Quantum Computing
Lecture 8

Quantum Automata and Complexity

Maris Ozols

Computational models and complexity

Shor's algorithm solves, in polynomial time, a problem for which no classical polynomial time algorithm is known.

What class of problems are solvable by quantum machines in polynomial time?

How does the model of quantum computing compare to other models, such as ones with randomness or non-determinism?

How does the quantum model of computation affect our understanding of computational complexity?

This lecture:

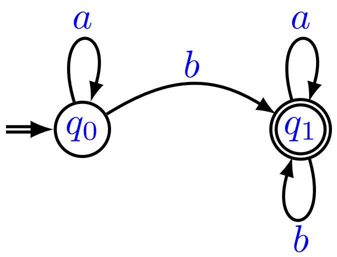
1. finite-state automata
2. computational complexity

Finite-state automata

A finite-state automaton (FSA) consists of

- a finite set of states Q
- a finite input alphabet Σ
- an initial state and some accepting states $A \subseteq Q$
- transitions between states

Example:



- states: $Q = \{q_0, q_1\}$
- alphabet: $\Sigma = \{a, b\}$
- initial state: q_0,
  accepting states: $A = \{q_1\}$
- transitions: $q_0 \xrightarrow{a} q_0$, $q_0 \xrightarrow{b} q_1$, ...

It accepts the set of all strings over $\{a, b\}$ that contain at least one b.

Its operation can be described by a pair of transition matrices:

$$M_a = \begin{array}{cc} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{array}{c} |q_0\rangle \\ |q_1\rangle \end{array} \end{array} \qquad M_b = \begin{array}{cc} \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} & \begin{array}{c} |q_0\rangle \\ |q_1\rangle \end{array} \end{array}$$

That is, $M_a|q_i\rangle = |q_i\rangle$ and $M_b|q_i\rangle = |q_1\rangle$ for all $i \in \{0, 1\}$. After reading the string abb, the transitions between states are described by $M_b M_b M_a$.

Different flavours of FSA

An n-state automaton with alphabet Σ is described by a collection of $n \times n$ matrices $\{M_a : a \in \Sigma\}$, i.e., one matrix for each symbol $a \in \Sigma$.

Depending on allowed M, we get different flavours of automata:

- **Deterministic:** M is a $0/1$ matrix and each column of M has exactly one entry 1
- **Reversible:** M is a $0/1$ matrix and each row and column of M has exactly one entry 1, i.e., M is a permutation matrix
- **Non-deterministic:** M is an arbitrary $0/1$ matrix
- **Probabilistic:** all entries of M are in $[0, 1]$ and each column sums up to 1, i.e., M is a stochastic matrix
- **Quantum:** M is unitary

Example: Here are different types of transitions matrices:

$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1/2 & 0 \\ 1/2 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
deterministic	*reversible*	*non-deterministic*	*probabilistic*	*quantum*

Intermediate states

Depending on the type of automata, we get different types of intermediate states:

- **Deterministic:** it is a standard basis vector $|q\rangle$ for some $q \in Q$
- **Reversible:** same as above; moreover, given the last input symbol we can also recover the previous state by applying M^{T}
- **Non-deterministic:** $\sum_{q \in Q} c_q |q\rangle$ where $c_q \in \{0, 1, 2, \ldots\}$
- **Probabilistic:** $\sum_{q \in Q} c_q |q\rangle$ where $c_q \in [0, 1]$ and $\sum_{q \in Q} c_q = 1$; equivalently, it is a probability distribution over Q
- **Quantum:** $\sum_{q \in Q} c_q |q\rangle$ where $c_q \in \mathbb{C}$ and $\sum_{q \in Q} |c_q|^2 = 1$; equivalently, it is a quantum state that is a superposition over Q

Note: For non-deterministic automata, $c_q \in \mathbb{N}$ counts the number of different paths from the initial state to q using transitions given by the input string. Normally one cares only about the existence of a path, so we can replace all $c_q \geq 1$ by 1 so that $c_q \in \{0, 1\}$ and the state is always of the form $\sum_{q \in S} |q\rangle$ for some subset $S \subseteq Q$.

