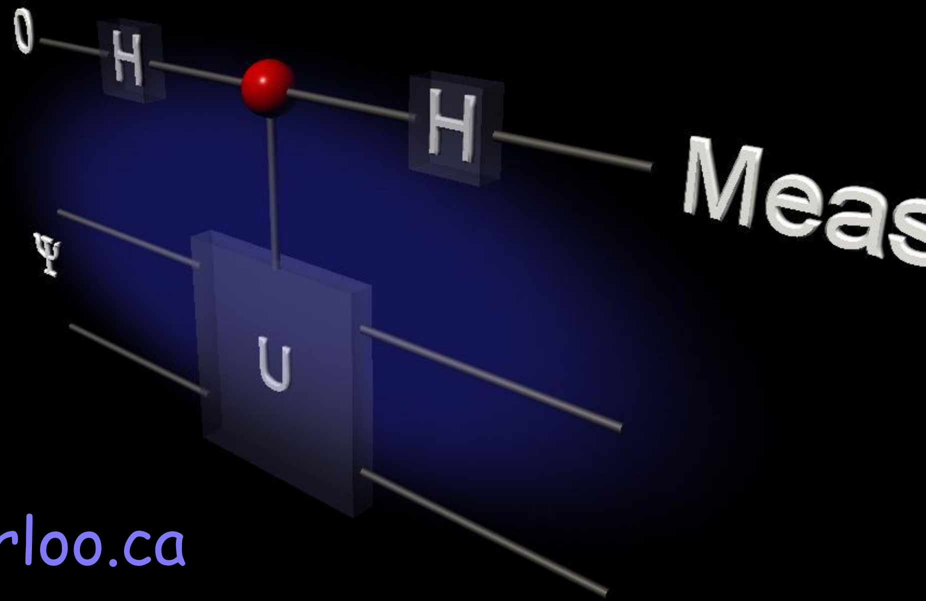
# Introduction to Quantum Information Processing

CS467 C&O481 PHYS467

**Michele Mosca** mmosca@iqc.uwaterloo.ca

Tuesdays and Thursdays 10am-11:15am



## COMPLEXITY LECTURE

## Computational Complexity Classes

- *(Computational) Complexity* refers to some measure of the resources *(e.g. time, space, basic operations, energy)* required to solve a problem. We will restrict attention to decision problems.

- Decision problems can be treated as the problem of recognizing elements of a language.

## What is a "language" ?

- Fix an alphabet, say $\Sigma = \{0,1\}$. The set $\Sigma^*$ denotes all finite length strings over that alphabet.

- A *language* $L$ is a subset $L \subseteq \Sigma^*$

  *($L$ is the set of strings with some property you are interested in)*

- An algorithm "solves the language recognition problem for $L$" if it accepts any string $\sigma \in L$ and rejects any string $\sigma \notin L$

## What is a language?

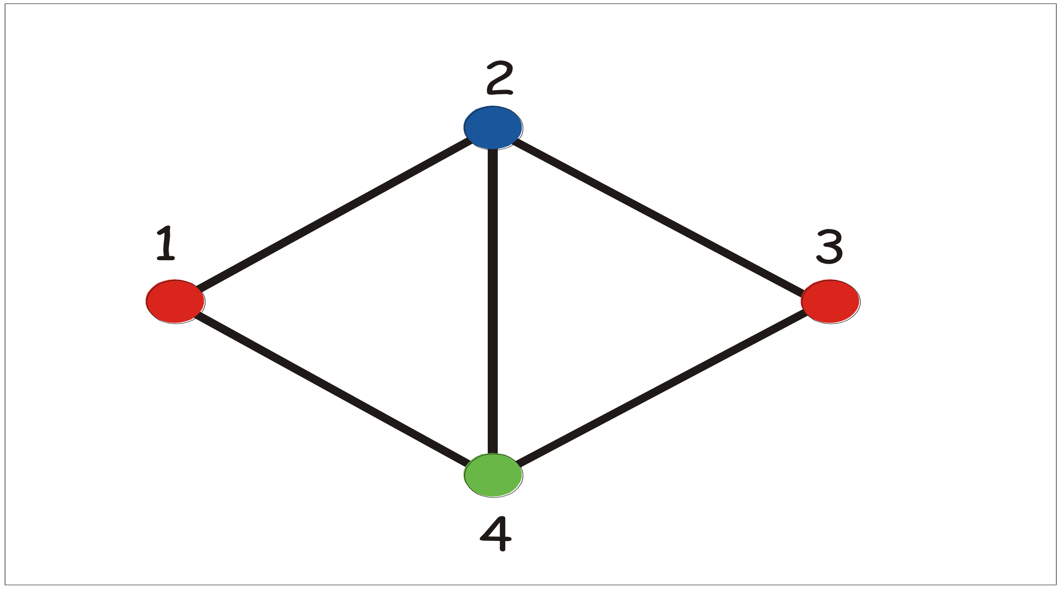
- E.g. $\text{PRIME} = \{10, 11, 010, 011, 101, 111, \ldots\}$

