The Classes P, NP and co-NP

George Voutsadakis[1]

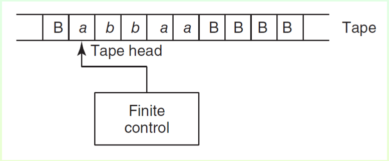
[1]Mathematics and Computer Science
Lake Superior State University

Seminar Presentation

Lake Superior State University

Turing Machines

- A **Turing machine** is a tuple



$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_Y, q_N \rangle,$$

where
  - Q is a finite set, called the set of **states**;
  - Σ is a finite set, called the **input alphabet**;
  - Γ is a finite set, called the **tape alphabet**, which contains Σ and has a special blank character (say B);
  - $q_0 \in Q$ is the **start state**;
  - $q_Y \in Q$ is the **accepting state**;
  - $q_N \in Q$ is the **rejecting state**;
  - $\delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ is the **transition function**.

Operation

- The heart of the Turing machine is its transition function

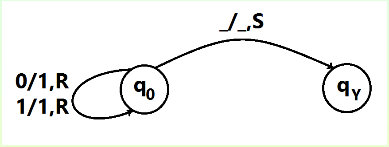
$$\delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{L, R, S\},$$

  which should be thought of as the control (or CPU) module that directs its operation.
- Here is the precise meaning of $\delta(p, x) = (q, y, D)$:

  If the machine is currently in state p and scans symbol x on its tape, then

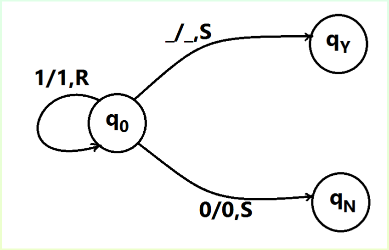
  - it will transition to state q,
  - replace symbol x by symbol y on its tape; and
  - move in direction D (left if $D = L$, right if $D = R$, and stay put if $D = S$).

Examples



- The Turing machine computes the function

$$f(x) = 1^{|x|}.$$



- The Turing machine decides the language

$$L = \{1^n : n \geq 0\}.$$

Turing Machines and Languages

- A **language** L is a set of strings over the alphabet $\{0,1\}$, $L \subseteq \{0,1\}^*$.
  Each string is finite, but a language may have infinitely many strings.
- Examples:
  - The language of palindromes;
  - The language of primes.
- Let M be a Turing machine and x be a string over $\{0,1\}$.
  We write $M(x)$ to denote the "output" of the run of M on input x.
- We say M **decides** the language L if
  - M stops on every input;
  - M accepts a string x if and only if $x \in L$, i.e.,

$$L = \{x \in \{0,1\}^* : M(x) = \mathrm{Y}\}.$$

The Church-Turing Thesis

- **The Church-Turing Thesis:** Virtually any model of digital computation one can define will be equivalent to Turing machines, in the sense that any language decidable by one model will be decidable by the other and vice versa.
- **The Extended Church-Turing Thesis:** Any simulation of one model by another will incur at most a polynomial overhead in both time and memory.
- **Caveats:**
  - The computer must be *classical*, *discrete* and *deterministic* (i.e., cannot be quantum, analog or call a random number generator);
  - There must be *no a priori limit on how much memory* the computer can address (but a program that runs for a finite amount of steps can only address a finite amount of memory).

The Class P

- Let M be a deterministic Turing machine.
- We say that M is **polynomial time** if there exists a polynomial p, such that, for all inputs x, M halts (either accepting or rejecting) after at most $p(|x|)$ steps.
- We denote by P the class of all languages $L \subseteq \{0,1\}^*$ for which there exists a polynomial time Turing machine M deciding L.

The Class NP via NonDeterministic Turing Machines

- A **non-deterministic Turing machine** N is defined like a Turing machine except that, when in a state p and reading a symbol a on its tape, it may transition to a number of possible triples (q, b, X), where q is a new state, a is replaced by b and X determines the head movement.
- We say N **accepts** input x if there *exists* a computation path, among all possible computation paths of N on input x, which accepts x.
- We say that N **decides a language** L if, for all inputs x, L halts on all computation paths on input x, and N accepts x if and only if $x \in L$.
- We say N is **polynomial time** if there exists a polynomial p, such that, for all inputs x, all computation paths of N on input x terminate after at most $p(|x|)$ steps.
- We denote by NP the class of all languages L for which there exists a polynomial time non-deterministic Turing machine deciding L.

The Class NP via Verifiers

- There is an alternative definition of NP defined using deterministic Turing machines provided with additional inputs.
- NP is the class of languages L for which there exists a polynomial time Turing machine V and a polynomial q, such that, for all inputs x,

$$x \in L \quad \text{iff} \quad \text{there exists } w \in \{0,1\}^{q(|x|)}, \text{ such that } V(x,w) = \mathsf{Y}.$$

- V is called a **verifier** for L.
- w is called a V-**witness** for $x \in L$.

Equivalence

- Suppose a non-deterministic Turing machine N decides L in time p.

  Assume that k is the maximum number of possible choices allowed in the transition function of N.

  We construct a deterministic Turing machine V that on input x and $w \in k^{p(|x|)}$ does the following:
  - It interprets w as a sequence of $p(x)$ choices among k possibilities;
  - It then simulates the operation of N for $p(|x|)$ steps, at each step following the choice detailed by the decoding of w.
  - It accepts if and only if the simulated branch of N accepts.
- Suppose verifier V decides L using witnesses of length q in time p.

  We construct a non-deterministic machine N that, on input x:
  - Guesses, using non-determinism, a witness w of length $q(|x|)$;
  - Simulates V on input x, w;
  - It accepts if and only if $V(x, w) =$Y.

$\mathrm{P} \overset{?}{=} \mathrm{NP}$: Formalization

- We trivially have $\mathrm{P} \subseteq \mathrm{NP}$.

  If $L \in \mathrm{P}$, then there exists a polynomial time Turing machine M that decides L.

  But then the Turing machine $V(x, \lambda) = M(x)$ is a polynomial time verifier (with $q = 0$) that accepts L.

  So $L \in \mathrm{NP}$.
- The $\mathrm{P} \overset{?}{=} \mathrm{NP}$ asks whether $\mathrm{NP} \subseteq \mathrm{P}$.
- We conjecture that $\mathrm{P} \neq \mathrm{NP}$ or, equivalently $\mathrm{P} \subsetneq \mathrm{NP}$.

The Class $\mathrm{CO\text{-}NP}$

- For a class C of languages, we define the class $\mathrm{CO\text{-}C}$ to be the class of all languages whose complements relative to Σ^* are in C:

$$\mathrm{CO\text{-}C} = \{L \subseteq \Sigma^* : \overline{L} \in \mathrm{C}\}.$$

- Classes defined by deterministic Turing machines are closed under complements, so, e.g., $\mathrm{P}{=}\mathrm{CO\text{-}P}$.
- On the other hand, because of the asymmetry in the acceptance condition in the case of nondeterministic Turing machines, it does not follow automatically that classes defined by such machines are closed under complements.
- If $\mathrm{NP}{\neq}\mathrm{CO\text{-}NP}$, then $\mathrm{P}{\neq}\mathrm{NP}$, which shows that proving the former is even tougher than proving the latter.
- Another observation is that to show that $\mathrm{NP}{=}\mathrm{CO\text{-}NP}$, it suffices to show that $\mathrm{NP}{\subseteq}\mathrm{CO\text{-}NP}$ (or $\mathrm{CO\text{-}NP}{\subseteq}\mathrm{NP}$).

Thank you!

- In closing...

Thank you for your Attention!!