Fundamental Algorithmic Techniques XI

December 7, 2025

Outline

Finite and Infinite Programming

Models of Computation

- **Circuits** $\rightarrow$ finite, fixed-size programs (exponential lengths...)
- **Automata** $\rightarrow$ handle infinite inputs/outputs, but limited to regular/simple problems
- **Turing Machine**
  - One per problem
  - Infinite, writable tape
  - Finite set of inner states
  - Transition function/table
- **Universal Turing Machine** Programmable computer!

Turing-Complete Systems

- NAND-TM language
- RAM model (e.g., Python, C)
- Lambda calculus (e.g., Lisp, OCaml, Clojure)
- Cellular automata (e.g., Conway's Game of Life)

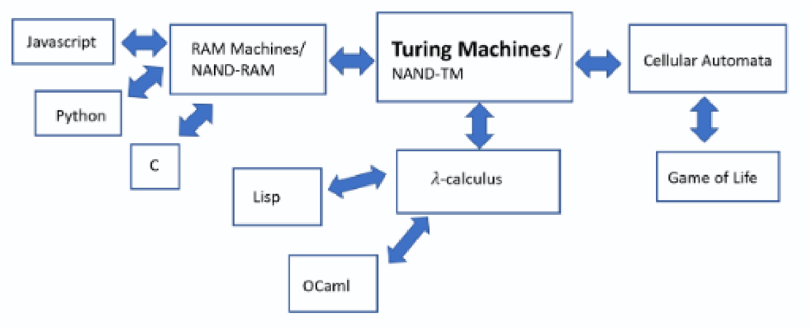
Church–Turing Thesis:

A function on natural numbers is effectively computable

$\Leftrightarrow$

It is computable by a Turing machine.

Turing Complete & Computability



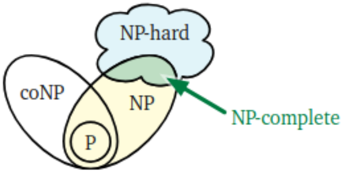
Let's have a closer look at computation types of classes!

Computational Complexity Classes — The Big Picture

Decision problems (answer: Yes/No)

Class	**Meaning**	**Example**
P	Solvable in poly-time	Sorting, shortest path
NP	Verifiable in poly-time	SAT, TSP (decision)
coNP	"No" answers verifiable in poly-time	Formula validity
NP-complete	In NP + NP-hard	3-SAT, Clique
NP-hard	At least as hard as any NP problem	Halting Problem, opt. TSP

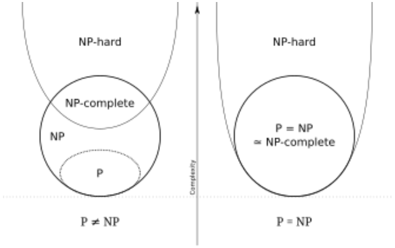
NP-hard, and NP-complete



Relationships among P, NP, NP-complete, and NP-hard classes. One assumes $P \neq NP$ but it is unproven.

- **NP-hard**: A problem Π is NP-hard if a polynomial-time algorithm for Π would imply a polynomial-time algorithm for every problem in NP.
- **NP-complete**: A problem that is both NP-hard and an element of NP.

$P \neq NP$ versus $P = NP$?



Relationships among P, NP, NP-complete, and NP-hard classes.
The standard assumption is $P \neq NP$, but this remains unproven.

If $P = NP$:

- Every efficiently verifiable solution can also be efficiently found.
- **Breakdown** of modern cryptography (e.g., RSA, ECC).
- **Revolution** in optimization, logistics, scheduling.
- **Transformative advances** in AI, machine learning, and automated reasoning.
- Many currently intractable problems become tractable.

Status: unsolved question

Shannon Entropy — Information is Surprise

How much information does a random variable carry?

$$H(X) = -\sum p_i \log_2 p_i \quad \text{(in bits per symbol)}$$

Coin	P(Heads)	$H(X)$
Fair	50%	**1.00 bit** $\leftarrow$ maximum uncertainty
99% heads	99%	≈ 0.08 bit $\leftarrow$ boring
1% heads	1%	≈ 6.6 bits $\leftarrow$ shocking when it lands!

Entropy = average surprise

1 bit = one perfect yes/no question

English text: $H \approx 1$ bit/character $\rightarrow$ 1 MB of text can be compressed to 125 KB (in theory)

The Source Coding Theorem — The Hard Limit

Shannon's Source Coding Theorem (1948):
For a source with entropy $H(X)$ bits/symbol:

- You **cannot** compress below $H(X)$ bits/symbol on average
- You **can** get arbitrarily close — but only for very long messages
- In practice: English ≈ 1 bit/character $\rightarrow$ best possible compression $\approx 12.5\%$ of raw text

Consequences for LLMs training:

Shannon Entropy versus Cross entropy loss: $H(P) = \sum_i p_i \cdot \log(p_i)$ versus $H(P, Q) = \sum_i p_i \cdot \log(q_i)$

- Modern LLMs reach 7–8 bits/token $\rightarrow$ within 10–20% of the theoretical limit.
- Limit for cross entropy loss around 1 (as $Q \rightarrow P$).

Kolmogorov Complexity — The Ultimate Compression

Kolmogorov: "What is the shortest program that outputs this exact string?"

K(x) = length of shortest program that prints x and halts

$\rightarrow$ The **true** information content of one object (not a distribution)

Uncomputable (thanks, Halting Problem)

String	Length	K(x) in bits
`01010101...` (1 million times)	8 MB	$\approx$ 100 bits
`= 3.14159...` (first million digits)	8 MB	$\approx$ 200 KB
`War and Peace`	3 MB	$\approx$ 4–5 MB
Random noise (1 MB)	1 MB	$\approx 8 \cdot 10^6$ bits (incompress!)

Solomonoff — Beautiful Synthesis

Data and programs as bits...

- **Solomonoff Complexity**:
  The **shortest program** that outputs a given string True amount of information in one object
- **Solomonoff Completeness**:
  All computable patterns have short programs & Occam's razor is mathematically provable
- **Solomonoff Induction**:
  Predict the future by weighting all programs by $2^{-\text{length}}$, Provably optimal Bayesian learning But uncomputable (halting problem)

LLMs are gradient descent trying to be Solomonoff induction with 175 billion parameters.