by boolean circuits.

Let us consider a Boolean circuit $C(x_1, \ldots, x_n)$ with n input gates labelled with the variables $x_1, \ldots, x_n$.

Such a circuit naturally computes a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$.

Example:



Circuits instead of Turing machines

Theorem 1

For every PTM M there is a PTM N that computes from the input string 1^n (n 1-bits) a description of a Boolean circuit $C_n(x_1, \ldots, x_n)$ such that:

$$\forall u = a_1 a_2 \cdots a_n \in \{0,1\}^n : u \in L(M) \iff f_{C_n}(a_1, a_2, \ldots, a_n) = 1.$$

Intuitively: The machine N builds for a given input length n the hardware needed to simulate the machine M.

Remarks:

- The family of circuits $(C_n)_{n \geq 0}$ from the above theorem is a called a **P**-uniform circuit family.
- The resources needed to compute the function $n \mapsto C_n$ are actually much smaller than polynomial time.

  Logarithmic space (on a deterministic Turing machine) suffices.

Promise problems

It was mentioned that it is not known whether the class **BPP** has complete problems (slide 19).

To get complete problems for **BPP**, we have to consider promise problems.

A promise problem is a pair (L_0, L_1) such that $L_0, L_1 \in \{0,1\}^*$ and $L_0 \cap L_1 = \emptyset$.

The class **promiseBPP** consists of all promise problems (L_0, L_1) for which there is a PTM M and a polynomial $r(n)$ such that the following hold for every $u \in \{0,1\}^n$:

- If $u \in L_1$ then $\langle u, v \rangle \in L(M)$ for at least $\frac{2}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.
- If $u \in L_0$ then $\langle u, v \rangle \in L(M)$ for at most $\frac{1}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.

Promise problems

A Karp polynomial time reduction f from the promise problem (K_0, K_1) to the promise problem (L_0, L_1) is a totally defined $f : \{0,1\}^* \to \{0,1\}^*$ such that:

- $f = f_M$ for a PTM M (f can be computed in polynomial time),
- $\forall u \in K_0 : f(u) \in L_0$.
- $\forall u \in K_1 : f(u) \in L_1$.

The following promise problem (L_0, L_1) is then complete for **promiseBPP**:

- L_0 is the set of all binary encodings of boolean circuits $C(x_1, \ldots, x_n)$ with $f_C(a_1, a_2, \ldots, a_n) = 0$ for $\geq \frac{2}{3} \cdot 2^n$ many $a_1 a_2 \cdots a_n \in \{0,1\}^n$.
- L_1 is the set of all binary encodings of boolean circuits $C(x_1, \ldots, x_n)$ with $f_C(a_1, a_2, \ldots, a_n) = 1$ for $\geq \frac{2}{3} \cdot 2^n$ many $a_1 a_2 \cdots a_n \in \{0,1\}^n$.

Complex number

We assume familiarity with the complex numbers $\mathbb{C}$.

Two ways of describing complex numbers:

- $x + iy$ for $x, y \in \mathbb{R}$ (and i satisfying $i^2 = -1$)
- $r \cdot e^{i\varphi}$ for $r \in \mathbb{R}_{\geq 0}$ and $\varphi \in [0, 2\pi)$

If $x + iy = z = r \cdot e^{i\varphi}$ then we have

- $r = \sqrt{x^2 + y^2} =: |z|$ and $\varphi = \arctan(y/x)$
  ($\varphi = \pi/2$ if $x = 0$ and $y > 0$ and $\varphi = 3\pi/2$ if $x = 0$ and $y < 0$)
- $x = r \cos\varphi$ and $y = r \sin\varphi$

The complex conjugate of $z = x + iy$ is $z^* = x - iy$.

Note $z \cdot z^* = x^2 + y^2 = |z|^2$.

Complex vector spaces

Quantum computing uses finite dimensional vector spaces over $\mathbb{C}$.

We use Dirac's bra-ket notation:

- Column vectors are denoted with $|x\rangle$.
- Row vectors are denoted with $\langle y|$.

Moreover, for a ket-vector

$$|x\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_d \end{pmatrix} \in \mathbb{C}^d$$

its conjugated transposed bra-vector is

$$\langle x| = (\alpha_1^*,\ \alpha_2^*,\ \ldots,\ \alpha_d^*).$$

Inner product and norm

The inner product of

$$|x\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_d \end{pmatrix} \quad \text{and} \quad |y\rangle = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \end{pmatrix}$$

is the complex number $\langle x|y\rangle = \sum_{i=1}^{d} \alpha_i^* \beta_i$.

This is a special case of a matrix product (row times column).

The norm of $|x\rangle$ is $\||x\rangle\| = \sqrt{\langle x|x\rangle} = \sqrt{\sum_{i=1}^{d} |\alpha_i|^2} \in \mathbb{R}_{\geq 0}$.

A unit vector is a vector $|x\rangle$ with $\|x\| = 1$.

For better readability we write $\|x\|$ for $\||x\rangle\|$ in the following.

Inner product and norm

The vector space $\mathbb{C}^d$ together with the inner product $\langle \cdot | \cdot \rangle$ is a so-called (finite dimensional) Hilbert space.

In general, a Hilbert space may have infinite dimension, but we will only consider finite dimensional Hilbert spaces.

Note that $\langle \cdot | \cdot \rangle$ satisfies the following laws, where $\alpha \in \mathbb{C}$:

- $\langle x_1 + x_2 | y \rangle = \langle x_1 | y \rangle + \langle x_2 | y \rangle$ (here, we write $\langle x_1 + x_2 |$ for $\langle x_1 | + \langle x_2 |$)
- $\langle x | y_1 + y_2 \rangle = \langle x | y_1 \rangle + \langle x | y_2 \rangle$
- $\langle \alpha x | y \rangle = \alpha^* \langle x | y \rangle$
- $\langle x | \alpha y \rangle = \alpha \langle x | y \rangle$
- $\langle y | x \rangle = \langle x | y \rangle^*$

Cauchy-Schwarz

Cauchy–Schwarz inequality

For all $|x\rangle, |y\rangle \in \mathbb{C}^d$ we have

$$\langle x|y\rangle \cdot \langle y|x\rangle = |\langle x|y\rangle|^2 \leq \langle x|x\rangle \cdot \langle y|y\rangle$$

with equality if and only if $|x\rangle$ and $|y\rangle$ are linearly dependent (i.e., $\alpha|x\rangle + \beta|y\rangle = 0$ where $\alpha, \beta \in \mathbb{C}$ and $\alpha \neq 0$ or $\beta \neq 0$).

By taking the square root on both sides of the Cauchy–Schwarz inequality, one gets

$$|\langle x|y\rangle| \leq \|x\| \cdot \|y\| .$$

Orthonormal base

A base of $\mathbb{C}^d$ is a set of (ket-)vectors $\{|x_1\rangle, \ldots, |x_d\rangle\}$ such that for every $|x\rangle$ there exist unique $\alpha_1, \ldots, \alpha_d \in \mathbb{C}$ with $|x\rangle = \sum_{i=1}^{d} \alpha_i |x_i\rangle$.

Note: every base of $\mathbb{C}^d$ consists of exactly d non-zero vectors (the dimension of the vector space).

An orthonormal base of $\mathbb{C}^d$ is a base $\{|x_1\rangle, \ldots, |x_d\rangle\}$ such that

$$\langle x_i | x_j \rangle = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

For every $|y\rangle \in \mathbb{C}^d$ we have $|y\rangle = \sum_{i=1}^{d} \langle x_i | y \rangle \cdot |x_i\rangle$

Exercise: If $|x\rangle$ is a unit vector and $\{|x_1\rangle, \ldots, |x_d\rangle\}$ an orthonormal base then $|x\rangle = \sum_{i=1}^{d} \alpha_i |x_i\rangle$ for unique $\alpha_1, \ldots, \alpha_d \in \mathbb{C}$ with $\sum_{i=1}^{d} |\alpha_i|^2 = 1$.

Orthonormal base

A particular orthonormal base is the standard base consisting of vectors

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad |1\rangle := \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \quad \ldots, \quad |d-1\rangle := \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

In quantum computing we will work in a vector space $\mathbb{C}^{2^n}$ of dimension 2^n and the standard base $\{|u\rangle : u \in \{0,1\}^n\}$.

Here, the bit strings in $\{0,1\}^n$ are identified with the numbers $0, \ldots, 2^n - 1$, e.g. $0 \mathrel{\widehat{=}} 00,\ 1 \mathrel{\widehat{=}} 01,\ 2 \mathrel{\widehat{=}} 10,\ 3 \mathrel{\widehat{=}} 11$ for $n = 2$.

The base $\{|u\rangle : u \in \{0,1\}^n\}$ is also called the computational base; its elements can be identified with the possible values of an n-bit register.

Linear mappings

A mapping $f : \mathbb{C}^d \to \mathbb{C}^d$ is linear if for all $|x\rangle, |y\rangle \in \mathbb{C}^d$ and all $\alpha \in \mathbb{C}$:

- $f(|x\rangle + |y\rangle) = f|x\rangle + f|y\rangle$ (we write $f|x\rangle$ for $f(|x\rangle)$)
- $f(\alpha\,|x\rangle) = \alpha f|x\rangle$

After fixing a base $\{|x_1\rangle, \ldots, |x_d\rangle\}$, one can identify the linear mapping f with the $(d \times d)$-matrix $A = (A_{i,j})_{1 \le i,j \le d}$, where

$$f|x_j\rangle = \sum_{i=1}^{d} A_{i,j}\,|x_i\rangle\,.$$

We then have: if $|x\rangle = \sum_{i=1}^{d} \alpha_i\,|x_i\rangle$ and $f|x\rangle = \sum_{i=1}^{d} \beta_i\,|x_i\rangle$ then

$$A \cdot \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_d \end{pmatrix}.$$

Note: If $\{|x_1\rangle, \ldots, |x_d\rangle\}$ is orthonormal then $A_{i,j} = \langle x_i|\, f|x_j\rangle$.

