$\forall k \in \mathbb{N} \quad DTIME(n^k) \subsetneq DTIME(n^{k+1}) \qquad \mathbb{P} \neq \mathbb{EXP}$

$$\mathbb{P} = \bigcup_{k \geq 0} DTIME(n^k) \qquad DTIME(2^n)$$

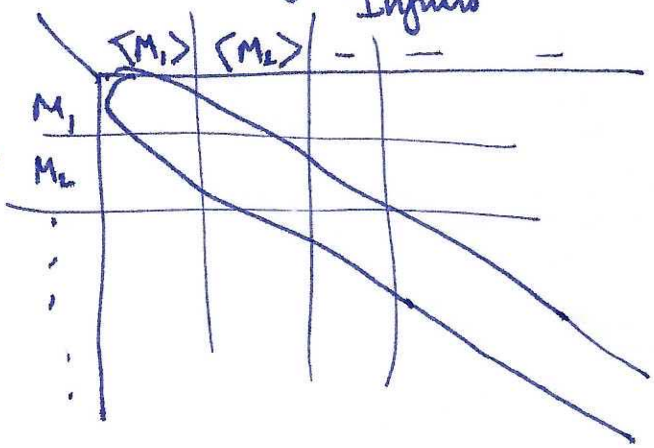
$$DTIME(n^k) \subsetneq DTIME(2^n)$$ (Time hierarchy thm)

$$\bigcup_{k \geq 0} DTIME(n^k) \subsetneq DTIME(2^n) \quad \times$$

$S_n = \{ i \in \mathbb{N} \mid i \leq n \}$

$\forall n \; S_n \subsetneq \mathbb{N}$ But $\bigcup_n S_n = \mathbb{N}$

Need to use the fact that $\mathbb{P}$ is a countable union.



All TMs that run in poly time

Construct a machine that runs in DTIME (2^n) that can "flip" diagonal.

Consider pairs $\langle M_i, k \rangle$ where M_i runs in time n^k.

Imagine a string to be code AND a natural number.

By countability, we can "order" these pairs

Input

	$\langle M_1, k_1 \rangle$	$\langle M_2, k_2 \rangle$	—	—
M_1				
M_2				
$\vdots$				

D Our "diagonal" machine (on input $\langle M_i, k\rangle$)

(1) Runs simulation M_i on input $\langle M_i, k\rangle$ for $|\underbrace{\langle M_i, k\rangle}_{n}|^k$ steps.

(2) If terminates, flip output. Else reject.

D runs in time $O(2^n)$ and we can argue to D is NOT any M_i (where M_i runs in poly time)

Suppose $L(D) \in P$.

$\exists$ Some $M_i \in DTIME(n^k)$ s.t. $L(D) = L(M_i)$

$D(\langle M_i, k\rangle) \neq M_i(\langle M_i, k\rangle)$ Contradiction

---

Beyond NP : the polynomial hierarchy

PH

CLIQUE $= \{ \langle G, k\rangle \mid G$ has a clique of size $\geq k \} \in NP$

EXACT-CLIQUE $= \{ \langle G, k\rangle \mid$ the largest clique in G has size exactly $k \}$

Can we design certificates for $\langle G, k\rangle \in$ or $\notin$ EXACT-CLIQUE?

$\langle G, k\rangle \in$ EXACT-CLIQUE : how do we certify that there isn't a larger clique than k?

$\langle G, k\rangle \notin$ EXACT-CLIQUE : (1) Either $\exists$ clique larger than k $\rightarrow$ NP

OR

co-NP $\leftarrow$ (2) All cliques are $< k$.

If there is a procedure that decides CLIQUE, then EXACT-CLIQUE can be solved by at most n calls to the $O(\log n)$ procedure.

<u>Circuit generation</u>: What is the smallest circuit to multiply two n-bit integers?
add/

encoding length

SMALLEST-FORMULA $= \{ \langle \Phi \rangle \mid \Phi$ is the smallest formula computing a fn $\}$

$$\underbrace{\forall \Phi' \ \exists x}_{\text{2 quantifiers}} \text{ st. } \left[\text{If } |\langle \phi' \rangle| < |\langle \Phi \rangle|, \ \boxed{\Phi(x) \neq \Phi'(x)} \right]$$

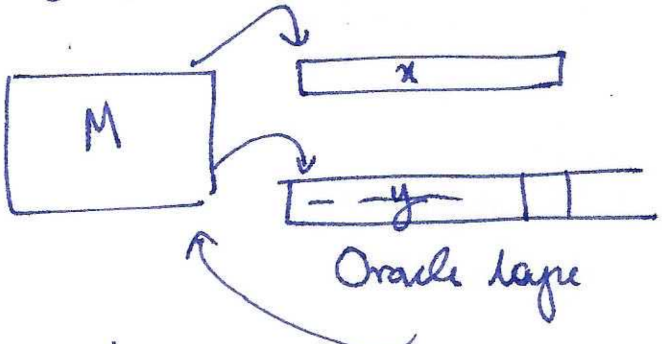
If Φ' is smaller, then $\Phi \neq \Phi'$

Even if we can solve SAT (to decide if $\Phi = \Phi'$), we still need to consider all Φ'.

<u>Oracles</u>

~~Let A be~~ Let $\mathcal{L}$ be a language.

