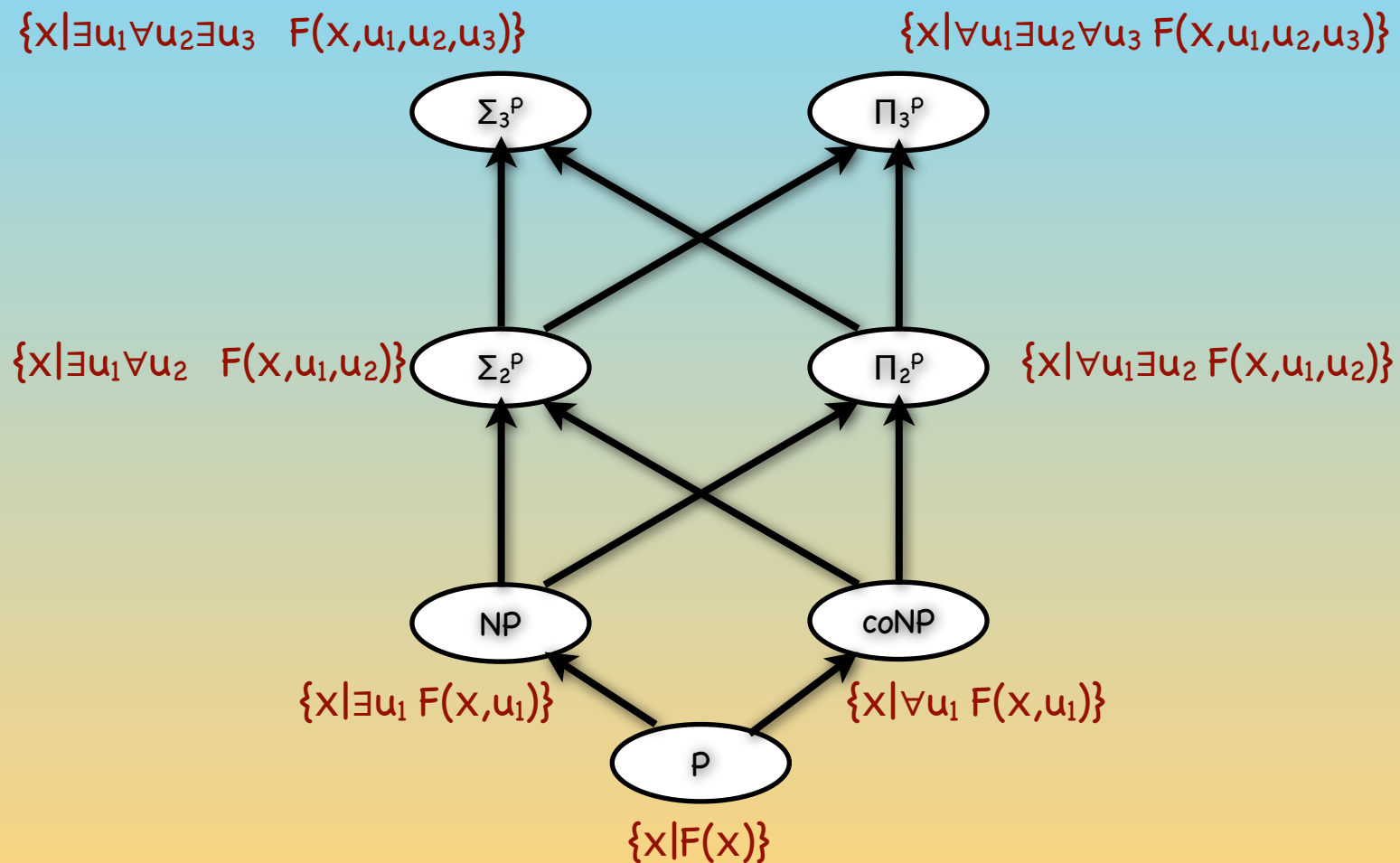


# Computational Complexity

Lecture 8

More of the Polynomial Hierarchy  
Oracle-based Definition

# Recall PH



# Oracle Machines

# Oracle Machines

- Recall Oracle Machine

# Oracle Machines

- Recall Oracle Machine
  - Writes queries on query-tape, enters and leaves query state, and expects answer from oracle on the tape

# Oracle Machines

- Recall Oracle Machine
  - Writes queries on query-tape, enters and leaves query state, and expects answer from oracle on the tape
  - Can run an oracle machine with any oracle

# Oracle Machines

- Recall Oracle Machine
  - Writes queries on query-tape, enters and leaves query state, and expects answer from oracle on the tape
  - Can run an oracle machine with any oracle
  - Oracle fully specified by the input-output behavior