$$\text{COMPOSITE} = \{100, 110, 0100, 0110, 1000, \ldots\}$$

- If we let strings $x$ represent a graph, then we can define

$$3-\text{COLOURABLE} = \{x \mid x \text{ is properly } 3-\text{colourable}\}$$

# What is a language?



- The string 101111 represents this graph by telling us which pairs of vertices $\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}$ are joined. This graph is 3-colourable so

$$101111 \in 3-COLOURABLE$$

## A complexity class: P

- The class P consists of all languages L for which there exists a classical algorithm A running in worst-case polynomial time such that for any input $x \in \Sigma^*$ the algorithm A on input x, A(x), accepts if and only if $x \in L$ .

# A Complexity class: BPP

- The class BPP (bounded-error probabilistic polynomial time) consists of all languages L for which there exists a randomized classical algorithm A running with worst-case expected polynomial time such that for any input $x \in \Sigma^*$

$$If \;\; x \in L \Rightarrow \Pr[A(x)\; accepts] \geq \frac{2}{3}$$

$$If \;\; x \notin L \Rightarrow \Pr[A(x)\; accepts] \leq \frac{1}{3}$$

- N.B. we are not averaging over x

# Chernoff bound and BPP

- Is $\frac{2}{3}$ special? No. $\frac{1}{2}+\delta, \delta > 0$ suffices.

- We can repeat the algorithm $\mathrm{A}$ $n$ times and take the majority answer. We now get the correct answer with probability at least $1-\varepsilon^{n}$ for some $\varepsilon, 0<\varepsilon<1$ (see Example A.1.5 in the text).

## BPP ≈ Efficient

- We view decision problems corresponding to recognizing languages in BPP as tractable on a classical computer, and problems without such worst-case polynomial time solutions intractable.

# Certificates

- Note that although it might be very difficult to decide if a graph $\mathrm{x}$ is $3\text{-colourable}$, it is very easy to check if a given colouring $\mathrm{y}$ is a proper $3\text{-colouring}$

- I.e. there exists a polynomial time algorithm $CHECK\_3COLOURING(a,b)$ such that $CHECK\_3COLOURING(x,y) = accept$ .

## Non-Deterministic Polynomial Time (*NOT* “Non-Polynomial”)

- The class NP (non-deterministic polynomial time) consists of all languages L for which there exists a polynomial time algorithm A such that for any input $x \in \Sigma^*$

$x \in L \Rightarrow \exists\ y \in \Sigma^*$ such that $A(x, y)$ accepts
(|y| is bounded by a polynomial in |x|)

$x \notin L \Rightarrow \forall\ y \in \Sigma^*$ such that $A(x, y)$ rejects
(|y| is bounded by a polynomial in |x|)

## Exercise: Does P = NP?

### *Millennium Problems*

*In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) has named seven Prize Problems. The Scientific Advisory Board of CMI selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI designated a $7 million prize fund for the solution to these problems, with **$1 million** allocated to each.*

### P vs NP Problem

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair from taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

## A quantum class: BQP

- The natural quantum analogue of $\mathrm{P}$ would be some sort of “exact quantum polynomial time” class (often called $\mathrm{EQP}$), but it’s not as natural, or robust, and not considered much.

- We mostly focus on the quantum analogy of $\mathrm{BPP}$.

