PH

Def 1 (Oracle machines)

$\Sigma_1^P = NP$ $\quad$ $\Pi_1^P = co\text{-}NP$

$\forall\, i > 1$ $\quad$ $\Sigma_i^P = NP^{\Sigma_{i-1}^P}$ $\quad$ $\boxed{\Pi_i^P = co\text{-}\Sigma_i^P}$

Def 2 (Alternating certificate viewpoint):

$L \in \Sigma_i^P$ if there exists a polynomial q and a polynomial time TM M s.t.

$x \in L$ iff $\exists u_1 \forall u_2 \exists u_3 \cdots$ ~~$M(x,$~~ u_i s.t. $\forall j \leq i$ $|u_i| \leq q(|x|)$ & $M(x, u_1, u_2 \cdots, u_i)$ accepts

(i switches quantifiers)

Def 3: Σ_i-SAT is the language of formulas $\Phi(u_1, u_2 \cdots u_i)$ where

($u_1, u_2 \cdots u_i$: Blocks of inputs)

$\langle \Phi \rangle \in \Sigma_i\text{-SAT}$ if $\exists u_1 \forall u_2 \cdots u_i \left[\Phi(u_1, u_2 \cdots u_i) \text{ is true} \right]$

(switching quantifiers)

Π_i-SAT is ~~same~~ analogous with first quantifier being $\forall$

Thm: Σ_i-SAT is Σ_i^P-complete.

Thm: Def 1 & Def 2 are equivalent

Proof: $\Rightarrow(\Leftarrow)$ Def 2 implies Def 1. We prove by induction over i

Base case is obvious

For $\mathcal{L}$, $\exists$ polynomial q and poly time TM M

s.t. $x \in \mathcal{L}$ iff $\exists u_1 \forall u_2 \cdots$ (i switches) s.t. $\forall\, j \le i \;\; |u_j| \le q(|x|)$ & $M(x, u_1, u_2 \cdots u_i)$ accepts

Consider language $\mathcal{L}' = \{ \langle x, u_1 \rangle \mid \forall u_2 \exists u_3 \cdots u_i : \forall 1 \le j \le i \; |u_j| \le q(|x|) \;\&\; M(x, u_1, u_2 \cdots u_i) \text{ accepts} \}$

$\mathcal{L}' \in \overline{\Pi\Pi}_{i-1}^{P}$ because $\exists$ poly q and a TM* M' s.t. (polytime)

$\langle x, u_1 \rangle \in \mathcal{L}'$ iff $\forall u_2 \cdots, u_i \cdots \; M'(\langle x, u_1 \rangle, u_2 \cdots u_i)$ accepts

(M' parses 1st input into x & u_1 and then runs M.)

$\overline{\mathcal{L}'} \in \sum_{i-1}^{P}$

Consider machine D (on input x)

1. D non-deterministically guess a string u of length $\le q(|x|)$.
2. Writes down $\langle x, u \rangle$ on oracle tape and asks "is $\langle x, u \rangle \in \overline{\mathcal{L}'}$"?
3. Reverse the output

Clm: The language of D is precisely $\mathcal{L}$.

($\Rightarrow$) Consider the language of D. x is accepted if $\exists\, u$ $(|u| \leq q(|x|))$ s.t. $\langle x,u\rangle \notin \overline{\mathcal{L}'}$

x is accepted if $\exists\, u$ s.t. $\langle x,u\rangle \in \mathcal{L}'$.

By definition of $\mathcal{L}'$, x is in $\mathcal{L}$.

($\Leftarrow$) If $x \in \mathcal{L}$, $\exists\, u_1, |u_1| \leq q(|x|)$ s.t. $\langle x,u_1\rangle \in \mathcal{L}'$
$\equiv \langle x,u_1\rangle \notin \overline{\mathcal{L}'}$

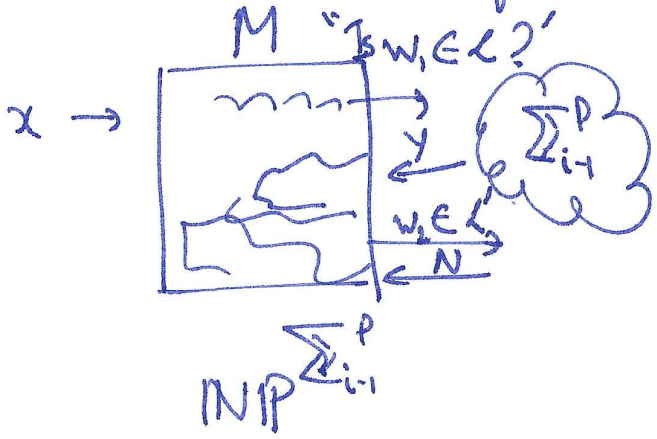
Hence the machine D can accept by guessing this string u_1.

$\therefore$ D is an $NP^{\Sigma^P_{i-1}}$ machine, completing the proof.

