- Church-Turing thesis:

Every realizable computing device — silicon, DNA, neuron, quantum, alien technology — can be simulated on a TM.

- Problems, solvable on a TM, are called computable, decidable, or recursive.

- Defn.: We say that a fn. $f$ is polynomial-time computable, if there is a TM $M$ computing $f$ in time $T(n)$, where $T(n) = O(n^c)$ for a constant $c \in \mathbb{R}_{\geq 0}$.

- Roughly, we say that $f$ has an efficient algorithm!

- Exercise: Any C-program that runs

efficiently can be converted to an efficient TM, and vice versa.

- Thus, if a problem is hard for TMs then it is also intractable in "real"-life!

- This might change if somebody can build a "real" quantum computer!

## Feeding TMs to themselves

- Since, a TM has a finite description $(\Gamma, Q, \delta)$, we can encode it as a boolean string.

Let us fix some encoding.

- For every string $\alpha \in \{0,1\}^*$, we write

$$M_\alpha := \begin{cases} \text{the TM that } \alpha \text{ encodes, if } \alpha \text{ does encode.} \\ \text{otherwise the TM computing the zero fn.} \end{cases}$$

▷ There is a <u>universal</u> TM $U$ that on input $(\alpha, x)$ simulates $M_\alpha$ on $x$.

– <u>Pf. idea:</u> $U$ reads the transition fn. $\delta$ from $\alpha$ & decides the next step.

## <u>The Halting problem</u>

– $HALT := \{(\alpha, x) \mid M_\alpha$ halts on $x$ & outputs a bit $\}$.

– Its (un) computability was proven by Church (1936) & much simplified by Turing.

<u>Theorem:</u> HALT is not TM computable.

<u>Proof:</u> • Suppose it is decidable by $M_0$.

Say, it outputs 1 on $(\alpha, x)$ if $M_\alpha(x)$ halts, else 0.

- We define a TM $M'$ as:

  On input $\alpha$:

  (1) If $M_0(\alpha, \alpha) = 0$ then <u>output</u> 1.

  (2) Simulate $M_\alpha(\alpha)$ & <u>output</u> the negation of $M_\alpha(\alpha)$.

- Let $\beta$ be an encoding of $M'$. So, $M' = M_\beta$.
- What is $M_\beta(\beta)$?

- If $M_\beta(\beta) = 1$ then,

  either $M_0(\beta, \beta) = 0$, so $M_\beta(\beta)$ should not halt!

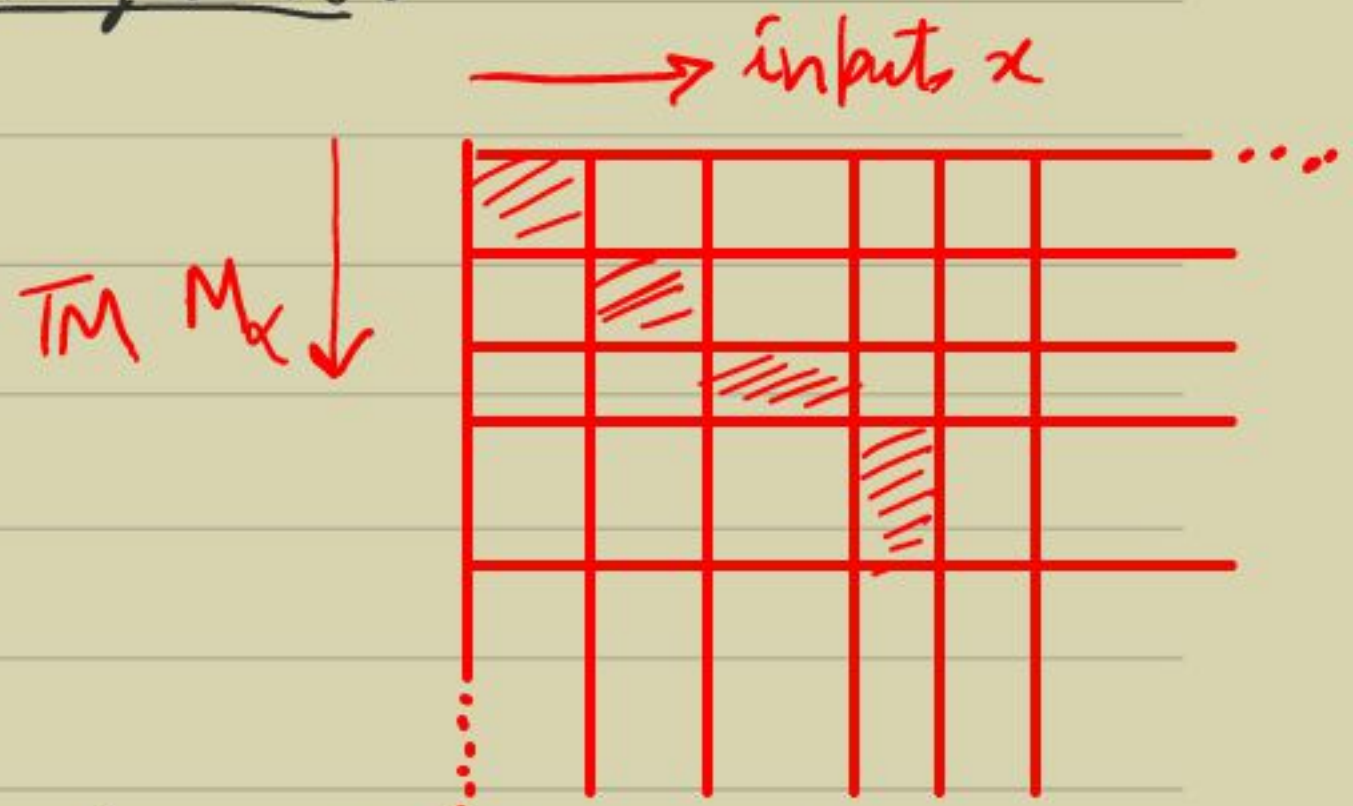
  or $M_\beta(\beta)$ halts & outputs 0!