# A quantum class: BQP

- The class BQP (bounded-error quantum polynomial time) consists of all languages L for which there exists a quantum algorithm A running with worst-case expected polynomial time such that for any input $x \in \Sigma^*$

$$If \;\; x \in L \Rightarrow \Pr[A(x)\ accepts] \geq \frac{2}{3}$$

$$If \;\; x \notin L \Rightarrow \Pr[A(x)\ accepts] \leq \frac{1}{3}$$

- N.B. we are not averaging over x

- BQP is considered to be the set of problems efficiently solvable on a quantum computers

## A special problem in NP

- The language CIRCUIT_SAT consists of all strings x that encode classical acyclic circuits (using some reasonable encoding) for which there exists an input string y to the circuit x for which the circuit outputs a 1

- CIRCUIT_SAT is clearly in NP

## Reductions

- A language $L_1$ is <u>reducible</u> to another language $L_2$ if there exists a polynomial time classical algorithm $R$ that given input $x$ outputs $R(x)$ with the property that

$$x \in L_1 \Leftrightarrow R(x) \in L_2$$

- So if you can decide $L_2$ in polynomial time, you can decide $L_1$ in polynomial time (but not necessarily vice versa)

## Reductions

- E.g. It is easy to reduce 3-COLOURABLE to 4-COLOURABLE

## Hardness and Completeness

- A language L is hard with respect to a particular class C ("C-hard") if every language in C can be reduced to L

- A language L is complete with respect to a class C ("C-complete") if $L \in C$ and L is C-hard

## IMPORTANT THEOREM

- CIRCUIT_SAT is NP-complete

  (Cook-Levin)

1. CIRCUIT_SAT is in NP (just supply the satisifying input y as a certificate)
2. CIRCUIT_SAT is NP-hard (sketch: given any other NP language L with certificate checking procedure A(x,y), produce a circuit with input y that simulates A(x,y) )

## Other NP-Complete problems

- If we can reduce CIRCUIT_SAT to any other language L in NP, then L is also NP-complete

- E.g. SAT, 3-SAT, 3-COLOURING, 4-COLOURING, TRAVELLING SALESMAN, SUBSET SUM, finding ground state of certain Hamiltonians

- We often refer to problems in NP not known to be in NP-complete nor in BPP as NPI (intermediate NP); e.g. graph automorphism, factoring, discrete logarithms in finite fields
- It is strongly suspected that NP is strictly larger than P (famous P vs NP question)
- It is not clear if quantum algorithms can efficiently solve all problems in NP

- It is widely believed that no NP-hard problems can be solved in the worst case with polynomial resources

- However, people work on <u>approximating</u> solutions to NP-hard problems

# Other classes

- PSPACE is the set of language than can be recognized by a classical computer using only polynomial space (can use exponential time)
- Clearly $P \subseteq PSPACE$
- Can also see that $NP \subseteq PSPACE$
- It hasn't even been proved that $P \neq PSPACE$

# Merlin-Arthur

- The class MA (Merlin-Arthur) consists of all languages L for which there exists a polynomial time probabilistic algorithm A such that for any input $x \in \Sigma^*$
  - $x \in L \Rightarrow \exists\, y \in \Sigma^*$ such that $A(x, y)$ *accepts* <u>with high probability</u>

    (|y| is bounded by a polynomial in |x|)
  - $x \notin L \Rightarrow \forall\ y \in \Sigma^*$ such that $A(x, y)$ *rejects* <u>with high probability</u>

    (|y| is bounded by a polynomial in |x|)
- For most practical purposes, this is like NP. It doesn’t lend itself to a “complete” problem in the strict sense, but certain “promise problems” can serve the same role.

# Quantum Merlin-Arthur

- Since deterministic (or “exact”) quantum computation is not so natural, it is more natural to define a quantum analogue of MA instead of NP.

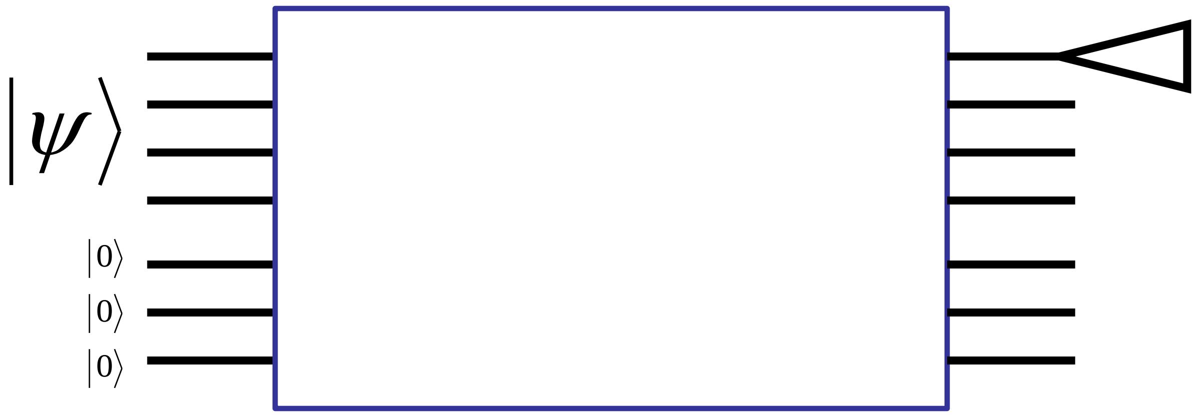
- The class QMA (Quantum Merlin-Arthur) consists of all languages L for which there exists a polynomial time quantum algorithm A such that for any input $x \in \Sigma^*$

