From last Time

Proving

$$L_{Halt} = \{ (\alpha, x) : M_\alpha(x) \text{ halts} \}$$

is undecidable

Proof: Reduction to

$$L_{UC} = \{ \alpha : [M_\alpha(\alpha) = 0 \text{ or } M_\alpha(\alpha) \text{ does not halt}] \}$$

→ This is not recognizable

- Suppose L_H is decidable
  - $\exists$ TM M_H that decides

if

$$\Rightarrow M_H(\alpha, x) = \begin{cases} 1 & \text{if } M_\alpha(x) \text{ halts} \\ 0 & \text{if not} \end{cases}$$

- Now: Build $\hat{M}$ which decides $\underline{Luc}$

$$\hat{M}(\alpha) = \begin{cases} 1 & \text{if } M_\alpha(\alpha) = 0 \text{ or does not halt} \\ 0 & \text{if } M_\alpha(\alpha) = 1 \end{cases}$$

↳ Use M_H to build this

$\hat{M}(\alpha) =$

(1) Run $M_H(\alpha, \alpha)$ → $M_\alpha(\alpha)$

② If $M_H(\alpha, \alpha) = 1$ ← halts

2.1 Check if $M_\alpha(\alpha) = 0$
- Output 1

2.2 If not
- output 0

③ If $M_H(\alpha, \alpha) = 0$
output 1

- $\hat{M}$ is a decider for L_{UC}
- But L_{UC} is undecidable □

- Rice's Theorem (informally)
  - All non-trivial properties about programs are undecidable

Efficient Computation

- Complexity classes
  - ↳ Decidable languages, where you can decide within a given resource bound

- Ex: - Space-limited
  - Time-limited
  - Programs where you read the input once
  - ⋮

Time-efficient Computations

- **Def:** Let $T : \mathbb{N} \to \mathbb{N}$ be a function.

  We say a language L is in $\mathrm{DTIME}(T)$ if and only if

  $\exists$ TM M which decides L in time $O(T)$.

$\forall x \in L$: $M(x) = 1$ in time $O(T(|x|))$

$\forall x \notin L$: $M(x) = 0$ in time $O(T(|x|))$.

- All TMs we've seen have been Deterministic

- the Class P poly-time

$$P = \bigcup_{c \in \mathbb{N}} DTIME(n^c)$$

↳ this is efficient

Example Problems in P

- Int mult: $x \cdot y = z$?
- are x, y relatively prime?
- Graph Connectivity

- Directed Graph Paths: does there exist path from $s \rightarrow t$?
- Does $A\underline{x} = b$?

- Church-Turing Thesis
  - Every physically realizable computational model can be simulated by a TM.
  - Strong: simulation has only polynomial overhead
    - → X

Criticisms:

① why polynomial time?

- n^{100} algo? $n = 2$, 2^{100}

- polynomials capture composition

$p(x) + q(x)$ is polynomial

(poly, poly)

$p(q(x))$ is polynomial

- historic/heuristic reasons

Ex. Someone gave n^{100} algo

→ someone else improves it to n^5

- TMs simulate other models in poly-time

- P is worst-case
  - You can have a problem where for 99% of input, it's time n
  - 1% time n^{17}

Efficient Verification

Ex: Prime-factors

- Given large number $N \in \mathbb{N}$, find all prime factors $p_1 \cdots p_k$ of N.

- Efficient verification
  - I give you $p_1', \ldots, p_k'$
  1. Check if $N = p_1' \cdot p_2' \cdot \ldots \cdot p_k'$
  2. Check if all $p_1', \ldots, p_k'$ are prime

$\in P$

Verifiable Languages

A language L is verifiable if $\exists$ a TM M_L such that

$x \in L \iff$

$\exists w \in \{0,1\}^*$ such that $M_L(x, w) = 1$

w: witness / certificate

L is efficiently verifiable if $\exists$ TM M_L and polynomial p such that

$x \in L \iff \exists w \in \{0,1\}^{p(|x|)}$ s.t.

$M_L(x, w) = 1$

The Class NP

$NP = \{$ all efficiently verifiable languages $\}$

Q: Is $P \subseteq NP$?

A: Yes! why?

If $L \in P$

$\exists M_L$ s.t. M_L decides L

$\Rightarrow \forall x : M_L(x) = 1$ if $x \in L$

$M_L(x) = 0$ if $x \notin L$

in polynomial time

→ Verifier Machine

$\hat{M}_2(x, \varepsilon) \rightarrow$

Run $M_L(x)$

Central Problem

$P \stackrel{?}{=} NP$

Widely believed that

$P \neq NP$

Alternate NP characterization

- Non-deterministic TMs

Def: A k-tape non-det. Turing Machine is identical to a Det. TM, except for the following

① $\delta : Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^{k-1} \times \{L, R, S\}^k)$