Changing the basis

Let f be linear mapping and let $\{|x_1\rangle, \ldots, |x_d\rangle\}$ and $\{|y_1\rangle, \ldots, |y_d\rangle\}$ be two bases of $\mathbb{C}^d$.

Let A (resp., B) be the matrix for f in the basis $\{|x_1\rangle, \ldots, |x_d\rangle\}$ (resp., $\{|y_1\rangle, \ldots, |y_d\rangle\}$).

Then there is an invertible matrix $C \in \mathbb{C}^{d\times d}$ such that $B = C^{-1}AC$.

Exercise: Find the matrix C explicitly.

Two matrices $A, B \in \mathbb{C}^{d\times d}$ are similar if there is an invertible matrix $C \in \mathbb{C}^{d\times d}$ such that $B = C^{-1}AC$.

Operations for matrices

Composition of linear mappings corresponds to matrix multiplication:

$$(AB)_{i,j} = \sum_{k=1}^{d} A_{i,k} B_{k,j}.$$

Recall: $AB \neq BA$ in general!

Transposed matrix: $(A^{\mathsf{T}})_{i,j} = A_{j,i}$

Conjugated matrix: $(A^*)_{i,j} = A^*_{i,j}$

Adjoint matrix: $A^\dagger = (A^*)^{\mathsf{T}} = (A^{\mathsf{T}})^*$

We have: $(A^\dagger)^\dagger = A$, $(A+B)^\dagger = A^\dagger + B^\dagger$, $(AB)^\dagger = B^\dagger A^\dagger$ and $(A^{-1})^\dagger = (A^\dagger)^{-1}$ for A invertible.

The operators $\mathsf{T}, *, \dagger$ can be defined also for rectangular matrices, in particular for bra- or ket-vectors. Note that $|x\rangle^\dagger = \langle x|$.

The outer product of vectors

For a ket-vector $|x\rangle$ and a bra-vector $\langle y|$ (both of dimension d) we can form their outer product $|x\rangle\langle y| \in \mathbb{C}^{d\times d}$.

It is a special case of a rectangular matrix product.

More specifically: if

$$|x\rangle = \begin{pmatrix} a_1 \\ \vdots \\ a_d \end{pmatrix} \quad \text{and} \quad \langle y| = (b_1, \ldots, b_d)$$

then $|x\rangle\langle y| = (a_i b_j)_{1\leq i,j\leq d}$.

Note that for every matrix $A = (A_{i,j})_{0\leq i,j\leq d-1}$ we have $A = \sum_{i,j} A_{i,j} |i\rangle\langle j|$ for the standard base $|0\rangle, \ldots, |d-1\rangle$.

Trace of a matrix

Trace of a matrix: $\mathrm{tr}(A) = \sum_{i=1}^{d} A_{i,i}$ (sum of the diagonal entries)

Important facts of the trace:

- $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$
- $\mathrm{tr}(\alpha A) = \alpha \cdot \mathrm{tr}(A)$
- $\mathrm{tr}(AB) = \mathrm{tr}(BA)$

Exercise: Prove this.

Warning: $\mathrm{tr}(AB) \neq \mathrm{tr}(A)\mathrm{tr}(B)$ in general!

$\mathrm{tr}(AB) = \mathrm{tr}(BA)$ implies $\mathrm{tr}(C^{-1}AC) = \mathrm{tr}(A)$ for an invertible matrix C.

Consequence: If A and B are similar then $\mathrm{tr}(A) = \mathrm{tr}(B)$.

Hence, the trace of a matrix is invariant under a basis change.

Special matrices

A matrix A is

- normal if $AA^\dagger = A^\dagger A$,
- unitary if $A^\dagger = A^{-1}$.
- Hermitian (or self-adjoint) if $A = A^\dagger$,
- positive semi-definite if A is Hermitian and $\langle x| A |x\rangle \geq 0$ for every $|x\rangle$.
- positive definite if A is Hermitian and $\langle x| A |x\rangle > 0$ for every $|x\rangle$.
- a projector if A is Hermitian and $A^2 = A$,

Note that Hermitian and unitary matrices are also normal.

If A and B describe the same linear function in two different orthonormal bases then there is a unitary matrix U with $B = U^{-1}AU = U^\dagger AU$.

Projectors onto subspaces

Let $S \leq \mathbb{C}^d$ be a subspace of $\mathbb{C}^d$ and let $\{|x_1\rangle, \ldots, |x_k\rangle\}$ be an orthonormal basis of S. The projector onto the subspace S is

$$\Pi_S = \sum_{i=1}^{k} |x_i\rangle\langle x_i| .$$

We have $\Pi_S^\dagger = \Pi_S$ and

$$\begin{aligned}
\Pi_S^2 &= \left(\sum_{i=1}^{k} |x_i\rangle\langle x_i| \right) \left(\sum_{j=1}^{k} |x_j\rangle\langle x_j| \right) \\
&= \sum_{i,j} |x_i\rangle\langle x_i|\, |x_j\rangle\langle x_j| = \sum_{i,j} |x_i\rangle\langle x_i|x_j\rangle\langle x_j| \\
&= \sum_{i=1}^{k} |x_i\rangle\langle x_i| = \Pi_S .
\end{aligned}$$

Projectors onto subspaces

Moreover, for every $|x\rangle \in \mathbb{C}^d$ we have

$$\Pi_S |x\rangle = \sum_{i=1}^{k} |x_i\rangle \langle x_i|x\rangle = \sum_{i=1}^{k} \langle x_i|x\rangle |x_i\rangle \in S$$

If $|x\rangle \in S$ then we can write $|x\rangle = \sum_{j=1}^{k} \alpha_j |x_j\rangle$ and we get

$$\begin{aligned} \Pi_S |x\rangle &= \left(\sum_{i=1}^{k} |x_i\rangle \langle x_i| \right) \left(\sum_{j=1}^{k} \alpha_j |x_j\rangle \right) \\ &= \sum_{i,j} \alpha_j |x_i\rangle \langle x_i|x_j\rangle = \sum_{j=1}^{k} \alpha_j |x_j\rangle = |x\rangle . \end{aligned}$$

Exercise: If Π is any projector ($\Pi^2 = \Pi$) find a subspace S with $\Pi = \Pi_S$.

The Pauli matrices

The matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

are the famous Pauli matrices.

They are unitary as well as Hermitian.

In particular, they satisfy: $X^2 = Y^2 = Z^2 = \mathrm{Id}_2$.

Eigenvectors and eigenvalues

For a $(d \times d)$-matrix A, a vector $|x\rangle \in \mathbb{C}^d$ is called an eigenvector of A if there is $\lambda \in \mathbb{C}$ with $A|x\rangle = \lambda |x\rangle$.

λ is the eigenvalue of A for $|x\rangle$ and $|x\rangle$ is an eigenvector for λ.

Let λ be an eigenvalue of A. Then the set of all eigenvectors for λ form a subspace of $\mathbb{C}^d$: if $A|x\rangle = \lambda |x\rangle$ and $A|y\rangle = \lambda |y\rangle$ then:

$$A(|x\rangle + |y\rangle) = \lambda(|x\rangle + |y\rangle) \quad \text{and} \quad A(\alpha |x\rangle) = \lambda\alpha |x\rangle$$

This subspace is called the eigenspace of λ and its dimension is the geometric multiplicity of λ.

The eigenspaces for the different eigenvalues of A are linearly independent.

Similar matrices have the same eigenvalues with the same geometric multiplicities (but the eigenvectors change).

The spectral theorem

Theorem 2 (spectral theorem, see e.g. Nielsen Chuang, page 72)

Let $A \in \mathbb{C}^{d\times d}$ be a matrix. Then there is an orthonormal basis of $\mathbb{C}^d$ consisting of eigenvectors of A if and only if A is normal.

Let A be a normal matrix and $\{|x_1\rangle, \ldots |x_d\rangle\}$ an orthonormal basis of eigenvectors of A. Let λ_i be the eigenvalue for $|x_i\rangle$. Then we have

$$A = \sum_{i=1}^{d} \lambda_i |x_i\rangle\langle x_i| .$$

In the orthonormal basis $\{|x_1\rangle, \ldots |x_d\rangle\}$, A becomes the diagonal matrix

$$\begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_d \end{pmatrix}$$

The spectral theorem

The diagonal entries are the eigenvalues of A.

Every eigenvalue appears its geometric multiplicity many times.

The spectral theorem says that the following two conditions are equivalent:

- A is normal.
- There is a unitary matrix U such that $U^\dagger AU$ is diagonal (A is unitarily diagonizable).

For a general (not necessarily normal) matrix $A \in \mathbb{C}^{d\times d}$ the following are equivalent:

- A is similar to a diagonal matrix.
- $\mathbb{C}^d$ has a basis (not necessarily orthonormal) consisting of eigenvectors of A.
- The sum of the geometric multiplicities of the eigenvalues of A is d.

Eigenvalues of special matrices

Theorem 3

The following are equivalent for a matrix U:

- U is unitary.
- The set of columns of U form an orthonormal basis.
- All eigenvalues λ of U satisfy $|\lambda| = 1$ (i.e. $\lambda = e^{i\phi}$ for some ϕ).
- U preserves the inner product. In formulas:
  $\langle x| U^\dagger U |y\rangle = \langle x|y\rangle$ (note that $(U|y\rangle)^\dagger = |y\rangle^\dagger U^\dagger = \langle y| U^\dagger$).

Theorem 4

The following are equivalent for a matrix H:

- H is Hermitian.
- All eigenvalues of H are real numbers.

Eigenvectors and eigenvalues

Theorem 5

The following are equivalent for a matrix P:

- P is positive definite (positive semi-definite).
- All eigenvalues of P are real and > 0 (≥ 0).

Exercise: For every matrix $A \in \mathbb{C}^{d \times d}$ the matrix $A^\dagger A$ is positive semi-definite.

Theorem 6

The following are equivalent for a matrix Π:

- Π is a projector
- All eigenvalues λ of Π are 0 or 1.

