Computability

Hilbert's question: Is it possible to devise an automatic procedure to decide the truth or falsity of mathematical statements?

This led to Alan Turing's work on computability of functions by computing devices.

Example 1:

$$f(x) = \begin{cases} 1 & x \text{ prefix of } \pi, \\ 0 & \text{otherwise.} \end{cases}$$

For example, $f(1) = 0$, $f(314) = 1$, $f(32) = 0$, etc. This function is computable.

Example 2:

$$g(x) = \begin{cases} 1 & x \text{ occurs in } \pi, \\ 0 & \text{otherwise.} \end{cases}$$

For example, $g(1) = 1$, $g(14) = 1$, but what about $g(863898)$?

Example 3:

$$h(x) = \begin{cases} 1 & \text{there is a run of} \geq x \text{ 7's in } \pi, \\ 0 & \text{otherwise.} \end{cases}$$

This function is computable.

Either there are arbitrarily long runs of 7's in which case $h(x) \equiv 1$.

Or, there is a finite maximum run-length m in which case $h(x) = 1$ for $0 \leq x \leq m$, and $h(x) = 0$ for $x > m$.

In all cases, we know that *there exists an algorithm that computes* h.

The Turing Machine

Are all functions computable, or, are there functions that are demonstrably uncomputable?

These questions cannot be answered without first fixing a computational model, in other words, specifying precisely what our computational capabilities are.

The Turing machine model:

- an infinite tape (memory)

- a read-write head (input-output device)

- a finite-state machine control (the program)

The Church-Turing Thesis

The intuitively conceived class of computable functions is precisely the class of functions that are computable by a Turing machine.

This statement cannot be proved because it refers to the undefined notion of ‘intuitive computability.’

The statement is refutable, however; if someone exhibited a more powerful model (that is consistent with the laws of physics) we would abondon this thesis.

The HALTING Problem

If the Turing model is the strongest model that we can think of, can it compute all functions? The answer is ‘no.’ A Turing-uncomputable function is the HALTING function:

$$H(P, x) = \begin{cases} 1 & \text{if } P \text{ halts on input } x, \\ 0 & \text{if } P \text{ runs forever on } x. \end{cases}$$

Quantum Turing Machine

The Turing model is classical in nature.

Could quantum mechanical models compute functions uncomputable by classical models?

The answer is NO. Because any quantum computer can be simulated on a classical Turing machine.

The promise and appeal of quantum computing: Speed up in computations.

Practical Computational Complexity

There are three main types of computational resources:

- time
- memory (space)
- energy.

Time and energy are non-recyclable, memory is recyclable.

Computational complexity: Art of counting computational resources necessary for the solution of a given problem.

Problem Size

Length of a natural encoding of a problem instance measured in bits.

Example: FACTORING an integer N. Problem size is $\log_2 N$ bits.

Example: Sort a database of N records in alphabetical order. Problem size is NK where K is the length of each record in bits.

Decision Problems

Problems with a YES/NO answer.

Example: Given an integer N, is it prime?

More general problems can usually be reduced to decision problems by binary search.

FACTORING as a decision problem: *Does N have a prime factor less than k?*

Polynomial-Time Problems: P

Polytime algorithm: Finds solution in time bounded polynomially in size of problem instance.

P: Class of decision problems solvable by a polytime classical algorithm.

Effectively solvable problems $\subset P$

PSPACE

Decision problems solvable on a classical computer using memory polynomially bounded in the size of the problem.

No constraints on running time.

Obvious: $P \subseteq PSPACE$.

Not known if inclusion is strict.

SATISFIABILITY (SAT)

Given a Boolean expression in conjunctive normal form (CNF), e.g.,

$f = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_1 \vee x_3 \vee x_5)$,

is there a truth assignment so that $f = 1$?

No polytime algorithm known for SAT.

A valid solution is easy to check.

Every YES instance of the SAT problem has a polytime *certificate* that consists of

- a proposed solution (a truth assignment),
- a polytime algorithm for verifying the solution.

A luckily guessing TM could solve SAT in polytime.

Non-Deterministic Polytime: NP

A decision problem belongs to NP if every YES instance of the problem has a polytime certificate.

SAT is in NP.

