P and NP

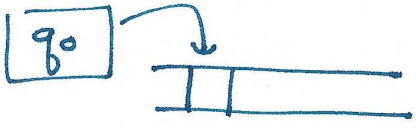
Deciding in polynomial time — Verifying in polynomial time

$P \subseteq NP$

Big question Is $P = NP$?

If language L can be verified in polynomial time, can L be decided in polynomial time?

Non-determinism



Def: A non-deterministic TM has a non-deterministic transition function $\delta : Q \times (\Sigma \cup \{\_\}) \to 2^{(Q \times (\Sigma \cup \{\_\}) \times \{L,R\})}$

The running time is an upper bound on all computational paths.

Def: $L \in NTIME(t(n))$, $\exists$ a standard NTM and constant c s.t. M decides L and runs in $\leq c\, t(n)$ time.

Verification: NP

If $x \in L$, $\exists$ certificate y s.t. $M(x,y)$ accepts

If $x \notin L$, $\forall$ cert. y $M(x,y)$ rejects

Non-determinism: NP

If $x \in L$, $\exists$ computational path where $M(x)$ accepts.

If $x \notin L$, $\forall$ comp. paths $M(x)$ rejects

Clm: $NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$

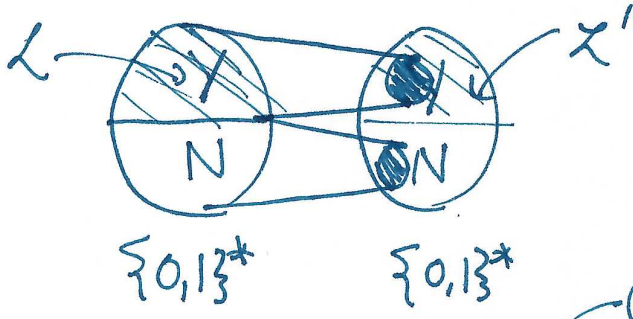
Efficient Reductions & NP-completeness

Def: A reduction from language $\mathcal{L}$ to $\mathcal{L}'$ is a computable mapping/function $f: \{0,1\}^* \to \{0,1\}^*$ s.t.

If $x \in \mathcal{L}$, $f(x) \in \mathcal{L}'$

If $x \notin \mathcal{L}$, $f(x) \notin \mathcal{L}'$

(Karp reduction, many-one reduction)

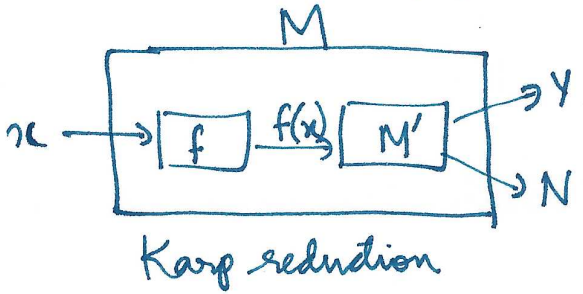


A reduction is efficient if f can be computed in polynomial time (poly time).

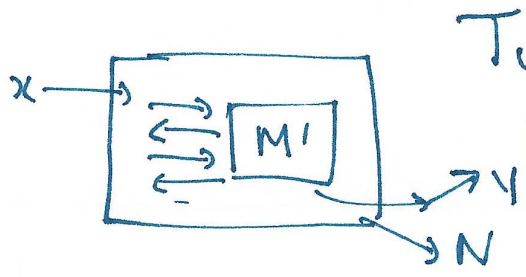
Denoted as $\mathcal{L} \leq_{p(n)} \mathcal{L}'$

Suppose M' decides $\mathcal{L}'$.

We want M that decides $\mathcal{L}$.



Karp reduction



Turing Reduction

Suppose ~~$\mathcal{L} \in$~~ $\mathcal{L} \leq_{p(n)} \mathcal{L}'$ (f computation can be done in n^c time) and $\mathcal{L}' \in \mathbb{P}$. Then $\mathcal{L} \in \mathbb{P}$. ~~$\exists M$~~

M' runs in n^d time

The ~~running~~ running time of M is

$$n^c + (n^c)^d$$

(n^c: compute f; $(n^c)^d$: size of $f(x)$; whole: Run of $M'(f(x))$)

Polynomial

$\mathcal{L} \leq_{p(n)} \mathcal{L}'$

I prove

Deciding $\mathcal{L}$ efficiently (is $\mathcal{L} \in \mathbb{P}$?) is longstanding open problem.

I want to decide $\mathcal{L}'$ efficiently.