Tensor product of vector spaces

Theorem 7

Let U and V be vector spaces. Then there is up to isomorphism of vector spaces a unique vector space $U \otimes V$ with the following properties:

- There is a bilinear mapping $f : U \times V \to U \otimes V$.
- For every bilinear mapping $g : U \times V \to W$, where W is a vector space, there is a unique linear mapping $h : U \otimes V \to W$ such that $g(|x\rangle, |y\rangle) = h(f(|x\rangle, |y\rangle))$ for all $|x\rangle \in U$ and $|y\rangle \in V$.

The vector space $U \otimes V$ is the tensor product of U and V.

In the following we write $|x\rangle \otimes |y\rangle$ for $f(|x\rangle, |y\rangle)$.

Assume moreover in the following that $U \cong \mathbb{C}^d$ and $U \cong \mathbb{C}^e$ are finite-dimensional (the only interesting case).

How can we construct (a vector space isomorphic to) $U \otimes V$?

Tensor product of vector spaces

Fix standard bases

- $\{|0\rangle, \ldots, |d-1\rangle\}$ for U and
- $\{|0\rangle, \ldots, |e-1\rangle\}$ for V.

Then $\mathbb{C}^d \otimes \mathbb{C}^e$ is the vector space with the standard

- $\{|i\,j\rangle : 0 \leq i \leq d-1, 0 \leq i \leq e-1\}$

(in particular, $\mathbb{C}^d \otimes \mathbb{C}^e \cong \mathbb{C}^{de}$) and the corresponding mapping $f : \mathbb{C}^d \times \mathbb{C}^e \to \mathbb{C}^d \otimes \mathbb{C}^e$ is uniquely defined by

$$f(|i\rangle, |j\rangle) = |i\rangle \otimes |j\rangle = |i\,j\rangle .$$

Bilinearity of $\otimes$ then implies that for $|x\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$ and $|y\rangle = \sum_{j=0}^{e-1} \alpha_j |j\rangle$ we have

$$|x\rangle \otimes |y\rangle = \sum_{i,j} \alpha_i \beta_j |i\,j\rangle . \tag{1}$$

Tensor product of vector spaces

Remark: $\otimes$ is not surjective (this will be shown on Slide 57).

If $g : U \to U$ and $h : V \to V$ are linear mappings then we can define a linear mapping

$$g \otimes h : U \otimes V \to U \otimes V$$

as follows for all $0 \leq i \leq d-1$, $0 \leq i \leq e-1$:

$$(g \otimes h)(|i\,j\rangle) = g(|i\rangle) \otimes h(|j\rangle)$$

If A (B) is the matrix for g (h) in the standard basis of $\mathbb{C}^d$ ($\mathbb{C}^e$), then the matrix for $g \otimes h$ in the standard basis of $\mathbb{C}^{de}$ is the Kronecker product of A and B, often also called the tensor product of A and B.

We define the Kronecker product for rectangular matrices on the next slide.

Tensor product of matrices

Let $A = (a_{i,j})_{1 \leq i \leq k, 1 \leq j \leq \ell}$ be an $(k \times \ell)$-matrix and $B = (b_{i,j})_{1 \leq i \leq m, 1 \leq j \leq n}$ be an $(m \times n)$-matrix.

Their tensor product (or Kronecker product) $A \otimes B$ is the following $(km \times \ell n)$-matrix:

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \dots & a_{1,\ell}B \\ a_{2,1}B & a_{2,2}B & \dots & a_{2,\ell}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{k,1}B & a_{k,2}B & \dots & a_{k,\ell}B \end{pmatrix}$$

Note: If $|x\rangle \in \mathbb{C}^d$ and $|y\rangle \in \mathbb{C}^e$ then $|x\rangle \otimes |y\rangle \in \mathbb{C}^{de}$ is as in (1).

If $|i\rangle$ and $|j\rangle$ are vectors from the standard bases ($0 \leq i \leq k-1$ and $0 \leq j \leq \ell - 1$) then we also write $|i\,j\rangle$ for the tensor product $|i\rangle \otimes |j\rangle$.

Tensor product of matrices

The tensor product satisfies the following laws:

- $(A + B) \otimes C = A \otimes C + B \otimes C$ and $A \otimes (B + C) = A \otimes B + A \otimes C$
- $(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$.
- $(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$, where
  A is $(k \times \ell)$, B is $(m \times n)$, C is $(\ell \times p)$, D is $(n \times q)$.
- $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$.

A special case of the 3rd law is the following, where $|x\rangle, |x'\rangle \in \mathbb{C}^k$, $|y\rangle, |y'\rangle \in \mathbb{C}^\ell$.

$$(\langle x| \otimes \langle y|)(|x'\rangle \otimes |y'\rangle) = \langle x|x'\rangle \langle y|y'\rangle$$

In particular $|x\rangle = |x'\rangle$ and $|y\rangle = |y'\rangle$ yields $\||x\rangle \otimes |y\rangle\| = \|x\| \cdot \|y\|$.

Tensor product of matrices

Note: in general we have $A \otimes B \neq B \otimes A$

Exercise: Show the following:

$$A \text{ and } B \text{ are } \left\{ \begin{array}{l} \text{unitary} \\ \text{Hermitian} \\ \text{positive definite} \\ \text{projectors} \end{array} \right\} \implies A \otimes B \text{ is } \left\{ \begin{array}{l} \text{unitary} \\ \text{Hermitian} \\ \text{positive definite} \\ \text{a projector} \end{array} \right\}$$

Principles of quantum computing: the state

Classical computing: at each time instant the system is in one of say d states $0, 1, \ldots d-1$.

Quantum computing: a (quantum) state is a unit vector $|x\rangle \in \mathbb{C}^d$:

$$|x\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle ,$$

where $\alpha_i \in \mathbb{C}$ (the amplitude of $|i\rangle$) and $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$.

Later, when we introduce quantum circuits, the standard basis

$$\{|0\rangle , \ldots , |d-1\rangle\}$$

will be replaced by the computational basis

$$\{|u\rangle : u \in \{0,1\}^n\}$$

of an n-bit quantum register.

Principles of quantum computing: the dynamics

Classical computing: the system evolves according to some (possibly time-dependant) function $f : \{0, 1, \ldots, d-1\} \to \{0, 1, \ldots, d-1\}$ on the set of states.

Quantum computing: systems evolve according to unitary transformations.

If at time t the system is in state $|x\rangle \in \mathbb{C}^d$ (a unit vector) then at time $t+1$ the system is in state $U(t)|x\rangle$ for some unitary matrix $U(t)$ (that may depend on time).

Note: $U(t)|x\rangle$ is again a unit vector since unitary matrices preserve the norm.

Principles of quantum computing: measurements

Classical computing: In principle, the observer of a classical system knows at each time instant the current state $s \in \{0, 1, \ldots, d-1\}$.

Quantum computing: The current state $|x\rangle \in \mathbb{C}^d$ is hidden for an observer (in particular, one cannot determine the amplitudes).

All she/he can do is a measurement.

The simplest possible measurement in our setting would be a full projective measurement in the standard base:

After measuring the current state $|x\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$ the system collapses to the basis states $|i\rangle$ (the new state of the system) with probability $|\alpha_i|^2$.

Recall: $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$.

The observer gets the knowledge to which basis state $|i\rangle$ the quantum state collapses.

Principles of quantum computing: measurements

More generally, a projective measurement is given by a collection of projectors $\{\Pi_1, \ldots, \Pi_k\}$ ($\Pi_i^2 = \Pi_i$ and $\Pi_i^\dagger = \Pi_i$) such that

- $\Pi_i \Pi_j = 0$ (the zero matrix) for $i \neq j$ and
- $\sum_{i=1}^k \Pi_i = \mathrm{Id}_d$ (the $(d \times d)$ identity matrix).

Applying this projective measurement to the state $|x\rangle$ results with probability $\|\Pi_i |x\rangle\|^2 = \langle x| \Pi_i |x\rangle$ in the post-measurement state

$$\frac{\Pi_i |x\rangle}{\|\Pi_i |x\rangle\|}.$$

This is a unit vector from the subspace $S_i = \{\Pi_i |x\rangle : |x\rangle \in \mathbb{C}^d\}$ onto which Π_i projects. The observer gets the knowledge of i.

Remark: If $i \neq j$ and $|x\rangle, |y\rangle \in \mathbb{C}^d$ then $\langle x| \Pi_i^\dagger \Pi_j |y\rangle = \langle x| \Pi_i \Pi_j |y\rangle = 0$. Thus, S_i and S_j are orthogonal for $i \neq j$.

Principles of quantum computing: measurements

Remark: $\mathbb{C}^d = S_1 \oplus S_2 \oplus \cdots \oplus S_k$, since $|x\rangle = \sum_{i=1}^{k} \Pi_i |x\rangle$ for all $|x\rangle \in \mathbb{C}^d$.

In a full projective measurement in the standard base, the S_i are the vector spaces spanned by the basis vectors $|i\rangle$ (and we have $k = d$).

A Hermitian matrix H yields a projective measurement as follows:

- Let $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ be the different eigenvalues of H and let S_i be the eigenspace of λ_i.
- Then $\{\Pi_{S_1}, \ldots, \Pi_{S_k}\}$ is a projective measurement!

There is an even more general notion of measurement in quantum information theory: POVM – positive operator-valued measurement.

We won't need it in this lecture.

Principles of quantum computing: combining systems

Classical computing: If we have two systems with state spaces $S = \{x_1, \ldots, x_d\}$ and $T = \{y_1, \ldots, y_e\}$, then the two systems can be viewed as a single system with state space $S \times T$.

Quantum computing: Two quantum systems with standard bases

- $\{|0\rangle, \ldots, |d-1\rangle\}$ (yielding space $\mathbb{C}^d$) and
- $\{|0\rangle, \ldots, |e-1\rangle\}$ (yielding space $\mathbb{C}^e$)

can be combined into a single system with standard base

- $\{|i\,j\rangle : 0 \leq i < d, 0 \leq j < e\}$ yielding space $\mathbb{C}^d \otimes \mathbb{C}^e \cong \mathbb{C}^{de}$.

