




Computational Complexity Theory

Lecture 2: Turing machines (contd.); Class P

Department of Computer Science,
Indian Institute of Science

Recap: Turing Machines

- An algorithm is a set of instructions or rules.
- To understand the performance of an algorithm we need a model of computation. Turing machine is one such *natural* model (introduced by Alan Turing in 1936).
- A TM consists of:
 - Memory tape(s)
 - A finite set of rules
- Turing machines  A mathematical way to describe algorithms.

Recap: Turing Machines

- **Definition.** A k -tape Turing Machine M is described by a tuple (Γ, Q, δ) such that
- M has k memory tapes (input/work/output tapes) with *heads*;
- Γ is a finite set of alphabets. (Every memory cell contains an element of Γ)
- Q is a finite set of states. (special states: q_{start} , q_{halt})
- δ is a function from $Q \times \Gamma^k$ to $Q \times \Gamma^k \times \{L, S, R\}^k$



known as **transition function**; it captures the dynamics of M

Recap: TM Computation

- Start configuration.
 - All tapes other than the input tape contain blank symbols.
 - The input tape contains the input string.
 - All the head positions are at the start of the tapes.
 - The machine is in the start state q_{start} .
- Computation.
 - A **step of computation** is performed by applying δ .
- Halting.
 - Once the machine enters q_{halt} it stops computation.

Recap: TM Running time

- Let $f: \{0,1\}^* \rightarrow \{0,1\}^*$ and $T: \mathbb{N} \rightarrow \mathbb{N}$ and M be a Turing machine.
- **Definition.** We say M **computes** f if on every x in $\{0,1\}^*$, M halts with $f(x)$ on its output tape beginning from the start configuration with x on its input tape.
- **Definition.** M computes f in $T(|x|)$ **time**, if for every x in $\{0,1\}^*$, M halts within $T(|x|)$ steps of computation and outputs $f(x)$.

Recap: Turing Machines that halt

- In this course, we would be dealing with
 - Turing machines that halt on every input.
 - Computational problems that can be solved by Turing machines.
- Can every computational problem be solved using Turing machines?

Recap: Uncomputability

- There are problems for which there exists **no** TM that halts on every input instances of the problem and outputs the correct answer.
 - **Input:** A system of polynomial equations in many variables with integer coefficients.
 - **Output:** Check if the system has **integer solutions** .
 - **Question:** Is there an algorithm to solve this problem?
- **Theorem.** There doesn't exist any algorithm (realizable by a TM) to solve this problem. (Davis, Putnam, Robinson, Matiyasevich 1970)

Recap: Why Turing Machines?

- TMs are natural and intuitive.
- **Church-Turing thesis:** *“Every physically realizable computation device – whether it’s based on silicon, DNA, neurons or some other alien technology – can be simulated by a Turing machine”.*
 - [quoted from Arora-Barak’s book]
- Several other computational models can be simulated by TMs.

Recap: Why Turing Machines?

- TMs are natural and intuitive.
- **Strong Church-Turing thesis:** “Every *physically realizable computation device* – whether it’s based on silicon, DNA, neurons or some other alien technology – can be simulated *efficiently* by a Turing machine”.

Possible exception: Quantum machines!

Basic facts about TMs

Turing Machines

- **Time constructible functions.** A function $T: \mathbb{N} \rightarrow \mathbb{N}$ is time constructible if $T(n) \geq n$ and there's a TM that computes the function that maps x to $T(\underbrace{|x|}_{\text{in binary}})$ in $O(T(|x|))$ time.
- Examples: $T(n) = n^2$, or 2^n , or $n \log n$

Turing Machines: Robustness

- Let $f: \{0,1\}^* \rightarrow \{0,1\}^*$ and $T: \mathbb{N} \rightarrow \mathbb{N}$ be a time constructible function.
- Binary alphabets suffice.
 - If a TM M computes f in $T(n)$ time using Γ as the alphabet set, then there's another TM M' that computes f in time $4 \cdot \log |\Gamma| \cdot T(n)$ using $\{0, 1, \text{blank}\}$ as the alphabet set.

Turing Machines: Robustness

- Let $f: \{0,1\}^* \rightarrow \{0,1\}^*$ and $T: \mathbb{N} \rightarrow \mathbb{N}$ be a time constructible function.
- Binary alphabets suffice.
 - If a TM M computes f in $T(n)$ time using Γ as the alphabet set, then there's another TM M' that computes f in time $4 \cdot \log |\Gamma| \cdot T(n)$ using $\{0, 1, \text{blank}\}$ as the alphabet set.
- A single tape suffices.
 - If a TM M computes f in $T(n)$ time using k tapes then there's another TM M' that computes f in time $5k \cdot T(n)^2$ using a single tape that is used for input, work and output.