(A) I have proof that $\mathcal{L}'$ cannot be decided efficiently

(B) I should give up trying to decide $\mathcal{L}'$ efficiently.

(C) I have learned nothing about $\mathcal{L}'$, so I should continue working on this.

Def: $\mathcal{L}$ ~~$\subseteq$~~ # is NP-complete if (Hardest problem in NP.)

(1) $\mathcal{L} \in \mathrm{NP}$

(2) $\forall\, \mathcal{M} \in \mathrm{NP}$, $\mathcal{M} \leq_{p(n)} \mathcal{L}$

(If only (2), $\mathcal{L}$ is NP-hard.)

Lemma: (hard)
If $\mathcal{L}$ is NP-complete & $\mathcal{L} \in \mathbb{P}$, $\mathbb{P} = \mathrm{NP}$

$L = \{ \langle M, x, 1^t \rangle \mid$ M is a NTM that runs in t steps and accepts $x \}$

↑ encoding of

$L \in NP$ (simulation)

Exercise: $\forall M \in NP$, $M \leq_{p(n)} L$

L is NP-complete

Clm: If L is NP-complete and $L \leq_{p(n)} L'$, $L' \in NP$

L' is NP-complete.

Closure of polynomials under composition

The Cook-Levin Theorem

Thm [Cook 71, Levin 72]: SAT is NP-complete.

A Boolean formula is an expression with Boolean variables $\{x_1, x_2, \ldots, x_n\}$ & operators $\{\overline{\vee}, \vee, \wedge\}$ x_i or $\sim x_i$, called LITERALS.

$$\phi = \left(x_1 \vee \sim(x_3 \vee x_5)\right) \wedge \sim(x_5 \vee x_{10}) \wedge \ldots$$

ϕ is satisfiable if $\exists$ setting of variables making the formula true.

$SAT = \{ \langle \phi \rangle \mid \phi$ is satisfiable$\}$

Proof: We argued SAT $\in$ NP. (Certificate is satisfying assignment.)

$\forall\, \mathcal{M} \in NP,\ \mathcal{M} \leq_{p(n)} SAT.$

Given We will describe a poly-time computable function that maps strings to Boolean formulas.

$$f : \{0,1\}^* \longrightarrow \{\langle\phi\rangle \mid \phi \text{ is formula}\}$$

$x \in \mathcal{M}$ iff $f(x)$ is satisfiable.

standard

There is an NTM M that decides $\mathcal{M}$ and runs in at most n^c steps (c is constant).

Tableau method $\quad n = |x|$

Configuration string: at any time (in the running of M on x), take the snapshot of

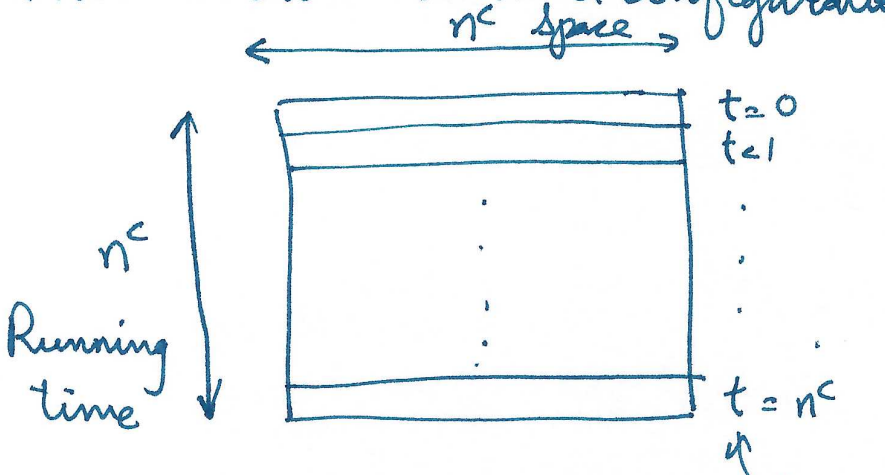
(1) State (2) Position of Head (3) Content of tape

constant $\quad \leq n^c \quad \leq n^c$

String $\quad \sigma_1 \sigma_2 - - \sigma_k \,|\, q \,|\, \sigma_{k+1} - - \sigma_\ell \qquad \ell \leq n^c$

Computation is JUST the evolution of configuration strings.

Tableau is a matrix where each row is a configuration string



Let $Y_{t,k}$ be a variable in $(\Sigma \cup Q \cup \{\_\})$

Symbol at k^{th} position of t^{th} config. string

$x \in A_M$ iff there is a setting of tableau such that

(1) First config is the input x on tape and start state.