If the two systems are in states $|x\rangle = \sum_{i=0}^{d-1} \alpha_i |i\rangle$ and $|y\rangle = \sum_{j=0}^{e-1} \beta_j |j\rangle$ then the combined system is in state

$$|x\rangle \otimes |y\rangle = \sum_{i,j} \alpha_i \beta_j |i\rangle \otimes |j\rangle = \sum_{i,j} \alpha_i \beta_j |i\,j\rangle .$$

Principles of quantum computing: combining systems

Important remark: Not every state of $\mathbb{C}^d \otimes \mathbb{C}^e$ can be written as $|x\rangle \otimes |y\rangle$ for states $|x\rangle \in \mathbb{C}^d$ and $|y\rangle \in \mathbb{C}^e$.

A state $|z\rangle \in \mathbb{C}^d \otimes \mathbb{C}^e$ is called entangled if it is not of the form $|x\rangle \otimes |y\rangle$ for states $|x\rangle \in \mathbb{C}^d$ and $|y\rangle \in \mathbb{C}^e$.

Example: The Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled.
To see this, assume that

$$\begin{aligned} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) \\ &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle . \end{aligned}$$

This implies $\alpha\gamma = \beta\delta = \frac{1}{\sqrt{2}}$ and $\alpha\delta = \beta\gamma = 0$, from which we get $\alpha\beta\gamma\delta = \frac{1}{2}$ and $\alpha\beta\gamma\delta = 0$, a contradiction.

Principles of quantum computing: combining systems

If two systems evolve according to the unitary transformations U and V then the combined system evolves according to the unitary $U \otimes V$.

This makes sense, since $(U \otimes V)(|x\rangle \otimes |y\rangle) = U|x\rangle \otimes V|y\rangle$.

Measuring the

- 1st system using $\{\Pi_1, \ldots, \Pi_k\}$ and independently the
- 2nd system using $\{\Phi_1, \ldots, \Phi_\ell\}$

is the same as measuring the combined system using

- $\{\Pi_i \otimes \Phi_j : 1 \leq i \leq k, 1 \leq j \leq \ell\}$.

This yields the intuitively correct probabilities!

Principles of quantum computing: combining systems

Assume that $|x\rangle$ and $|y\rangle$ are the states of the two systems and measure them independently from each other.

With $\pi_i = \|\Pi_i |x\rangle\|$ and $\rho_j = \|\Phi_j |y\rangle\|$ we get

- $\text{Prob}\big[\text{ post measurement state of system 1} = \frac{\Pi_i|x\rangle}{\pi_i}\big] = \pi_i^2$
- $\text{Prob}\big[\text{ post measurement state of system 2} = \frac{\Phi_j|y\rangle}{\rho_j}\big] = \rho_j^2$

Hence, with probability $\pi_i^2\rho_j^2$ the combined system is in state

$$\frac{\Pi_i |x\rangle}{\pi_i} \otimes \frac{\Phi_j |y\rangle}{\rho_j} = \frac{\Pi_i |x\rangle \otimes \Phi_j |y\rangle}{\pi_i\rho_j} = \frac{\Pi_i |x\rangle \otimes \Phi_j |y\rangle}{\|\Pi_i |x\rangle \otimes \Phi_j |y\rangle\|}$$

after the measurements.

With the combined measurement $\{\Pi_i \otimes \Phi_j : 1 \leq i \leq k, 1 \leq j \leq \ell\}$ the same result is obtained!

Global phase and the density matrix

Consider quantum states $|x\rangle, |y\rangle \in \mathbb{C}^d$ (so $\langle x|x\rangle = \langle y|y\rangle = 1$) and assume there is $\alpha \in \mathbb{C}$ such that $|x\rangle = \alpha |y\rangle$.

We must have $|\alpha| = 1$, i.e., $\alpha = e^{i\phi}$ for some $\phi \in [0, 2\pi)$.

α is called a global phase factor.

It has no physical meaning in the following sense: For every measurement $\{\Pi_1, \ldots, \Pi_k\}$ and every j we have $\langle x| \Pi_j |x\rangle = \langle y| \Pi_j |y\rangle$.

Hence, $|x\rangle$ and $|y\rangle$ cannot be distinguished by measurements.

Exercise: Show that: $|x\rangle = e^{i\phi} |y\rangle$ for some $\phi \iff |x\rangle\langle x| = |y\rangle\langle y|$.

The matrix $|x\rangle\langle x|$ is called the density matrix of the state $|x\rangle$.

Quantum bits (qubits)

Unit vectors of the form $\alpha\,|0\rangle + \beta\,|1\rangle \in \mathbb{C}^2$ are the states of a single qubit.

Physically, one could realize a qubit by a quantum mechanical system with two distinguished states (e.g. the spin of an electron can be $\uparrow$ and $\downarrow$).

An *n*-bit quantum register is the combination of n qubits. Its state space is

$$\underbrace{\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{n \text{ times}} \cong \mathbb{C}^{2^n}.$$

with the computational basis $\{|u\rangle : u \in \{0,1\}^n\}$.

Note: if $u = a_1 a_2 \cdots a_n$ then $|u\rangle = |a_1\rangle \otimes |a_2\rangle \otimes \cdots \otimes |a_n\rangle$.

In the following let us write $\mathcal{U}_d$ for the set of unitary $(d \times d)$-matrices.

Quantum circuits

A quantum circuit is the quantum analog of a classical Boolean circuit.

An n-bit quantum circuit operates on the state space of an n-bit quantum register.

The circuit consists of a sequence of unitary matrices $U_1, \ldots, U_m \in \mathcal{U}_{2^n}$ followed by a measurement.

Each unitary matrix U_i should be simple but also powerful enough in the same sense as a single Boolean gate is simple.

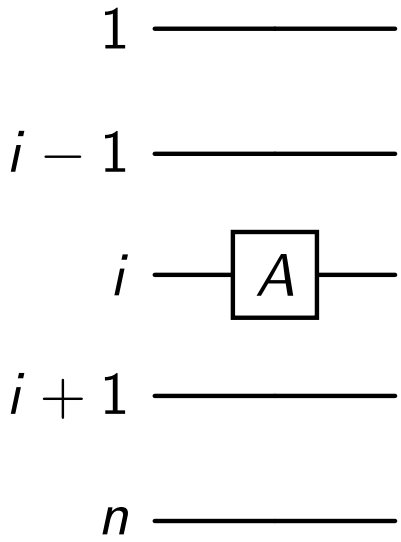
It is common to restrict the U_i to unitary transformations that operate locally on only one or two qubits. (analogous to the boolean gates $\neg$, $\vee$ and $\wedge$ that operate on one or two bits).

1-qubit quantum gates

1-qubit quantum gates: $\mathrm{Id}_{2^{i-1}} \otimes A \otimes \mathrm{Id}_{2^{n-i}}$ where $A \in \mathcal{U}_2$ and $1 \leq i \leq n$.

Intuition: A operates on qubit i and does not touch the j-th qubit for every $j \in \{1, \ldots, n\} \setminus \{i\}$.

In pictures:



CNOT gate

It turns out that only one type of two-bit quantum gates is needed: CNOT-gate (controlled not-gates).

$\mathrm{CNOT}_{i,j}$ for $1 \leq i, j \leq n$ with $i \neq j$ is the unique linear transformation that operates on the computational basis as follows, where $a_1, \ldots, a_n \in \{0, 1\}$

$$\mathrm{CNOT}_{i,j}(|a_1 \cdots a_n\rangle) = |a_1 \cdots a_{j-1}(a_j \oplus a_i)a_j \cdots a_n\rangle \,,$$

where $\oplus$ is the boolean XOR.

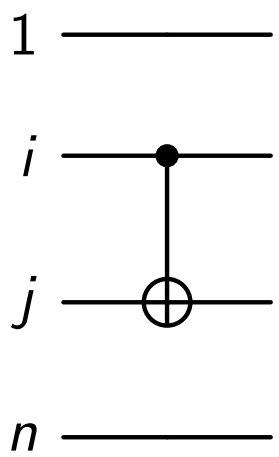
Intuition: Flip the j-th bit if the i-th bit is 1, otherwise do nothing.

Since $\mathrm{CNOT}_{i,j}$ permutes the computational basis, the columns (and rows) of $\mathrm{CNOT}_{i,j}$ form again the computational basis.

Therefore $\mathrm{CNOT}_{i,j}$ is indeed unitary.

CNOT gate

In pictures:



The matrix for $\mathrm{CNOT}_{1,2}$ is:

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

CNOT and classical copying

CNOT can be used to copy classical information: for all $a \in \{0,1\}$ we have:

$$\mathrm{CNOT}_{1,2}\,|a\,0\rangle = |a\,(0 \oplus a)\rangle = |a\,a\rangle$$

This does not contradict the following famous result of quantum computing:

No-cloning theorem

There is no unitary transformation U on $2n$ qubits such that for every n-bit quantum state $|x\rangle \in \mathbb{C}^{2^n}$ we have

$$U(|x\rangle \otimes |0^n\rangle) = |x\rangle \otimes |x\rangle\,. \tag{2}$$

The above comment on CNOT only implies that the identity (2) can be achieved if we restrict to computational basis states $|u\rangle$ for $u \in \{0,1\}^n$.

Proof of the no-cloning theorem

Assume there is a unitary transformation U of the form excluded in the no-cloning theorem.

Take two differet quantum states $|x\rangle, |y\rangle$ (on n qubits). We get:

$$\begin{aligned}
\langle x|y\rangle &= \langle x|y\rangle \cdot \langle 0^n|0^n\rangle \\
&= (\langle x| \otimes \langle 0^n|) \cdot (|y\rangle \otimes |0^n\rangle) \\
&= (\langle x| \otimes \langle 0^n|) U^\dagger U (|y\rangle \otimes |0^n\rangle) \\
&= (\langle x| \otimes \langle x|)(|y\rangle \otimes |y\rangle) \\
&= \langle x|y\rangle \cdot \langle x|y\rangle
\end{aligned}$$

The equation $a^2 = a$ has only two solutions in $\mathbb{C}$: 0 and 1.