It is known that

$$P \subseteq NP \subseteq PSPACE.$$

It is not known if any of the inclusions is strict.

Conjecture: $P \neq NP$.

NP-Completeness

Cook (1971) proved that SAT is *NP*-complete.

This means SAT is as hard as any other problem in *NP*.

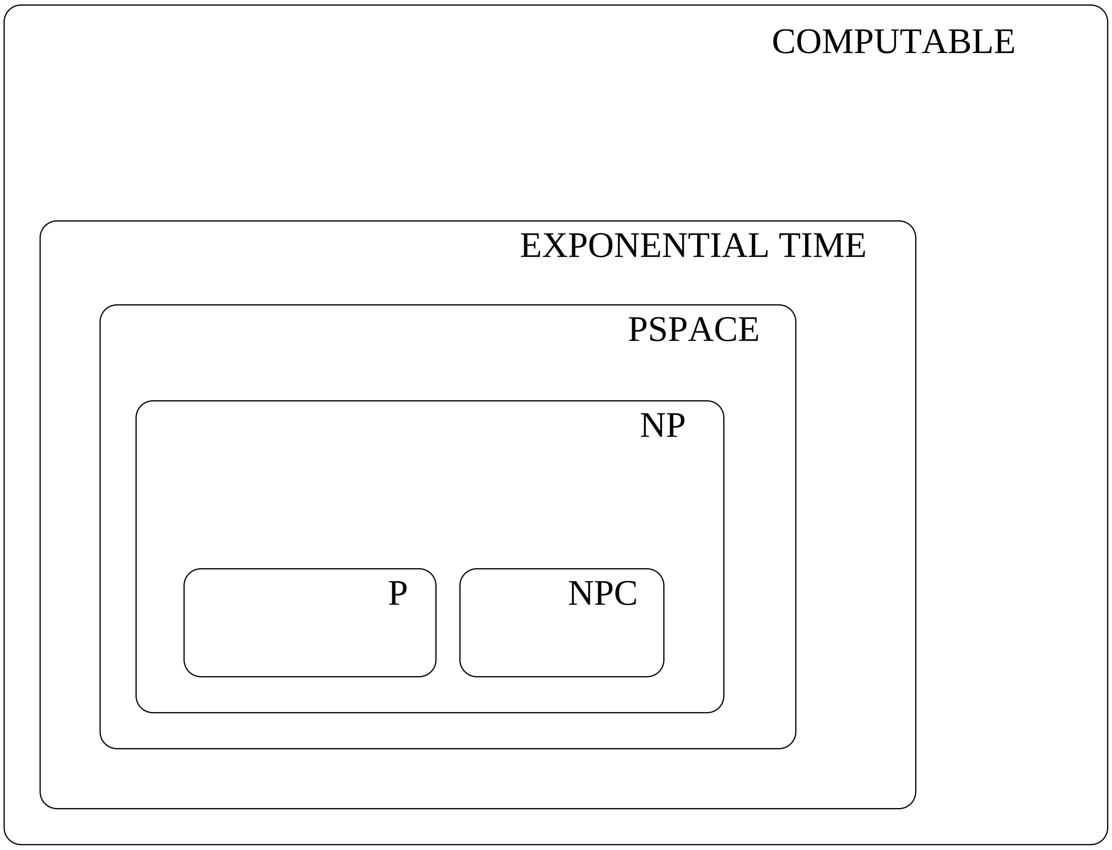
Proof by showing that every problem in *NP* is polytime reducible to SAT.

Corollary: SAT $\in P \Longrightarrow NP = P$.

$NPC =$ The class of *NP*-complete problems.

Conjecture: $NPC \neq P$.

Complexity Hierarchy



Bounded Error Polytime Classical: *BPP*

A randomized algorithm finds the correct solution (YES or NO) in polytime with probability $\geq 3/4$.

PRIMALITY $\in$ *BPP*.

DEUTSCH's problem $\in$ BPP.

Clearly, $P \subseteq BPP$.

The relation of *BPP* to *NP* is unknown.

Bounded Error Polytime Quantum: BQP

A quantum computer solves the problem in polytime with probability of success $\geq 3/4$.

$BQP =$ Class of problems effectively solvable on a quantum computer.