$x \in L \Rightarrow \exists |y\rangle$ such that $A(x, |y\rangle)$ *accepts* with high probability (|y| is bounded by a polynomial in |x|)

$x \notin L \Rightarrow \forall |y\rangle$ such that $A(x, |y\rangle)$ *rejects* with high probability (|y| is bounded by a polynomial in |x|)

# QMA-hard problem

The most obvious QMA-hard problem would be the equivalent of CIRCUIT-SAT, where one is given a quantum circuit with some fixed input bits, a particular output bit, and one asks whether there is an input state $|\psi\rangle$ on the remaining input bits such that the output bit value is $1$ with probability at least $2/3$.



It is still QMA-hard if we promise that either there is such a state $|\psi\rangle$ or for any state $|\psi\rangle$ the circuit outputs $1$ with probability less than $\frac{2}{3}-\varepsilon$ for some $\varepsilon > 0$. People loosely refer to such problems as "QMA-complete", or more accurately as "promise QMA-complete".

## (promise) QMA-complete problems

Many more natural (promise) problems have been shown to be "QMA-complete".

e.g. Does a given $n$-qubit Hamiltonian $\mathrm{H}$ (described by poly($n$) local Hamiltonians) have an eigenvector with eigenvalue (i.e. energy) less than $\mathrm{E}$? (with a promise that either it does, or the energies are above $\mathrm{E} + \mathrm{g}$, for some positive constant gap $\mathrm{g}$).

For many families of natural Hamiltonians, this problem is "QMA-complete".

e.g. Suppose we have an $n$-qubit system, and we are given a collection of local density matrices where each density matrix describes a specific subset of the qubits. We say that the matrices are "consistent" if there exists some global state $\sigma$ (on all $n$ qubits) that matches each of the local density matrices on the appropriate subsets.

With an appropriate promise, this is "QMA-complete".

# Interactive Proof Systems

- NP, MA, and QMA correspond to restricted “interactive proof systems”

- A <u>prover</u>, P, presents a proof that a given string $n$ is a member of some language
- A <u>verifier</u>, V, checks that the proof is correct.

- *The prover is assumed to have unbounded computational power, while the verifier is a probabilistic polynomial-time machine with access to a random bit string whose length is polynomial in the size of n.*
- *These two machines exchange a polynomial number, p(n), of messages and once the interaction is completed, the verifier must decide whether or not n is in the language.*

# Interactive Proof Systems

- The class IP consists of all languages L such that there exists a verifier V and a prover P such that for any $x \in \Sigma^*$ , and any prover Q