Hence, we have $\langle x|y\rangle \in \{0, 1\}$.

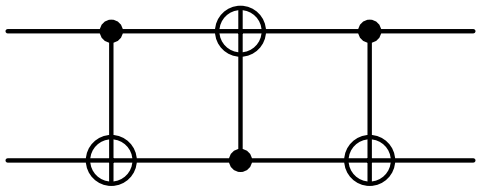
We get a contradiction if we choose x, y such that $0 \neq \langle x|y\rangle \neq 1$. □

CNOT and SWAP

CNOT can be used to swap two qubits, i.e., to compute a unitary operation SWAP (on 2 qubits) such that for all $a, b \in \{0, 1\}$:

$$\text{SWAP} \, |ab\rangle = |ba\rangle \, .$$

We need 3 CNOT-gates for this:



CNOT + 1-qubit quantum gates are universal

Theorem 8 (see Nielsen, Chuang, Section 4.5.2)

Every unitary transformation on n-qubits can be composed from at most $\mathcal{O}(n^2 4^n)$ 1-qubit quantum gates and CNOT-gates.

The bound $\mathcal{O}(n^2 4^n)$ cannot be significantly improved (see Nielsen, Chuang, Section 4.5.4)!

Problem: Using arbitrary 1-qubit quantum gates is not realistic in practice.

There are uncountably many 1-qubit quantum gates and we cannot expect to find a physical implementation of every 1-qubit quantum gate.

Our goal is to approximate arbitrary 1-qubit quantum gates with high precision using a fixed finite set of 1-qubit quantum gates.

Spectral norm

For a matrix $A \in \mathbb{C}^{d \times d}$ we define its spectral norm

$$\|A\| = \max\{\|A\,|x\rangle\| : |x\rangle \in \mathbb{C}^d, \|x\| = 1\}.$$

For $A, B \in \mathbb{C}^{d \times d}$ we define their distance as $d(A, B) = \|A - B\|$.

Some facts about the spectral norm, where $A, B \in \mathbb{C}^{d \times d}$, $\alpha \in \mathbb{C}$, and $U, V \in \mathcal{U}_d$:

- $\|A + B\| \leq \|A\| + \|B\|$ (triangle inequality)
- $\|\alpha A\| = |\alpha| \cdot \|A\|$
- $A = 0$ if and only if $\|A\| = 0$
- $\|AB\| \leq \|A\| \cdot \|B\|$ (submultiplicativity)
- $\|UAV\| = \|A\|$ (unitary invariance)

Spectral norm

Proof of unitary invariance: Let $U, V \in \mathcal{U}_d$.

The mapping $|x\rangle \mapsto V|x\rangle$ induces a bijection on the set of unit vectors.

Hence, we have:

$$\|AV\| = \max\{\|A(V|x\rangle)\| : |x\rangle \in \mathbb{C}^d, \|x\| = 1\} = \|A\|$$

Since $\|U|y\rangle\| = \|y\|$ for all vectors $|y\rangle$, we have:

$$\begin{aligned} \|UA\| &= \max\{\|UA|x\rangle\| : |x\rangle \in \mathbb{C}^d, \|x\| = 1\} \\ &= \max\{\|A|x\rangle\| : |x\rangle \in \mathbb{C}^d, \|x\| = 1\} \\ &= \|A\| \end{aligned}$$

Spectral norm and eigenvalues

Theorem 9

For every $A \in \mathbb{C}^{d\times d}$ we have $\|A\| = \sqrt{\rho}$, where ρ be the largest eigenvalue of $A^\dagger A$ (a positive semi-definite matrix).

Proof: Since $A^\dagger A$ is positive semi-definite, there is a unitary matrix U such that

$$D := U^\dagger A^\dagger A U = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix}$$

where the λ_i are the (non-negative real) eigenvalues of $A^\dagger A$.

Spectral norm and eigenvalues

We obtain the following, where w.l.o.g. $\lambda_d = \rho$ is the largest eigenvalue of the matrix $A^\dagger A$:

$$\begin{aligned}
\|A\|^2 &= \max\{\langle x| A^\dagger A |x\rangle : |x\rangle \in \mathbb{C}^d, \||x\rangle\| = 1\} \\
&= \max\{\langle x| U^\dagger A^\dagger A U |x\rangle : |x\rangle \in \mathbb{C}^d, \||x\rangle\| = 1\} \\
&= \max\{\langle x| D |x\rangle : |x\rangle \in \mathbb{C}^d, \||x\rangle\| = 1\} \\
&= \max\left\{\sum_{i=1}^d |\alpha_i|^2 \lambda_i : \alpha_1, \ldots, \alpha_d \in \mathbb{C}, \sum_{i=1}^d |\alpha_i|^2 = 1\right\} \\
&= \max\left\{\sum_{i=1}^d p_i \lambda_i : 0 \leq p_1, \ldots, p_d \leq 1, \sum_{i=1}^d p_i = 1\right\} \\
&= \lambda_d
\end{aligned}$$

Spectral norm and eigenvalues

Corollary 10

$\|U\| = 1$ for a unitary matrix U.

Corollary 11

$\|H\| = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } H\}$ for a Hermitian matrix H.

Corollary 12

$\|P\| = \max\{\lambda : \lambda \text{ is an eigenvalue of } P\}$ for a positive-semi-definite P.

Spectral norm and tensor products

Lemma 13

$\|A \otimes B\| = \|A\| \cdot \|B\|$

Proof: We have $(A \otimes B)^\dagger (A \otimes B) = (A^\dagger \otimes B^\dagger) \cdot (A \otimes B) = A^\dagger A \otimes B^\dagger B$.

The matrix $A^\dagger A \otimes B^\dagger B$ is positive semi-definite
(since $A^\dagger A$ and $B^\dagger B$ are positive semi-definite).

Choose orthonormal bases

- $|x_1\rangle, \ldots, |x_d\rangle$ consisting of eigenvectors of $A^\dagger A$ and
- $|y_1\rangle, \ldots, |y_d\rangle$ consisting of eigenvectors of $B^\dagger B$ and let
- λ_i be the eigenvalue of $A^\dagger A$ for $|x_i\rangle$, where $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d$,
- μ_i be the eigenvalue of $B^\dagger B$ for $|y_i\rangle$, where $0 \leq \mu_1 \leq \mu_2 \leq \cdots \leq \mu_d$.

Spectral norm and tensor products

Then $\{|x_i\rangle \otimes |y_j\rangle : 1 \leq i, j \leq d\}$ is an orthonormal basis of eigenvectors for $A^\dagger A \otimes B^\dagger B$.

The eigenvalue for $|x_i\rangle \otimes |y_j\rangle$ is $\lambda_i \lambda_j$.

These must be all eigenvalues! There is no more space for further eigenvalues.

Hence, the largest eigenvalue of $(A \otimes B)^\dagger (A \otimes B)$ is

$$\lambda_d \cdot \mu_d = \|A\|^2 \cdot \|B\|^2 .$$

We get $\|A \otimes B\| = \sqrt{\lambda_d \cdot \mu_d} = \|A\| \cdot \|B\|$. □

The Solovay-Kitaev Theorem

A finite subset $S \subseteq \mathcal{U}_d$ of d-dimensional unitary matrices is dense if for all $V \in \mathcal{U}_d$ and $\epsilon > 0$ there is product $U = U_1 U_2 \cdots U_k$ such that

- $U_1, U_2, \ldots, U_k \in S$ and
- $d(U, V) \leq \epsilon$.

Theorem 14 (Solovay 1995, Kitaev 1997)

Fix a dimension d and let $S \subseteq \mathcal{U}_d$ be finite and dense. Then for every $V \in \mathcal{U}_d$ and every $\epsilon > 0$ there is product $U = U_1 U_2 \cdots U_k$ such that

- $U_1, U_2, \ldots, U_k \in S$,
- $d(U, V) \leq \epsilon$ and
- $k \leq \mathcal{O}(\log^c(1/\epsilon))$ for some constant c.

The Solovay-Kitaev Theorem

Remarks:

- In the first proofs of the Solovay-Kitaev theorem, the constant c was $3 + \delta$, where $\delta > 0$ can be chosen arbitrarily small.
- Recently, Greg Kuperberg gave a new proof showing that one can take $c = 1.44042\cdots + \delta$ for $\delta > 0$ arbitrarily small.
- Given a good approximation of V, one can compute the sequence $U_1 U_2 \cdots U_k$ also efficiently.

We are mainly interested in the case $d = 2$ (unitary transformations on 1 qubit).

Fortunately, there is a finite and dense subset of $\mathcal{U}_2$.

A universal gate set

Theorem 15 (see Nielsen and Chuang, Section 4.5.3)

The set consisting of

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \text{ and } T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

is a dense subset of $\mathcal{U}_2$.

H is called the Hadamard matrix (or the Hadamard gate), T is called the $\pi/8$ gate.

Hence, by the Solovay-Kitaev Theorem every $U \in \mathcal{U}_2$ can be approximated using H and T up to error $\epsilon > 0$ using a sequence of length $\mathcal{O}(\log^c(1/\epsilon))$.

Getting the Pauli matrices from H and T

Recall the Pauli matrices X, Y, Z from Slide 39.

- $Z = T^4$
- $X = HZH = HT^4H$
- $iY = XZ = HT^4HT^4$

In the last line, the factor i is a physically irrelevant global factor.

Error approximation in quantum circuits

Assume we have an n-bit quantum circuit $U_1, U_2, \ldots, U_m$ (for the moment, we ignore the final measurement).

Every U_i is either a

- CNOT-gates or an
- arbitrary 1-qubit quantum gate $\mathrm{Id}_{2^{i-1}} \otimes A \otimes \mathrm{Id}_{2^{n-i}}$ where $A \in \mathcal{U}_2$.

Assume we have approximated (using H and T) every A in the 2nd point by a unitary $B \in \mathcal{U}_2$ with $d(A, B) \leq \epsilon$ (using Solovay-Kitaev).