Acceptance

Let $A \subseteq Q$ denote the set of all accepting states. Then we accept the input word if the current state $\sum_{q \in Q} c_q |q\rangle$ satisfies:

- **Deterministic:** the state is $|q\rangle$ for some $q \in A$
- **Reversible:** same as above
- **Non-deterministic:** $\sum_{q \in A} c_q \geq 1$, i.e., at least one accepting state
- **Probabilistic:** $\sum_{q \in A} c_q \geq 2/3$
- **Quantum:** $\sum_{q \in A} |c_q|^2 \geq 2/3$

Note: The threshold $2/3$ is arbitrary (any fixed constant $> 1/2$).

The language accepted or recognized by a finite automaton consists of all finite strings over Σ that are accepted: $L \subseteq \Sigma^*$.

Some remarks

Deterministic, non-deterministic, and probabilistic automata accept exactly the class of regular languages.

Automaton's complexity is $|Q|$, its number of states. Different models may require different number of states to solve the same problem.

Reversible and quantum automata are weaker because of reversibility.

Possible fix: we can allow intermediate measurements in the quantum case. Then we can simulate randomness by measuring $|+\rangle$. This makes quantum automata at least as powerful as probabilistic ones.

Turing machines are much more interesting than finite automata...

Turing machine

Turing machine (TM) consists of a finite automaton attached to an infinite read-write tape. The tape is initially blank and contains only a finite-length input.

TM is determined by an alphabet Σ, a finite set of states Q, and a transition function δ which, for each symbol and state, gives:

- next state,
- new replacement symbol to write on the tape,
- direction in which to move the tape head.

TM has infinitely many possible configurations (reserving the word "state" for elements of Q). Each configuration c is determined by a state, the contents of the tape (a finite string), and the head's position.

Acceptance

Let c_0 denote the starting configuration of a TM: the automaton is in its initial state, the tape contains the input $w \in \Sigma^*$, and the tape head is at the left end of the string.

The input w is accepted if the computation $c_0 \to c_1 \to \cdots \to c_t$ eventually reaches an accepting state, where t denotes the amount of time the computation takes.

The action of the TM can equivalently be described as a linear operator on an infinite-dimensional space, where the set of configurations form a basis for the space.

This operation should be reversible/unitary for reversible/quantum Turing machines. Such description is very hard to work with, so that's why we instead use quantum circuits (recall Lecture 4).

Computational resources and models

Models of computation:

- deterministic
- non-deterministic
- probabilistic
- quantum

Resources:

- time
- space

Randomness supplied upfront: Randomized computation can implemented by a deterministic Turing machine that has a supply of random bits written on the tape.

Computation vs verification: Non-deterministic computation can be thought of as verification. If we make all the non-deterministic choices upfront, this corresponds to guessing the answer and then verifying that it is indeed correct.

Randomness from quantumness: Quantum computer can obtain random bits by measuring $|+\rangle$.

Basic complexity classes

Promise problem: $A = (A_{\text{yes}}, A_{\text{no}})$ where $A_{\text{yes}}, A_{\text{no}} \subseteq \Sigma^*$ such that $A_{\text{yes}} \cap A_{\text{no}} = \varnothing$. We promise not to give inputs $\Sigma^* - (A_{\text{yes}} \cup A_{\text{no}})$.

Example (cat videos): $A_{\text{yes}} \cup A_{\text{no}}$ is the set of all possible videos where A_{yes} contains cats but A_{no} does not. $\Sigma^* - (A_{\text{yes}} \cup A_{\text{no}})$ are not videos.