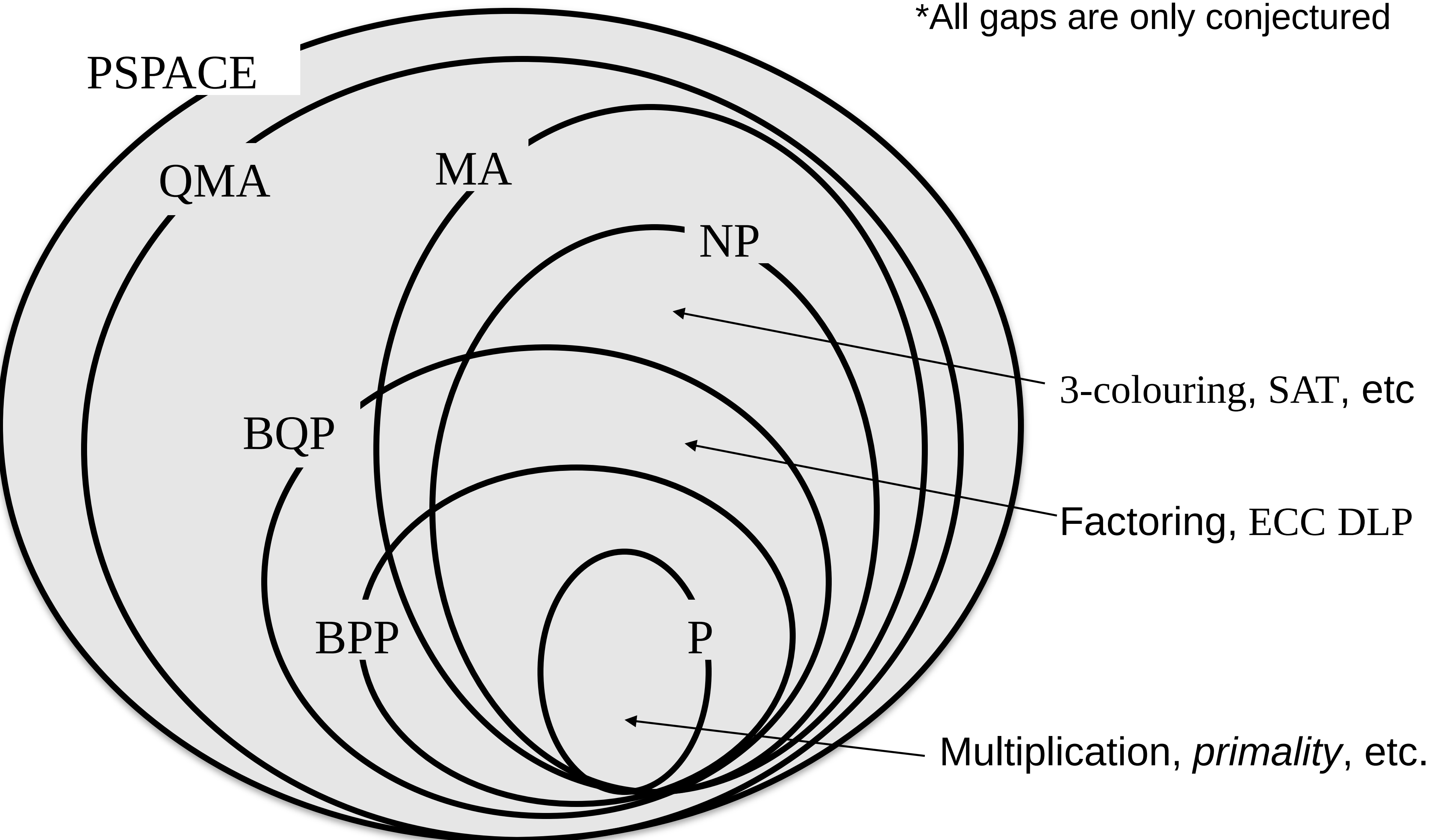
$$If\ \ x \in L \Rightarrow \Pr[V(x, P)\ accepts] \geq \frac{2}{3}$$

$$If\ \ x \notin L \Rightarrow \Pr[V(x, Q)\ accepts] \leq \frac{1}{3}$$

- Big result: IP = PSPACE (Shamir; Lund, Fortnow, Karloff, Nisan)
- Quantum version: *QIP* = PSPACE (Jain, Ji, Upadhyay and Watrous)

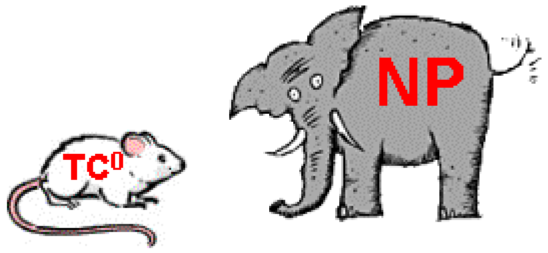
# A micro-zoo of complexity classes
## (see http://www.complexityzoo.com/ for larger zoo)

*All gaps are only conjectured



## http://www.complexityzoo.com/
## http://qwiki.stanford.edu/wiki/Complexity_Zoo

Welcome to the **Complexity Zoo**... There are now 495 classes and counting!



What's your problem?

*Complexity classes by letter:* Symbols - A - B - C - D - E - F - G - H - I - J - K - L - M - N - O - P - Q - R - S - T - U - V - W - X - Y - Z
*Lists of related classes:* Communication Complexity - Hierarchies - Nonuniform
This information was originally moved from http://www.complexityzoo.com/ in August 2005, and is currently under the watchful eyes of its original creators:
**Zookeeper**: Scott Aaronson
**Co-Zookeeper**: Charles Fu
**Veterinarian**: Greg Kuperberg
**Tour Guide**: Christopher Granade