We then have:

$$\begin{aligned} & d(\mathrm{Id}_{2^{i-1}} \otimes A \otimes \mathrm{Id}_{2^{n-i}}, \mathrm{Id}_{2^{i-1}} \otimes A' \otimes \mathrm{Id}_{2^{n-i}}) \\ = & \left\| \mathrm{Id}_{2^{i-1}} \otimes A \otimes \mathrm{Id}_{2^{n-i}} - \mathrm{Id}_{2^{i-1}} \otimes A' \otimes \mathrm{Id}_{2^{n-i}} \right\| \\ = & \left\| \mathrm{Id}_{2^{i-1}} \otimes (A - B) \otimes \mathrm{Id}_{2^{n-i}} \right\| \\ = & \left\| \mathrm{Id}_{2^{i-1}} \right\| \cdot \|A - B\| \cdot \left\| \mathrm{Id}_{2^{n-i}} \right\| \\ = & \|A - B\| = d(A, B) \leq \epsilon. \end{aligned}$$

Error approximation in quantum circuits

Lemma 16

Let $U_1, V_1, \ldots, U_m, V_m \in \mathcal{U}_d$ with $d(U_i, V_i) \leq \epsilon$ for all $1 \leq i \leq m$.
Then we have $d(U_1 U_2 \cdots U_m, V_1 V_2 \cdots V_m) \leq m\epsilon$.

Proof: Induction over m.

The case $m = 1$ is clear.

Assume now that $m > 1$.

Define $A = U_1 U_2 \cdots U_{m-1}$ and $V = V_1 V_2 \cdots V_{m-1}$.

The induction hypothesis tells us that $d(A, B) \leq (m-1)\epsilon$.

We obtain:

Error approximation in quantum circuits

$$
\begin{aligned}
d(AU_m, BV_m) &= \|AU_m - BV_m\| \\
&= \|AU_m - AV_m + AV_m - BV_m\| \\
&= \|A(U_m - V_m) + (A - B)V_m\| \\
&\leq \|A(U_m - V_m)\| + \|(A - B)V_m\| \\
&\leq \epsilon + (m-1)\epsilon \;=\; m\epsilon
\end{aligned}
$$

These are good news: If we want to approximate the quantum circuit up to precision δ, we have to approximate every 1-qubit quantum gate with an error bounded by δ/m.

By the Solovay-Kitaev theorem, this can be achieved by replacing every 1-qubit quantum gate by a sequence of H-gates and T-gates of length $\mathcal{O}(\log^c(m/\delta)) = \mathcal{O}((\log m - \log \delta)^c)$.

Measurements in quantum circuits

Lemma 17

Let $|x\rangle \in \mathcal{C}^d$ be a quantum state (a unit vector), $U, V \in \mathcal{U}_d$ such that $d(U,V) \leq \epsilon$ and Π a projector of dimension d.

Then we have: $|\langle x| U^\dagger \Pi U |x\rangle - \langle x| V^\dagger \Pi V |x\rangle| \leq 2\epsilon$.

Proof: With $|y\rangle = (U - V)|x\rangle$ we have

$$
\begin{aligned}
& |\langle x| U^\dagger \Pi U |x\rangle - \langle x| V^\dagger \Pi V |x\rangle| \\
= \; & |\langle x| U^\dagger \Pi U |x\rangle - \langle x| U^\dagger \Pi V |x\rangle + \langle x| U^\dagger \Pi V |x\rangle - \langle x| V^\dagger \Pi V |x\rangle| \\
= \; & |\langle x| U^\dagger \Pi |y\rangle + \langle y| \Pi V |x\rangle| \\
\leq \; & |\langle x| U^\dagger \Pi |y\rangle| + |\langle y| \Pi V |x\rangle| \\
\leq \; & \||y\rangle\| + \||y\rangle\| \quad \text{(here we apply Cauchy-Schwarz)} \\
\leq \; & 2d(U,V) \leq 2\epsilon.
\end{aligned}
$$

Measurements in quantum circuits

Similar to the unitary operations in a quantum circuit we should also restrict the allowed measurements.

Most of the time, we measure only a single qubit, say the first of the n qubits on which the quantum circuit works.

This means that the measurement is given by the projectors Π_0, Π_1, where

$$\Pi_0 = |0\rangle\langle 0| \otimes \mathrm{Id}_{n-1} \quad \text{and} \quad \Pi_1 = |1\rangle\langle 1| \otimes \mathrm{Id}_{n-1}.$$

Π_0 (Π_1) projects on the subspace spanned by all basis vectors $|0y\rangle$ ($|1y\rangle$) for $y \in \{0,1\}^{n-1}$.

Note that we have $\Pi_0\Pi_1 = \Pi_1\Pi_0 = 0$ and

$$\begin{aligned} \Pi_0 + \Pi_1 &= |0\rangle\langle 0| \otimes \mathrm{Id}_{n-1} + |1\rangle\langle 1| \otimes \mathrm{Id}_{n-1} \\ &= (|0\rangle\langle 0| + |1\rangle\langle 1|) \otimes \mathrm{Id}_{n-1} = \mathrm{Id}_n. \end{aligned}$$

Acceptance probability of a quantum circuit

For a quantum circuit $C = U_1, \ldots, U_m$ working on n quibts and a quantum state $|x\rangle \in \mathbb{C}^{2^n}$ we define the acceptance probability as follows:

$$\text{Prob}[C \text{ accepts } |x\rangle] = \langle x| U_1^\dagger \cdots U_m^\dagger \Pi_1 U_m \cdots U_1 |x\rangle .$$

In other words:

- We first apply the unitary operations $U_1, \ldots, U_m$ to the quantum state $|x\rangle$ (first U_1, then U_2, etc.) and obtain $|y\rangle = U_m \cdots U_1 |x\rangle$.
- We then measure the first qubit in $|y\rangle$. The probability that we get 1 in the first qubit (i.e., that the post measurement state is of the form $|1\rangle \otimes |z\rangle$ for an $(n-1)$-qubit state $|z\rangle$ is $\langle y| \Pi_1 |y\rangle$.

  Often, we will also measure another qubit in $|y\rangle$ instead of the 1st one – this makes no difference.

If $u \in \{0,1\}^n$ then we define: $\text{Prob}[C \text{ accepts } u] = \text{Prob}[C \text{ accepts } |u\rangle]$.

P-uniform families of quantum circuits

In the following we only consider quantum circuits that are composed of CNOT-gates, H-gates and T-gates, and where the first qubit is measured at the end.

Consider a family $(Q_n)_{n\geq 0}$ of such quantum circuits, where every Q_n works on $p(n)$ qubits for a polynomial $p(n)$.

$(Q_n)_{n\geq 0}$ is called **P**-uniform if there is a PTM M that produces on input 1^n (n 1-bits) a binary encoding of Q_n.

Note that for a **P**-uniform quantum circuit family $(Q_n)_{n\geq 0}$ there must exist a polynomial $q(n)$ such that if $Q_n = U_1, U_2, \ldots, U_m$ then $m \leq q(n)$.

The class **promiseBQP**

The class **promiseBQP** consists of all promise problems (L_0, L_1) for which there exists a **P**-uniform quantum circuit family $(Q_n)_{n\geq 0}$ and a polynomial $a(n)$ such that Q_n works on $n + a(n)$ qubits and for every $u \in \{0,1\}^n$ the following hold:

- If $u \in L_1$ then $\text{Prob}[Q_n \text{ accepts } u0^{a(n)}] \geq \frac{2}{3}$.
- If $u \in L_0$ then $\text{Prob}[Q_n \text{ accepts } u0^{a(n)}] \leq \frac{1}{3}$.

Remark:

- In the beginning, the first n qubits are set to the classical bits from the input $u \in \{0,1\}^n$.
- The other $a(n)$ qubits are called ancilla qubits and initialized to 0.
- Ancilla qubits are needed to make the whole computation reversible.
- In the following we will simply say **BQP** instead of **promiseBQP**; it stands for bounded-error quantum polynomial time.

$\mathbf{BPP} \subseteq \mathbf{BQP}$

BQP is usually identified with the class of all (promise) problems that can be solved efficiently using a quantum computer.

In the following, we also refer implicitly for classical complexity classes (like **P**, **BPP** or **PSPACE**) always to the corresponding promise class.

Theorem 18

$\mathbf{BPP} \subseteq \mathbf{BQP}$

Proof: We first show that $\mathbf{P} \subseteq \mathbf{BQP}$.

By Theorem 1 (slide 21) it suffices to simulate a classical Boolean circuit by a quantum circuit.

BPP $\subseteq$ BQP

What does it mean to simulate a Boolean circuit by a quantum circuit?

Boolean circuits are in general irreversible (i.e., one cannot reverse the computation).

For instance the binary AND-function $(a, b) \mapsto a \wedge b$ is irreversible.

Quantum circuits are always reversible (if we omit the final measurement).

The Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be simulated by the bijection $\tilde{f} : \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ with:

$$\forall u \in \{0,1\}^n \, \forall a \in \{0,1\} : \tilde{f}(u\, a) \mapsto u\, (a \oplus f(u)),$$

where $\oplus$ denotes the XOR (addition modulo two).

$\mathbf{BPP} \subseteq \mathbf{BQP}$

In particular: $\tilde{f}(u\,0) \mapsto u\,f(u)$.

We say that a quantum circuit $Q = U_1, U_2, \ldots, U_m$ on $(n+1+k)$ qubits computes the boolean function $f : \{0,1\}^n \to \{0,1\}$ if for all

$$\forall u \in \{0,1\}^n \; \forall a \in \{0,1\} : U_m \cdots U_2 U_1 \, |u\,a\,0^k\rangle = |u\,(f(u) \oplus a)\,0^k\rangle \,.$$

Consider now a boolean circuit family $(C_n(x_1, \ldots, x_n))_{n \geq 1}$.

Goal: compute $f_n := f_{C_n} : \{0,1\}^n \to \{0,1\}$ by a quantum circuit Q_n.