Complexity classes: Given $A = (A_{\text{yes}}, A_{\text{no}})$, let M be a deterministic poly-time Turing machine that receives $x \in A_{\text{yes}} \cup A_{\text{no}}$. Then

- $A \in \mathsf{P} =$ **(deterministic) Polynomial-time:** *if M accepts all $x \in A_{yes}$ and rejects all $x \in A_{no}$*
- $A \in \mathsf{PP} =$ **Probabilistic Polynomial-time:** *if M can access random bits and accepts/rejects with probability $> 1/2$*
- $A \in \mathsf{BPP} =$ **Bounded-error Probabilistic Polynomial-time:** *if M can access random bits and accepts/rejects with probability $\geq 2/3$*
- $A \in \mathsf{BQP} =$ **Bounded-error Quantum Polynomial-time:** *if M produces a poly-time quantum circuit that accepts/rejects with probability $\geq 2/3$*

Verification complexity

King Arthur asks an all-powerful Wizard Merlin to convince him that x is a correct solution. But he is afraid to get fooled by Merlin!

Arthur has x, Merlin provides a proof y

Given $A = (A_{\text{yes}}, A_{\text{no}})$, let M be a deterministic poly-time Turing machine (Arthur) that receives $x \in A_{\text{yes}} \cup A_{\text{no}}$. Moreover:

- for every $x \in A_{\text{yes}}$ there is a proof y such that M accepts (x, y)
- for every $x \in A_{\text{no}}$ there is no proof y such that M accepts (x, y)

Verification complexity classes

Complexity classes:

- $A \in$ NP $=$ **Non-deterministic Polynomial-time:**
  Arthur is a deterministic poly-time P *algorithm*
- $A \in$ MA $=$ **Merlin–Arthur:**
  Arthur is probabilistic, BPP*; accepts/rejects with probability* $\geq 2/3$
- $A \in$ QMA $=$ **Quantum Merlin–Arthur:**
  Arthur is quantum, BQP*; Merlin supplies him with a quantum state as a proof and Arthur accepts/rejects with probability* $\geq 2/3$

Summary of complexity classes

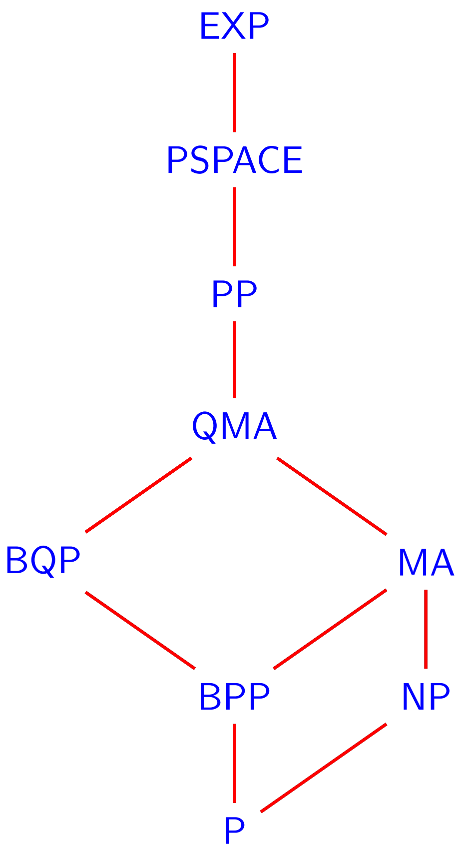
	Finding solution	Verifying solution
Deterministic	P	NP
Probabilistic	BPP	MA
Quantum	BQP	QMA

Some more classes:

- PP $=$ like BPP but unbounded error
- EXP $=$ like P but exponential-time
- PSPACE $=$ like P but polynomial-space (and any time)
- NP-complete $=$ "the hardest" problems in NP

See `https://complexityzoo.uwaterloo.ca/` for even more!