(2) Last config is accepting (state is accept state)

(3) Each config is a valid config. string (only one symbol is a state)

(4) Each intermediate config. follows from previous configuration using a valid transition of M.

---

(1) START: $Y_{0,0}$ is initial state and $Y_{0,k}$ is k^{th} input symbol (of x)

(2) END: One of $Y_{n^c,k}$ is an accepting state.

(3) VALID: For each t, ~~only a~~ exactly one of $Y_{t,k}$ is a state.

(4) MOVE: $\forall t$ & $Y_{t,*}$ follows from $Y_{t-1,*}$ using transition fn.

Construct Boolean formula from tableau

$\forall\ t \le n^c,\ \forall\ k \le n^c,\ \forall\ \alpha \in \Sigma \cup \{\sqcup\} \cup Q$

$$y_{t,k,\alpha} = \begin{cases} 1 & \text{if } Y_{t,k} = \alpha \\ 0 & \text{else} \end{cases}$$

For fixed, t,k, only one of $y_{t,k,\alpha}$ is true

UNIQUE: $\bigwedge_{\substack{t \le n^c \\ k \le n^c}} \left[\left(\bigvee_{\alpha \in \Sigma \cup \{\sqcup\} \cup Q} x_{y_{t,k,\alpha}} \right) \wedge \left(\bigwedge_{\substack{\alpha,\beta \\ \alpha \neq \beta}} \left(\sim y_{t,k,\alpha} \vee \sim y_{t,k,\beta} \right) \right) \right]$

(exactly)

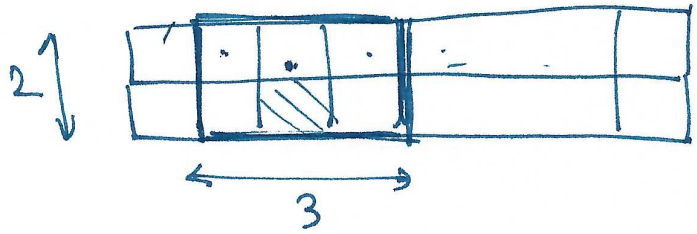
x

Size of formula is $O(n^{2c})$

START: $y_{00q_0} \wedge y_{01x_1} \wedge y_{02x_2} \wedge y_{03x_3} \cdots \wedge y_{0n^c x_n} \wedge y_{0(n+1)\sqcup} \cdots$

Size is $\le n^c$

MOVE:

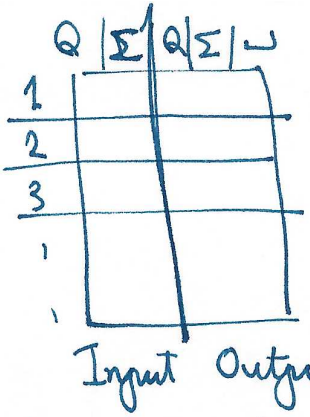


$\forall\ t \le n^c,\ k \le n^c$

If $Y_{t,k-1}, Y_{t,k}, Y_{t,k+1}$ are all NOT state symbols,

then $Y_{t+1,k} = Y_{t,k}$

$\delta : Q \times (\Sigma \cup \{\triangleright, \sqcup\}) \to 2^{Q \times \{\leftarrow, \rightarrow\}}$



δ as a table

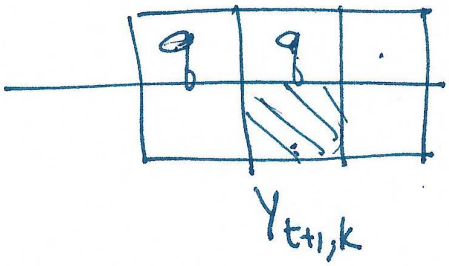
$Z_{t,i} = \begin{cases} 1 & \text{if the } i^{th} \text{ rule is used in } t^{th} \text{ step} \\ 0 & \text{else} \end{cases}$

(Ensure that $\forall\, t$, $\exists$ unique i, s.t. $Z_{t,i} = 1$)

$\forall\, t \le n^c \quad \forall\, i \le$ size of transition table

$Z_{t,i} \implies \left(\bigwedge_{k \le n^c} (Y_{t,k-1}, Y_{t,k}, Y_{t,k+1}) \text{ determine } Y_{t+1,k} \right)$

(i^{th} rule chosen in t^{th} step)



i^{th} rule of transition table

Overall, we can write down (in poly time) a boolean formula ϕ_x of size $O(n^{2c})$ that encodes the tableau.

$x \in M$ iff $\langle \phi_x \rangle$ is satisfiable.

SAT is NP-complete