We can assume that C_n is built up from NAND-gates.

NAND stands for not-and and it works as follows:

$$\mathrm{NAND}(0,0) = \mathrm{NAND}(0,1) = \mathrm{NAND}(1,0) = 1, \quad \mathrm{NAND}(1,1) = 0$$

$\mathbf{BPP} \subseteq \mathbf{BQP}$

NAND is universal; it can simulate AND, OR and NOT.

First consider the so-called Toffoli-gate, also called CCNOT: for all $a, b, c \in \{0, 1\}$ we have

$$\mathrm{CCNOT}\,|a\,b\,c\rangle = |a\,b\,(a \wedge b) \oplus c\rangle\,.$$

It swaps the 3rd qubit if the 1st and 2nd qubit are 1, otherwise nothing is done.

We have $\mathrm{CCNOT}\,|a\,b\,1\rangle = |a\,b\,\mathrm{NAND}(a, b)\rangle$.

The ancilla 1-bit can be obtained from an ancilla 0-bit using a Pauli-X.

Using SWAP, CCNOT and copying of classical bits (see Slide 66) we can realize a quantum circuit $U_1, \ldots, U_m$ such that

$$\forall u \in \{0, 1\}^n : U_m \cdots U_1\,|u\,0^{k+1}\rangle \mapsto |u\,f(u)\,g(u)\rangle\,,$$

where $g(u) \in \{0, 1\}^k$ is garbage produced during the computation.

$\mathbf{BPP} \subseteq \mathbf{BQP}$

We can get rid of the garbage $g(u)$ as follows:

Add an additional qubits.

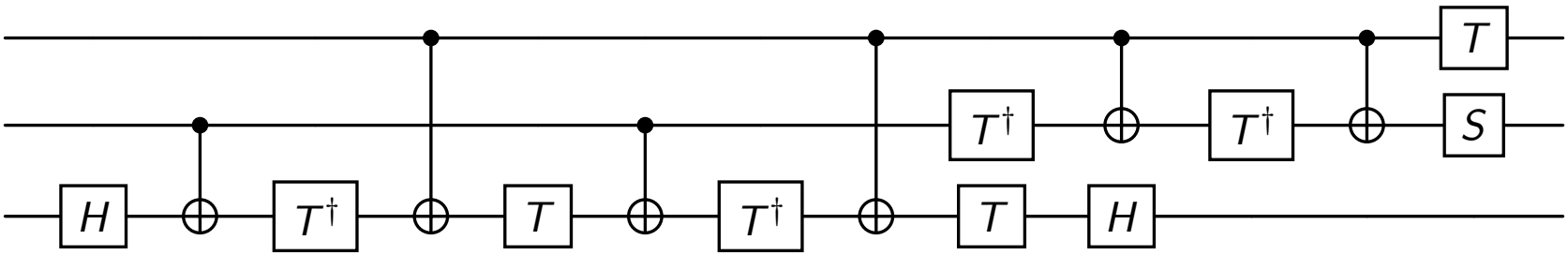
Starting from the computational basis state $|u\,0\,0^k\,a\rangle$ with $u \in \{0,1\}^n$, $a \in \{0,1\}$, our quantum circuit (for input length n) behaves as follows:

- Apply $U_1, \ldots, U_m$ on the first $n+1+k$ qubits to get $|u\,f(u)\,g(u)\,a\rangle$
- With a CNOT we get $|u\,f(u)\,g(u)\,(f(u) \oplus a)\rangle$
- Reversing the first step (apply $U_m^\dagger, \ldots, U_1^\dagger$) yields $|u\,0\,0^k\,(f(u) \oplus a)\rangle$

This is called the compute-uncompute trick

BPP $\subseteq$ BQP

CCNOT can be obtained from H, T and CNOT as follows, where $S = T^2$ (note that $T^\dagger = T^{-1} = T^7$):



(see Nielsen, Chuang, page 182).

This finally yields the desired quantum circuit Q_n.

Q_n is error-free: Applying Q_n to the computational basis state $|u\,0^{k+2}\rangle$ results in the computational basis state $|u\,0^{k+1}\,f_n(u)\rangle$ and measuring the last qubit gives the correct answer with probability 1.

Note: If the family of Boolean circuits $(C_n)_{n\geq 0}$ is **P**-uniform, then also $(Q_n)_{n\geq 0}$ is **P**-uniform.

$\mathbf{BPP} \subseteq \mathbf{BQP}$

This shows $\mathbf{P} \subseteq \mathbf{BQP}$.

It remains to show $\mathbf{BPP} \subseteq \mathbf{BQP}$. Let $L \in \mathbf{BPP}$.

Recall: In **BPP** we have $r(n)$ random bits available (where n is the length of the input). These are randomly set to 0 or 1.

Let C_n be a Boolean circuit with $n + r(n)$ inputs such that for all $u \in \{0,1\}^n$ we have (with $f_n = f_{C_n}$):

- If $u \in L$ then $f_n(u, v) = 1$ for $\geq \frac{2}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.
- If $u \notin L$ then $f_n(u, v) = 1$ for $\leq \frac{1}{3} \cdot 2^{r(n)}$ many $v \in \{0,1\}^{r(n)}$.

BPP $\subseteq$ BQP

In the quantum circuit Q_n (for input length n) the $r(n)$ random bits from C_n are $r(n)$ many additional ancilla qubits that are initially set to 0.

To these ancilla qubits we apply $H^{\otimes n} = \underbrace{H \otimes H \otimes \cdots \otimes H}_{r(n) \text{ many}}$.

Then we apply a quantum circuit that computes the Boolean function f_n (using k additional ancilla bits):

$$|u\, 0^{r(n)}\, 0^k\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^{r(n)/2}} \cdot \sum_{v \in \{0,1\}^{r(n)}} |u\, v\, 0^k\rangle$$

$$\xrightarrow{\text{quantum circuit for } f_n} \frac{1}{2^{r(n)/2}} \cdot \sum_{v \in \{0,1\}^{r(n)}} |u\, v\, 0^{k-1}\, f_n(u,v)\rangle$$

Measuring now the last qubit yields the same acceptance probability as the original boolean circuit. $\square$

Probability amplification for **BQP**

Theorem 19

The error probability $1/3$ in den Definition of **BQP** can be replaced by 2^{-n} where $n =$ input length.

Proof sketch: Assume we have a promise problem $(L_0, L_1) \in \mathbf{BQP}$.

Let $(Q_n)_{n \geq 1}$ be the corresponding quantum circuit family with error probability $\leq 1/3$.

Fix an input $u \in \{0,1\}^n$ and let $Q_n = U_1, U_2, \ldots, U_m$ be the corresponding quantum circuit for input length n. We ignore ancilla qubits below.

Then we can write $U_m \cdots U_2 U_1 |u\rangle = (\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes |x\rangle$ for a (not necessarily basis) state $|x\rangle$ on $n-1$ qubits and we have:

- If $u \in L$ then $|\alpha_0|^2 \leq 1/3$ (and $|\alpha_1|^2 = 1 - |\alpha_0|^2 \geq 2/3$).
- If $u \notin L$ then $|\alpha_1|^2 \leq 1/3$ (and $|\alpha_0|^2 = 1 - |\alpha_1|^2 \geq 2/3$).

Probability amplification for **BQP**

The idea of the proof is the same as for **BPP**: execute $k = \Theta(n)$ independent copies of Q_n on input u and make a majority vote.

W.l.o.g. we assume $k = 2^\ell - 1$.

Step 1: Produce from the classical input u the following state using CNOT-gates (see Slide 66).

$$|u\rangle^{\otimes k} = \underbrace{|u\rangle \otimes \cdots \otimes |u\rangle}_{k \text{ many}}$$

Step 2: Execute $U_1^{\otimes k}, U_2^{\otimes k}, \ldots, U_m^{\otimes k}$ (a quantum circuit with km gates).

Step 3: The result can be written as $\left((\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes |x\rangle\right)^{\otimes k}$, or, after applying SWAP operations as ($v[i]$ is the i-th bit in the bit string v):

$$(\alpha_0 |0\rangle + \alpha_1 |1\rangle)^{\otimes k} \otimes |x\rangle^{\otimes k} = \sum_{v \in \{0,1\}^k} \left(\prod_{i=1}^{k} \alpha_{v[i]} \right) |v\rangle \otimes |x\rangle^{\otimes k}.$$

Probability amplification for **BQP**

We ignore $|x\rangle^{\otimes k}$ in the following.

Step 4: One could now measure the first k bits and make a majority vote (i.e., accept iff at least $k/2$ of the measured bits are 1).

Alternatively one can also apply to the first k qubits plus ℓ many ancilla qubits w the transformation $|v\ w\rangle \to |v\ (w \oplus \mathrm{bin}(|v|_1)\rangle$.

Here, $|v|_1$ is the number of 1's in v (a number in $[0, 2^\ell - 1]$) and $\mathrm{bin}(|v|_1) \in \{0,1\}^\ell$ its binary representation.

Assuming $w = 0^\ell$ in the beginning, we get the state

$$\sum_{v\{0,1\}^k} \left(\prod_{i=1}^{k} \alpha_{v[i]} \right) |v\ \mathrm{bin}(|v|_1)\rangle\,.$$

Probability amplification for **BQP**

Step 5: We then measure the $(k+1)$st qubit.

Note: the first bit of $\text{bin}(|v|_1)$ is 1 if and only if $|v|_1 \geq k/2$.

Therefore, the probability that in the post-measurement state the $(k+1)$st qubit is 1 is

$$\sum_{v \in \{0,1\}^k, |v|_1 \geq k/2} \prod_{i=1}^{k} |\alpha_{v[i]}|^2. \tag{3}$$

Assume $u \notin L$ (the case $u \in L$ is analogous). Then (3) is the error probability and we have $|\alpha_1|^2 \leq 1/3$ and $|\alpha_0|^2 \leq 2/3$.