- If $M_\beta(\beta) = 0$ then,

  $M_\beta(\beta)$ halts & outputs 1!

- In all the three cases we have a contradiction. So $M_0$ does not exist. □

- This proof technique is called <u>diagonalization</u>.




- A TM is designed that disagrees with $M_\alpha(\alpha)$, $\forall \alpha$.

- This technique was used in other foundational results as well:

- <u>Cantor's pf.</u> of the uncountability of the reals (1891). $\mathbb{R} \leftrightarrow 2^{\mathbb{N}}$
- <u>Russell's</u> paradox (1901). $y := \{x \mid x \notin x\}$.
- <u>Gödel's</u> incompleteness theorem (1931).
  $\exists$ unprovable statements

- Reduction:

A function $f$ can be shown uncomputable by reducing the Halting problem to it.

I.e design a computable fn. $f'$ s.t. $\forall x$, $HALT(x) = f \circ f'(x)$.

We then write $HALT \leq_T f$.

# Complexity classes P & NP

- We now study computable functions.

- For a TM $M$, we denote the maximum number of steps, on an input of size $n$, by $time_M(n)$.

- We collect all the problems solvable in time $\approx T(n)$ in a class:

$Dtime(T(n)) := \{L \subseteq \{0,1\}^* \mid TM\ M \text{ decides } L, time_M(n) = O(T(n))\}$.

- As discussed before, we consider poly-time "fast". So,

Defn: • The poly-time solvable problems are

$$P := \bigcup_{c \in \mathbb{N}} \mathrm{Dtime}(n^c).$$

• Formally, P is called the deterministic polynomial time class.

- For practical people poly-time may not always be "fast".

For eg. $g(n) = n^{\log\log n}$ is "faster" than $f(n) = n^{100}$ time, up to input size $n < 2^{2^{100}}$ !!

- Historically, P has served as a rather good abstraction of fast.

- Problems in P? Many interesting examples.

- Let us now consider some practically "hard" problems.

(1) Travelling salesperson :

Let G be a graph on n vertices with edges labelled with "distances". Is there a closed tour, visiting all the nodes exactly once, of length $\leq k$?

(2) Subset sum :

Let S be a set $\{a_1, \ldots, a_n\} \subseteq \mathbb{N}$ & $t \in \mathbb{N}$. Is there a subset of S summing to t?

(3) Integer programming :

Given m linear inequalities in n variables. Is there a boolean feasible point?

Exercise: (1), (2), (3) are easily solved in time $O(n!)$, $O(2^n)$, $O(2^m)$ respectively.

- So, these three problems have algos. that run in <u>exponential-time</u>.

<u>Defn:</u> $\underline{\underline{EXP}} := \bigcup_{c \in \mathbb{N}} Dtime(2^{n^c})$.

- Can these problems be solved any faster?
  Not known!

- They have a common feature:
  Given a closed tour, or subset, or a boolean point; it can be "verified" in poly-time whether it is the right one!

- This feature motivated Cook (1971) & Levin (1973) to study all the problems that have <u>poly-time verifiers</u>.

**Defn:** A language $L \subseteq \{0,1\}^*$ is in **NP** if $\exists c > 0$ & a poly-time TM $M$ s.t.

$\forall x \in \{0,1\}^*$, $x \in L$ iff

$\exists u \in \{0,1\}^{|x|^c}$, $M(x,u) = 1$.

M is called the verifier for L

u is called the certificate for x

- Eg. in (2) the corresponding language is $L = \{(S,t) \mid S$ has a subset of sum $t\}$

On input $(S,t)$:

- The certificate $u$ is the subset of $S$ of sum $t$.
- The verifier $M$ is TM that given $(S,t,u)$ checks whether $u \subseteq S$ & sum of $u$ is $t$.

- Clearly, $(S,t) \in L$ iff $\exists u, M(S,t,u) = 1$.
- Also, $|u| \leq |(S,t)|$ & $M$ is poly-time.

- Thus, $L \in NP$. (Same with (1) & (3)).

- Let us "place" NP :

▷ $P \subseteq NP$.

Pf: • Say, $L \in P$ is decided by a poly-time TM $M$. Then, $\forall x \in \{0,1\}^*$,

$$x \in L \text{ iff } M(x) = 1.$$

• So, with the empty $u$, we can deduce $L \in NP$. □

▷ $NP \subseteq EXP$.

Pf: • Say, $L \in NP$ with the setting:

$x \in L$ iff $\exists u \in \{0,1\}^{|x|^c}$, $M(x,u) = 1$.

• Say, $time_M(n) \leq n^d$.

• On input $x$, we can test $x \in ? L$ by running $M$ on <u>all</u> possible certificates! Output 1 iff one of these time $M$ outputs 1.

• The time-complexity is: $2^{|x|^c} \times (|x| + |x|^c)^d$

• which is, clearly, $O(2^{|x|^{c+1}})$
for $|x|$ large enough.

• $\Rightarrow L \in \text{Dtime}(2^{n^{c+1}}) \subseteq EXP$. □

▷ $P \subseteq NP \subseteq EXP$.

- It is not known whether NP is different from the other two.

- Though, later, we will prove that P & EXP are different!

- The 'N' in NP refers to <u>nondeterminism</u>.

- To study NP we generalize TMs to <u>nondeterministic TMs</u> (NDTM).