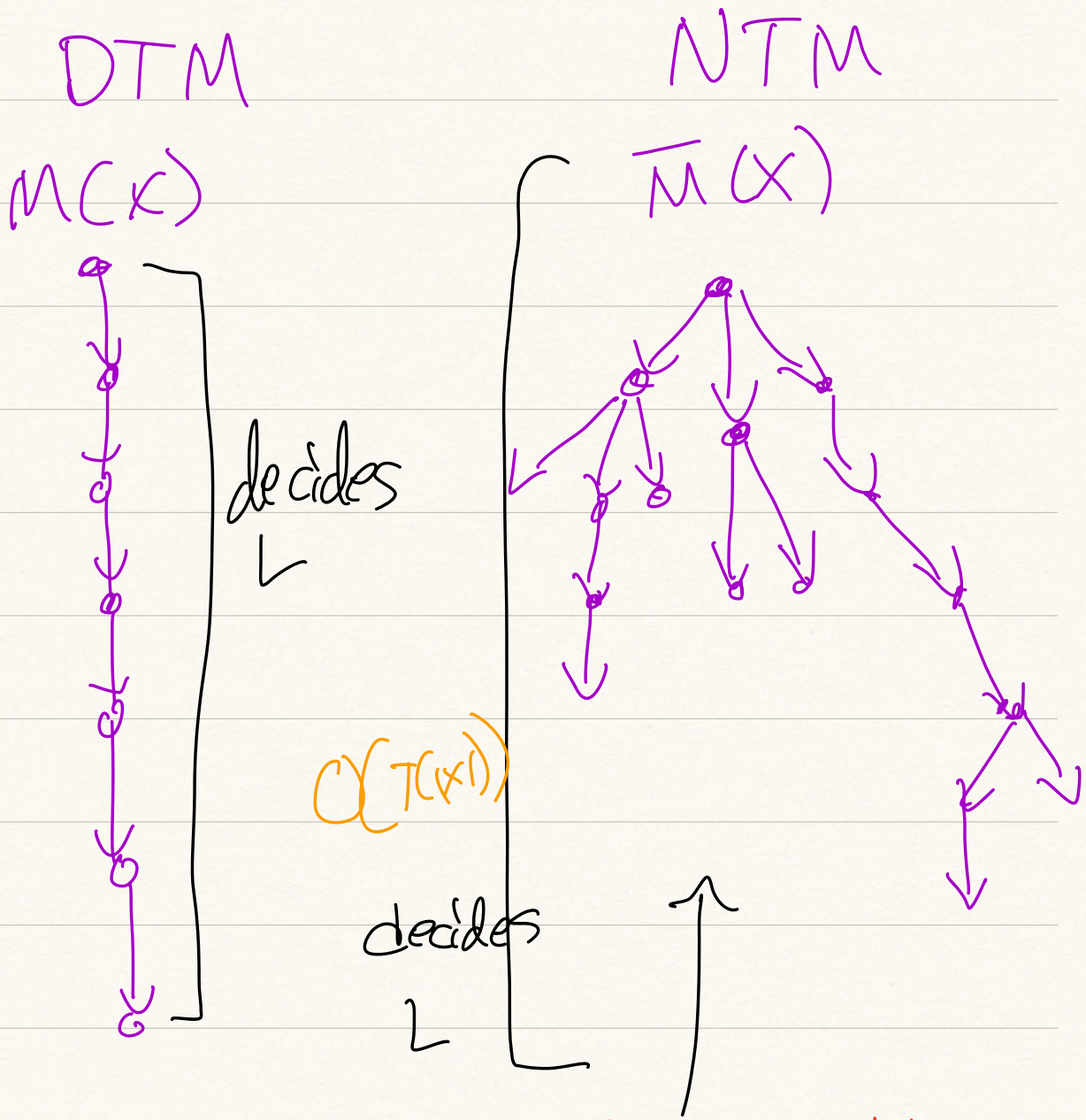
$|Q| \cdot |\Gamma|^{k-1} \cdot 3^k$

for DTM

$\mathcal{P}(S)$ = all possible subsets of S

$|\mathcal{P}(S)| = 2^{|S|}$

② The TM non-det. chooses next configuration



- All execution branches halt
- $\exists$ at least one branch

that accepts $x \in L$

- For $x \notin L$, all branches reject

$NTIME(T) =$
{ all languages decided by a time $O(T)$ NTM }

longest execution branch has $O(T)$ length

Lemma:

$$NP = \bigcup_{c \in \mathbb{N}} NTIME(n^c)$$

Ex Prime factors

- Given N, find k-prime factors of N

NTM $M^*(N)$

- Pick k prime numbers $p_1, \ldots, p_k$ (nondet part)
- Check if $N = p_1 \cdot \ldots \cdot p_k$

- $EXP = EXPTIME = \bigcup_{c \in \mathbb{N}} DTIME(2^{n^c})$

- **Lemma:**
  $NP \subset EXP$

Proof: Enumerate all possible branches of the NTM, check for accept/reject

If NTM ran in time $T(n)$, then this runs in time $O(2^{T(n)})$ $\square$