Inclusions



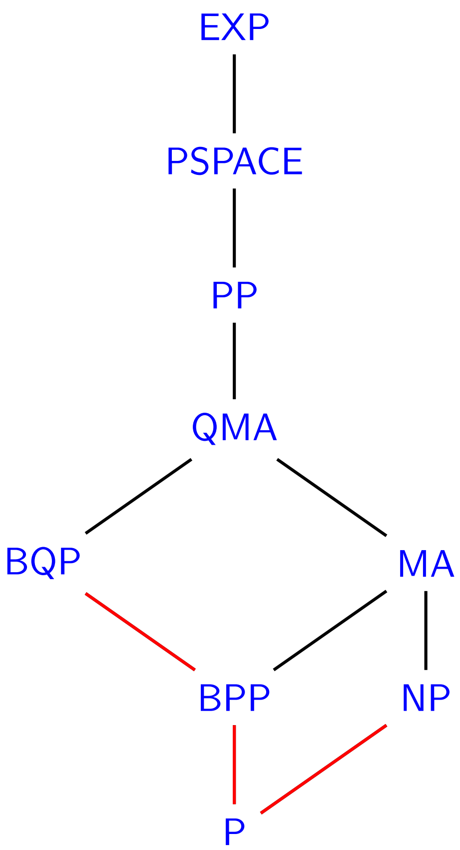
Trivial inclusions:

- randomness at least as good as determinism: P $\subseteq$ BPP, NP $\subseteq$ MA
- quantumness at least as good as randomness: BPP $\subseteq$ BQP, MA $\subseteq$ QMA
- verifying a solution cannot be harder than finding one: P $\subseteq$ NP, BPP $\subseteq$ MA, BQP $\subseteq$ QMA
- poly space can store at most exp number of different strings: PSPACE $\subseteq$ EXP

Non-trivial inclusions:

- GapP functions: BQP $\subseteq$ PP
- guess a random proof: QMA $\subseteq$ PP
- try all strings and count solutions: PP $\subseteq$ PSPACE

Separations?



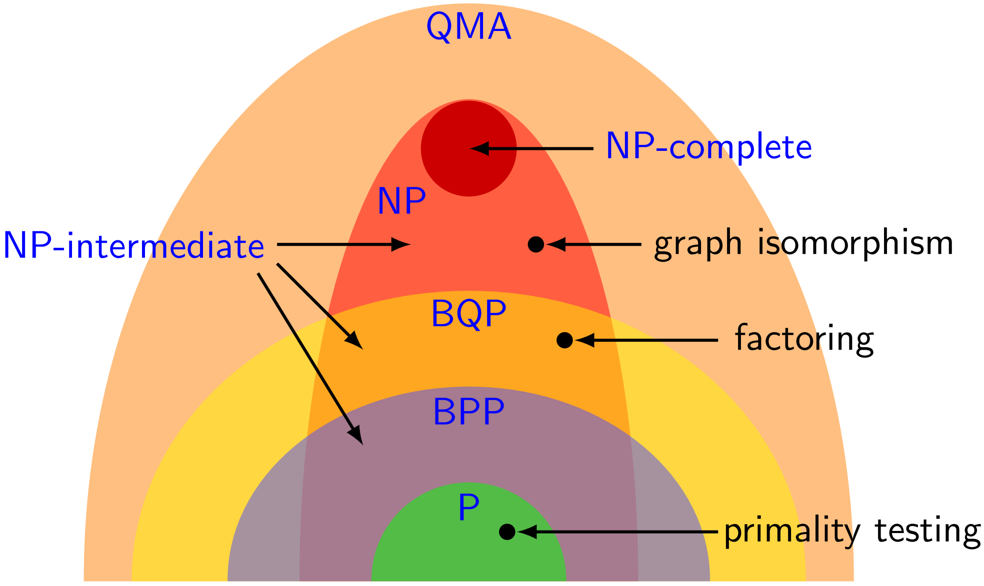
Non-trivial separations:

- Time hierarchy theorem: P $\subsetneq$ EXP

Major open problems:

- P vs NP – is finding a solution really harder than verifying one? (you can get $\$10^6$ for solving this)
- P vs BPP – does randomness help?
- BPP vs BQP – are quantum computers more powerful that (probabilistic) classical ones? E.g., is factoring hard classically (cannot be done in poly time)?
- P vs PSPACE – we have no clue even on how to show that these to classes are different. . .

Problems of interest



Most problems in NP are either complete or in P. The outliers are called

$$\text{NP-inermediate} = \text{NP} - (\text{NP-complete} \cup \text{P})$$

Ladner's Theorem: $\text{P} = \text{NP}$ if and only if NP-inermediate is empty.

Technology cannot make
the world a better place,
people can.

Thank you!