Turing Machines: As strings

- Every TM can be represented by a finite string over $\{0,1\}$.

...simply encode the description of the TM.

Turing Machines: As strings

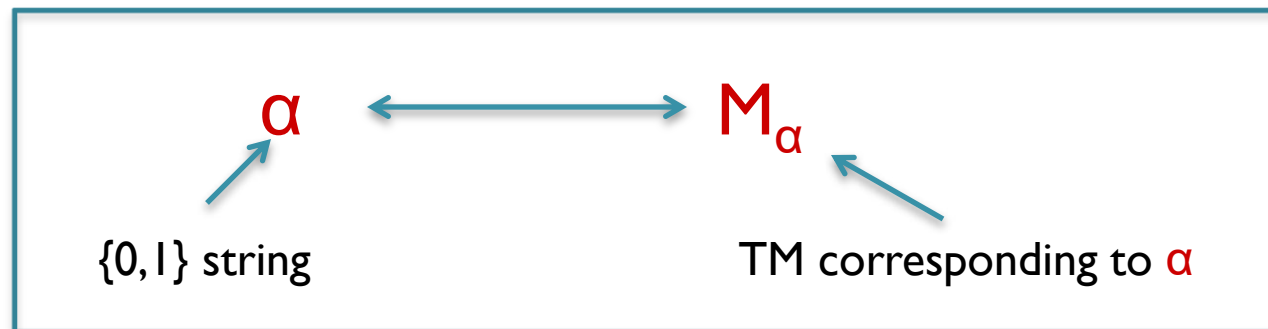
- Every TM can be represented by a finite string over $\{0,1\}$.
- Every string over $\{0,1\}$ represents some TM.
...invalid strings map to a fixed, trivial TM.

Turing Machines: As strings

- Every TM can be represented by a finite string over $\{0,1\}$.
- Every string over $\{0,1\}$ represents some TM.
- Every TM has infinitely many string representations.
 - ... allow padding with arbitrary number of 0's

Turing Machines: As strings

- Every TM can be represented by a finite string over $\{0,1\}$.
- Every string over $\{0,1\}$ represents some TM.
- Every TM has infinitely many string representations.



Turing Machines: As strings

- Every TM can be represented by a finite string over $\{0,1\}$.
- Every string over $\{0,1\}$ represents some TM.
- Every TM has infinitely many string representations.
- A TM (i.e., its string representation) can be given as an input to another TM !!

Universal Turing Machines

- **Theorem.** There exists a TM U that on every input x , α in $\{0,1\}^*$ outputs $M_\alpha(x)$.
- Further, if M_α halts within T steps then U halts within $C \cdot T \cdot \log T$ steps, where C is a constant that depends only on M_α 's alphabet size, number of states and number of tapes.

Universal Turing Machines

- **Theorem.** There exists a TM U that on every input x , α in $\{0,1\}^*$ outputs $M_\alpha(x)$.
- Further, if M_α halts within T steps then U halts within $C \cdot T \cdot \log T$ steps, where C is a constant that depends only on M_α 's alphabet size, number of states and number of tapes.
- Physical realization of UTMs are modern day electronic computers.

Complexity class P

Decision Problems

- In the initial part of this course, we'll focus primarily on **decision problems**.

Decision Problems

- In the initial part of this course, we'll focus primarily on **decision problems**.
- Decision problems can be naturally identified with **Boolean functions**, i.e., functions from $\{0,1\}^*$ to $\{0,1\}$.

Decision Problems

- In the initial part of this course, we'll focus primarily on **decision problems**.
- Decision problems can be naturally identified with **Boolean functions**, i.e., functions from $\{0,1\}^*$ to $\{0,1\}$.
- Boolean functions can be naturally identified with sets of $\{0,1\}$ strings, also called **languages**.

Decision Problems

Decision problems \leftrightarrow Boolean functions \leftrightarrow Languages


- **Definition.** We say a TM M decides a language $L \subseteq \{0,1\}^*$ if M computes f_L , where $f_L(x) = 1$ if and only if $x \in L$.

The characteristic function of L .

Complexity Class P

- Let $T: \mathbb{N} \rightarrow \mathbb{N}$ be some function.
- **Definition:** A language L is in $\text{DTIME}(T(n))$ if there's a TM that decides L in time $O(T(n))$.
- **Definition:** Class $P = \bigcup_{c > 0} \text{DTIME}(n^c)$.

Complexity Class P

- Let $T: \mathbb{N} \rightarrow \mathbb{N}$ be some function.
- **Definition:** A language L is in $\text{DTIME}(T(n))$ if there's a TM that decides L in time $O(T(n))$.
- **Definition:** Class $P = \bigcup_{c > 0} \text{DTIME}(n^c)$.


Deterministic polynomial-time