The probability (3) can be obtained also by taking k independent Bernoulli random variables $X_i \in \{0,1\}$ with $\text{Prob}[X_i = 1] = |\alpha_1|^2 \leq 1/3$ for all i.

Then (3) $= \text{Prob}[\sum_{i=1}^{k} X_i \geq k/2] \leq e^{-\Theta(k)}$ by the Chernoff bound. □

Probability amplification for **BQP**

The ability to do probability amplification is important for BQP.

Assume that we solve a problem with a **P**-uniform family $(Q_n)_{n\geq 0}$ of quantum circuits using only d-qubit quantum gates for a constant d.

Let $p(n)$ be the number of gates in Q_n (a polynomial) and let $1/3$ be the error probability.

- Theorem 8 (slide 69) $\rightarrow$ quantum circuit of size $\mathcal{O}(p(n)$ consisting of CNOT and 1-qubit gates.
- Probability amplification $\rightarrow$ error probability can be reduced to $1/6$.

  Thereby the number of gates only increases by a constant.

  Solovay-Kitaev (slide 77) & Lemma 17 (slide 84) with $\epsilon = 1/12 \rightarrow$ quantum circuit with $\mathcal{O}(p(n)\log^c p(n)) = \mathcal{O}(p(n)\log^c n)$ gates and error of $1/6 + 2/12 = 1/3$:

Quantum Fourier transformation (QFT)

The d-dimensional QFT is the quantum transformation defined by

$$\forall a \in \{0, \ldots, a-1\} : |a\rangle \mapsto \frac{1}{\sqrt{d}} \sum_{b=0}^{d-1} \omega^{ab} \cdot |b\rangle$$

where $\omega = e^{2\pi i/d}$ is a primitive root of unity of order d.

It satisfies $\omega^d = 1$ and $\omega^k \neq 1$ for $1 \leq k \leq d-1$.

The corresponding $(d \times d)$-matrix is

$$\mathrm{QFT}_d = \frac{1}{\sqrt{d}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^{d-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(d-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{d-1} & \omega^{2(d-1)} & \cdots & \omega^{(d-1)^2} \end{pmatrix}$$

Quantum Fourier transformation (QFT)

Lemma 20

QFT_d is unitary.

This will follow as a corollary soon.

Assume in the following that $d = 2^n$ and identify the basis state $|a\rangle$ $(0 \leq a \leq 2^n - 1)$ with the n-bit binary representation $|a_1 a_2 \cdots a_n\rangle$ of a.

In other words: $a = \sum_{j=1}^{n} a_j 2^{n-j}$.

In the following, we make use of the rational numbers

$$0.a_j a_{j+1} \cdots a_n = \frac{a_j}{2} + \frac{a_{j+1}}{4} + \cdots + \frac{a_n}{2^{n-j+1}} = \frac{a}{2^j}.$$

Quantum Fourier transformation (QFT)

Lemma 21

QFT_{2^n} maps the computational basis state $|a\rangle = |a_1 a_2 \cdots a_n\rangle$ $(0 \leq a \leq 2^n - 1)$ to

$$\frac{1}{2^{n/2}} \bigotimes_{j=n}^{1} \left(|0\rangle + e^{2\pi i (0.a_j a_{j+1} \cdots a_n)} |1\rangle \right) \tag{4}$$

Proof: We have

$$\begin{aligned} \mathrm{QFT}_{2^n} |a\rangle &= \frac{1}{\sqrt{2^n}} \sum_{b=0}^{d-1} \omega^{ab} \cdot |b\rangle \\ &= \frac{1}{2^{n/2}} \sum_{b_1=0}^{1} \cdots \sum_{b_n=0}^{1} \exp(2\pi i \, a \sum_{j=1}^{n} b_j 2^{n-j}/2^n) \, |b_1 b_2 \cdots b_n\rangle \end{aligned}$$

Quantum Fourier transformation (QFT)

$$
\begin{aligned}
&= \frac{1}{2^{n/2}} \sum_{b_1=0}^{1} \cdots \sum_{b_n=0}^{1} \exp(2\pi i\, a \sum_{j=1}^{n} b_j 2^{-j}) \,|b_1 b_2 \cdots b_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{b_1=0}^{1} \cdots \sum_{b_n=0}^{1} \left(\prod_{j=1}^{n} \exp(2\pi i\, a\, b_j 2^{-j}) \right) |b_1\rangle \otimes |b_2\rangle \otimes \cdots |b_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{b_1=0}^{1} \cdots \sum_{b_n=0}^{1} \bigotimes_{j=1}^{n} \exp(2\pi i\, a\, b_j 2^{-j}) \,|b_j\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{j=1}^{n} \sum_{b_j=0}^{1} \exp(2\pi i\, a\, b_j 2^{-j}) \,|b_j\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{j=1}^{n} (|0\rangle + \exp(2\pi i\, a\, 2^{-j}) \,|1\rangle) \overset{(*)}{=} (4)
\end{aligned}
$$

For $(*)$ note that $\exp(2\pi i\, a_1 \cdots a_{k-1}.a_k \cdots a_n) = \exp(2\pi i\, 0.a_k \cdots a_n)$. □

Quantum Fourier transformation (QFT)

Using Lemma 21 we can obtain a quantum circuit for QFT_{2^n}.

Define the 1-qubit quantum gate

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$$

and use it's controlled version: Apply R_k to the 2nd quibt if $a = 1$, otherwise do noting.

Formally, this is the 2-qubit quantum gate such that for all $a \in \{0, 1\}$ and all quantum states $|x\rangle \in \mathbb{C}^2$:

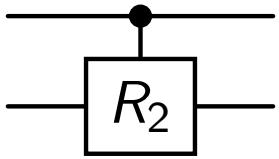
$$|a\rangle \otimes |x\rangle \mapsto |a\rangle \otimes R_k^a |x\rangle .$$

Its matrix is

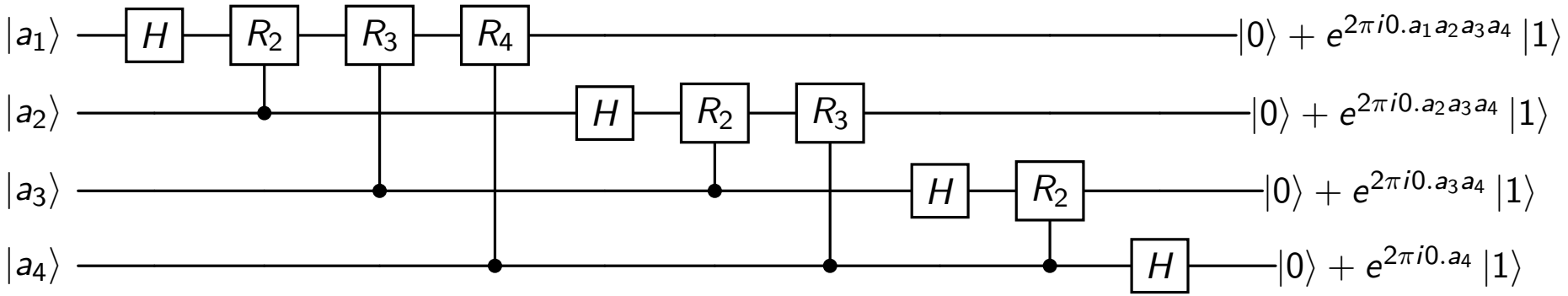
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix}$$

Quantum Fourier transformation (QFT)

The graphical notation for the controlled-R_k gate is:



Then we obtain the following quantum circuit for QFT_{2^n} (here for $n = 4$):



At the end one has to do SWAP-operations in order to exchange the qubits i and $n+1-i$ for $1 \leq i \leq n$.

Some BQP-complete problems

The class **MA**

MA stands for Merlin-Arthur (we should call it **promiseMA**, but we omit the **promise**) and is defined as follows:

A promise problem (L_0, L_1) belongs to **MA** if there is a PTM M and a polynomials $p(n), r(n)$ such that for every $u \in \{0,1\}^n$:

- If $u \in L_1$ then $\exists v \in \{0,1\}^{p(n)}$: $\langle u, v, w \rangle \in L(M)$ for $\geq \frac{2}{3} \cdot 2^{r(n)}$ many $w \in \{0,1\}^{s(n)}$
- If $u \notin L_0$ then $\forall v \in \{0,1\}^{p(n)}$: $\langle u, v, w \rangle \notin L(M)$ for $\leq \frac{1}{3} \cdot 2^{s(n)}$ many $w \in \{0,1\}^{r(n)}$.

MA can be seen as a randomized version of **NP**:

- Merlin proposes to Arthur a "proof" v for $u \in L_1$.
- Arthur can verify in polynomial with high probability whether it is really a proof.

The class **QMA**

We now define a quantum analog of the class **NP** (actually a quantum analog of **MA**).

A promise problem (L_0, L_1) belongs to **QMA** if there are polynomials $p(n)$, $a(n)$ and a **P**-uniform quantum circuit family $(Q_n)_{n\geq 0}$ working on $n + a(n) + p(n)$ qubits such that for every $u \in \{0,1\}^n$:

- if $u \in L_1$ then there is a quantum state $|x\rangle \in \mathbb{C}^{2^{p(n)}}$:

$$\text{Prob}[Q_n \text{ accepts } |u\, 0^{a(n)}\rangle \otimes |x\rangle] \geq \frac{2}{3}.$$

- If $u \in L_0$ then for every quantum state $|x\rangle \in \mathbb{C}^{2^{p(n)}}$:

$$\text{Prob}[Q_n \text{ accepts } |u\, 0^{a(n)}\rangle \otimes |x\rangle] \leq \frac{1}{3}.$$

The class **QMA**

Remarks:

- the "proof" $|x\rangle$ is an arbitrary $p(n)$-qubit quantum state and not necessarily a computational basis state.
- If one requires that the "proof" $|x\rangle$ is a computational basis state $|v\rangle$ with $v \in \{0,1\}^{p(n)}$ then one obtains the class **QCMA**.
- **NP** $\subseteq$ **QCMA** $\subseteq$ **QMA**:
- **BQP** $\subseteq$ **QMA**.