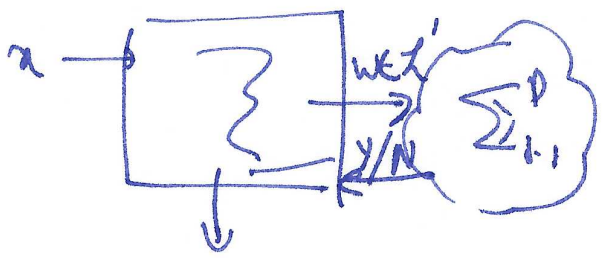
($\Rightarrow$) Consider $\mathcal{L} \in NP^{\Sigma^P_{i-1}}$ (Use induction over i)

We need to "get Def 2" for $\mathcal{L}$.

We need to construct alternating certificates and the machine ("decider") for $\mathcal{L}$.



We want



We need to prove that $\sum_{i-1}^{P}$ is closed under concatenation.

$$\mathcal{L}_1 , \mathcal{L}_2 \in \sum_{i-1}^{P}$$

$$\mathcal{L}_1\mathcal{L}_2 = \left\{ \langle x_1, x_2 \rangle \;\middle|\; x_1 \in \mathcal{L}_1 , x_2 \in \mathcal{L}_2 \right\}$$

$\sum_{i-1}^{P}$ by induction is $NP^{\sum_{i-2}^{P}}$

Hence, there is a machine M_1 deciding $\mathcal{L}_1$ that runs in non-deterministic polynomial time and makes oracle calls to languages in $\sum_{i-2}^{P}$.

There is an analogous machine for M_2 deciding $\mathcal{L}_2$

The machine that runs M_1 on x_1 & M_2 on x_2 (and accepts iff both accept) is also in $\sum_{i-1}^{P}$.

This machine decides $\mathcal{L}_1\mathcal{L}_2$.

---

x is accepted by M iff there exists a "transcript" t where $t \leq p(|x|)$ and t is correct.

polynomial

t is the full tableau of configurations;

t is a list of configuration strings.

t contains queries $w_1, w_2 \ldots w_m$ $\quad m \leq p(|x|)$

and the answers $b_1, b_2 \ldots b_m$

$(b_i \in \{0,1\})$

Let us define a "machine" that verifies if the transcript is correct.

The machine needs to verify if each config. follows from previous config. according to TM rules (of M) and if $\forall\ j \leq m$ the answer to "Is $w_j \in \mathcal{L}_j$?" is b_j.

We construct a machine P that determines if

$\forall\ j \leq m$ "Is $w_j \in \mathcal{L}_j$?" is b_j s.t.

iff $\exists\ U_1\ \forall\ U_2 - - -$ (i quantifiers) $P(U_1, U_2 - - -, U_i)$ accepts

$(\forall\ \forall_j\ |U_k| \leq p(|x|))$

By induction, $\mathcal{L}_j \in \sum_{i-1}^{P}$

$\forall\ j \leq m$

$b_{j<1}$

$x \quad w_j \in \mathcal{L}_j$ iff $\exists\ u_{1,j}\ \forall\ u_{2,j}\ \exists\ u_{3,j} - -$ ($(i-1)$ quantifiers) $M_j(w_j, u_{1,j}, u_{2j} - -)$ accepts

$w_r \notin \mathcal{L}_r$ iff $\forall\ u_{2,r}\ \nexists\ u_{3,r} - -u_{i,r}$ ($(i-1)$ quantifiers) $M_r(w_r, u_{2,r}, u_{3,r} - -)$ accepts

We can combine all of these into a single alternating certificate.

$b_r = 0$

$\bigwedge_{j \le m}$ (Answer to "Is $w_j \in L_j$?" is b_j)

$\equiv \exists u_{1,1} u_{1,2} u_{1,3} - - u_{1,jm} \forall u_{2,j} u_{2,2} u_{2,3} - - u_{2,m}$

$\exists u_{3,1} u_{3,2} u_{3,3} - - u_{3,m} - -$

(i switches) ($\bigwedge_{j \le m} M_j(w_j, u_{1,j}, u_{2,j}, u_{3,j} - -)$ accepts)

$\Longleftrightarrow$

For all $j \le m$:

(1) If $b_j = 1$, run $M_j(w_j, u_{1,j}, u_{2,j} - - u_{i-1,j})$

(2) If $b_j = 0$, run $M_j(w_j, u_{2,j}, u_{3,j} - - , u_{i,j})$

Accept if all accept

$x \in L$ if $\exists$ transcript s.t. P accepts (with alt. certificates)

$\exists t$ s.t. $\exists U_1 \forall U_2 - - \exists U_i$

s.t. $P'(U_1, U_2 - - , U_i)$ accepts

P' also verifies transitions

$\exists t, U_1 \forall U_2 \exists U_3 - - U_i$

s.t. $P'(- -)$ accepts

$\blacksquare$