$\mathbb{P}^{\mathcal{L}} \leftarrow$ poly-time machine with an $\mathcal{L}$-oracle



<u>Is $y \in \mathcal{L}$?</u>

With access to a subroutine that decides $\mathcal{L}$ in UNIT time (1 time step).

$\mathbb{NP}^{\mathcal{L}}$: non-deterministic poly-time machines with an $\mathcal{L}$ oracle

SMALLEST-FORMULA $\in$ co-$(\mathbb{NP}^{SAT})$

When $\langle \phi \rangle \notin$ SMALLEST-FORMULA, there is a certificate that can be polynomially verified with access to a SAT oracle.

$\exists \underline{\langle \phi' \rangle}$ s.t. $\underbrace{\phi' = \phi}_{\text{SAT oracle}}$

$\mathbb{C}$ is a complexity class

$\mathbb{NP}^{\mathbb{C}} \leftarrow$ non-det. poly time machines with oracles to languages in $\mathbb{C}$

(It suffices to have oracle for a $\mathbb{C}$-complete language).

$\Pi_2^{p} = \text{co-}(\mathbb{NP}^{\mathbb{NP}})$

$p \leftarrow$ poly

$2 \leftarrow$ # alternations of quantifiers

<u>Def 1</u>: (Oracle machines) $\Sigma_1^p = \mathbb{NP}$ $\quad \Pi_1^p = \text{co-}\Sigma_1^p = \text{co-}\mathbb{NP}$

$$\Sigma_{i+1}^p = \mathbb{NP}^{\Sigma_i^p} \qquad \Pi_{i+1}^p = \text{co-}\Sigma_{i+1}^p$$

Def 2: (Alternating certificate viewpoint)

$\mathcal{L} \in \Sigma_i^p$ if $\exists$ poly-time machine M and a polynomial q

s.t. $x \in \mathcal{L}$ iff $\exists u_1 \forall u_2 \exists u_3 \dots u_i$ s.t. $M(x, u_1, u_2, \dots, u_i)$ accepts

where all $|u_i| \leq q(|x|)$

$\Pi_i^p = \text{co-}\Sigma_i^p$

$$\mathbb{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^p$$

NOT-SMALLEST-FORMULA $= \{ \langle \Phi \rangle \mid \Phi$ is not smallest formula computing a function $\}$
$\mathcal{L}$

$M(\Phi, \Phi', x)$ accepts if $\Phi(x) = \Phi'(x)$.
and $|\Phi'| < |\Phi|$
$\searrow$ poly time

$\Phi \in \mathcal{L}$ iff $\exists \Phi' \forall x$ $M(\Phi, \Phi', x)$ accepts
$\searrow \in \Sigma_2^p$ Π_i-SAT $\forall u_1 \exists u_2 \dots$

Def: Σ_i-SAT is language of formulas $\Phi(u_1, u_2, \dots, u_i)$

$\langle \Phi \rangle \in \Sigma_i$-SAT if $\exists u_1 \forall u_2 \dots u_i \, [\Phi(u_1, \dots, u_i)]$ is true

Special case of QBF

switching quantifiers $\underline{i}$ times

Proof: We prove by induction on i (let's use Def 2 for Σ_i^p)

Base case ($i=1$): SAT is NP-complete (Cook-Levin Thm)
TAUT is (co-NP-complete).

Induction: Assume for i. Consider $L \in \Sigma_{i+1}^p$.

We need to prove that $L \leq_p \Sigma_{i+1}$SAT

Using defn of Σ_i^p (Def 2):

$\exists$ TM M that runs in poly time and polynomial q s.t.

$x \in L$ iff $\exists u_1 \forall u_2 \cdots u_{i+1}$ $M(x, u_1, u_2 \cdots, u_{i+1})$ accepts
$(|u_i| \leq q(|x|))$

$$L' = \{ \langle x, u_1 \rangle \mid \underbrace{\forall u_2 \exists u_3 \cdots u_{i+1}}_{i \text{ alternations}} M(x, u_1, u_2 \cdots u_{i+1}) \text{ accepts} \}$$

$L' \in \Pi_i^p$. By induction $L' \leq_p \Pi_i$-SAT.

$\exists$ exists poly-time "reducer" R that given $\langle x, u_1 \rangle$ as input, ~~computes~~ outputs formula $\Phi_{x,u_1}(v_1, v_2 \cdots, v_i)$
s.t. $\langle x, u_1 \rangle \in L'$ iff $(\Phi_{x,u_1}) \in \Pi_i$-SAT.

$x \in L$ iff $\exists u_1$ s.t. $\langle x, u_1 \rangle \in L'$

iff $\Phi_{x,u_1} \in \Pi_i$-SAT

$x \in L$ iff $\exists u \forall v_1 \exists v_2 \ldots v_i$ s.t. $\{\Phi_{x,u}(v_1, v_2, \ldots, v_i)$ is true$\}$ ~~are~~

Quantification over formula (Function of x, u)

Given x, in poly time

We want to compute $\psi(z_1, z_2, \ldots, z_{i+1})$ s.t.

$x \in L$ iff $\psi(z_1, \ldots z_{i+1}) \in \Sigma_i\text{-SAT}$

$\exists z_1 \forall z_2 \ldots z_{i+1}\ \psi(\ldots)$ is true

ψ is a fn. of x