FACTORING $\in BQP$.

Effective Computability

BQP is the quantum analog of BPP.

$BPP \subseteq BPP$

If inclusion strict, then $P \neq PSPACE$.

Likely, but proof must be very difficult.

If $BPP = BQP$, then $P = PSPACE$.

This collapse of classical complexity classes is unlikely, but nothing certain.

Summary: Not yet proven that quantum computers more effective than classical computers even in a practical sense.

BQP $\subseteq$ *PSPACE*

Outline of proof: Classical simulation computes probabilities

$$P(a) = |\langle a|U|0\rangle|^2$$

where U is the quantum computation, 0 is the initial n-qubit state.

$U = U_T U_{T-1} \cdots U_1$ where U_i are intermediate quantum gate operations.

U_i have sparse matrix representations because gates act on bounded number of inputs.

Feynman path integral:

$$\langle a|U|0\rangle = \sum_{x_t} \langle a|U_T|x_{T-1}\rangle \cdots \langle x_1|U_1|0\rangle$$

Compute term by term.

The Circuit Model

Equivalent to the Turing model in computational power.

There are classical and quantum circuit models.

Classical models may be *irreversible* or *reversible*.

Quantum models are necessarily reversible.

Classical Circuits

Any computation can be reduced to computation of Boolean functions

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

We may take $m = 1$ since

$$f = (f_1, \ldots, f_m)$$

where the 'coordinate' functions are of the form

$$f_i : \{0,1\}^n \to \{0,1\}$$

Elementary Gates

$$\mathsf{NOT}(x) = \overline{x} = 1 - x.$$

$$\mathsf{AND}(x, y) = x \wedge y = x \cdot y.$$

$$\mathsf{OR}(x, y) = x \vee y = x + y - x \cdot y.$$

$$\mathsf{AND}(x_1, \ldots, x_n) = \mathsf{AND}(x_1, \ldots, x_{n-1}) \wedge x_n,$$

$$\mathsf{OR}(x_1, \ldots, x_n) = \mathsf{OR}(x_1, \ldots, x_{n-1}) \vee x_n.$$

Computing Boolean Functions

Let f be defined by the truth table:

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Then,

$$f = (\overline{x_1} \wedge \overline{x_2} \wedge x_3) \vee (x_1 \wedge \overline{x_2} \wedge x_2) \vee (x_1 \wedge x_2 \wedge x_3).$$

Every Boolean function can be put in this *disjunctive normal form* (DNF).

Universal Gates

A set of gates *universal* if arbitrary Boolean functions can be synthesized in terms of them.

NOT, OR and AND gates are universal.

NAND (not AND) is universal by itself.

$$\text{NAND}(x, y) = \overline{x \wedge y} = \overline{x} \vee \overline{y}.$$

$$\text{NOT}(x) = \text{NAND}(x, x) = \text{NAND}(x, 1),$$

$$\text{OR}(x, y) = \text{NAND}(\overline{x}, \overline{y}),$$

$$\text{AND}(x, y) = \text{NAND}(\text{NAND}(x, y), 1).$$

COPY Operation

In classical circuits we freely generate copies of bits

$$\mathrm{COPY}(x) = xx$$

In quantum circuits, COPY not allowed because of the *no-cloning* theorem.)

Elements of Classical Circuits

- a number of gates (NOT, AND, OR, etc.),
- n input wires where inputs $x_1, \ldots, x_n$ are applied,
- m output wires where outputs $y_1, \ldots, y_m$ are read,
- ancillary bits 0 and 1 applied as necessary,
- interconnection of gates by a network of wires with permission for fan-out (COPY).

Circuit Complexity

Circuits are memoryless.

Regardless of the depth of the circuit, the operation of a circuit has no measurable time-complexity.

Complexity of a circuit = No of gates.

Compare with Turing model: Hardware fixed, complexity measured by time.

Uniform Circuits

A family of circuits $\{C_n\}$ is *uniform* if design of C_n can be generated by a polytime(n) algorithm on a Turing machine.

Functions f_n can be computed in polytime(n) on a TM iff they can be computed by a uniform family of circuits C_n.

Thus, the class *P* applies to uniform circuit families as well.