# Oracle Machines

- Recall Oracle Machine
  - Writes queries on query-tape, enters and leaves query state, and expects answer from oracle on the tape
  - Can run an oracle machine with any oracle
  - Oracle fully specified by the input-output behavior
  - Language oracle: answer is a single bit

# Oracle Machines

- Recall Oracle Machine
  - Writes queries on query-tape, enters and leaves query state, and expects answer from oracle on the tape
  - Can run an oracle machine with any oracle
  - Oracle fully specified by the input-output behavior
  - Language oracle: answer is a single bit
    - This is what we consider

# Oracle Machines (ctd.)

# Oracle Machines (ctd.)

- Non-deterministic oracle machine

# Oracle Machines (ctd.)

- Non-deterministic oracle machine
  - Can make non-deterministic choices and make oracle queries. (Note: oracles are deterministic!)

# Oracle Machines (ctd.)

- Non-deterministic oracle machine
  - Can make non-deterministic choices and make oracle queries. (Note: oracles are deterministic!)
  - Said to accept if any thread reaches accept state

# Oracle Machines (ctd.)

- Non-deterministic oracle machine
  - Can make non-deterministic choices and make oracle queries. (Note: oracles are deterministic!)
  - Said to accept if any thread reaches accept state
  - Equivalently, a deterministic oracle machine which takes a (read-once) certificate  $w$  (the list of non-deterministic choices)

# Oracle Machines (ctd.)

- Non-deterministic oracle machine
  - Can make non-deterministic choices and make oracle queries. (Note: oracles are deterministic!)
  - Said to accept if any thread reaches accept state
  - Equivalently, a deterministic oracle machine which takes a (read-once) certificate  $w$  (the list of non-deterministic choices)
    - Said to accept  $x$  if  $\exists w$  such that  $(x,w)$  takes it to accepting state

# Oracle Machines (ctd.)

co-

## • Non-deterministic oracle machine

- Can make non-deterministic choices and make oracle queries. (Note: oracles are deterministic!)  
 $\text{all threads reach}$
- Said to accept if ~~any thread reaches~~ accept state
- Equivalently, a deterministic oracle machine which takes a (read-once) certificate  $w$  (the list of non-deterministic choices)  
 $\forall w$ 
  - Said to accept  $x$  if  ~~$\exists w$  such that~~  $(x, w)$  takes it to accepting state

NPA

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form
  - $B = \{x \mid \exists w M^A(x, w) = 1\}$

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form
  - $B = \{x \mid \exists w \ M^A(x, w) = 1\}$ 
    - where  $M$  deterministic oracle machine

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form
  - $B = \{x \mid \exists w M^A(x, w) = 1\}$ 
    - where  $M$  deterministic oracle machine
    - $M^A$  runs in  $\text{poly}(|x|)$  time and  $|w| = \text{poly}(|x|)$

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form
  - $B = \{x \mid \exists w M^A(x, w) = 1\}$ 
    - where  $M$  deterministic oracle machine
    - $M^A$  runs in  $\text{poly}(|x|)$  time and  $|w| = \text{poly}(|x|)$
- $\text{co-}(NP^A) = (\text{co-NP})^A$

# $NP^A$

- $NP^A$  : class of languages accepted by oracle NTMs with oracle for  $A$  in poly time
- Certificate version:  $NP^A$  has languages of the form
  - $B = \{x \mid \exists w \ M^A(x, w) = 1\}$ 
    - where  $M$  deterministic oracle machine
    - $M^A$  runs in  $\text{poly}(|x|)$  time and  $|w| = \text{poly}(|x|)$
- $\text{co-}(NP^A) = (\text{co-NP})^A$ 
  - languages of the form  $\{x \mid \forall w \ M^A(x, w) = 1\}$

NPA

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$ 
  - Can “implement” the oracle as a subroutine

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$ 
  - Can “implement” the oracle as a subroutine
- If  $A \in NP$ ?

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$ 
  - Can “implement” the oracle as a subroutine
- If  $A \in NP$ ?
  - Oracle for  $A$  is an oracle for  $A^c$  too!  $NP^A = NP^{A^c}$

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$ 
  - Can “implement” the oracle as a subroutine
- If  $A \in NP$ ?
  - Oracle for  $A$  is an oracle for  $A^c$  too!  $NP^A = NP^{A^c}$
  - $NP \cup co-NP \subseteq NP^{SAT}$

# $NP^A$

- If  $A \in P$ ,  $NP^A = NP$ 
  - Can “implement” the oracle as a subroutine
- If  $A \in NP$ ?
  - Oracle for  $A$  is an oracle for  $A^c$  too!  $NP^A = NP^{A^c}$
  - $NP \cup co-NP \subseteq NP^{SAT}$ 
    - Can we better characterize  $NP^{SAT}$ ?

# $\text{NP}^{\text{NP}}$ and relatives

# $\text{NP}^{\text{NP}}$ and relatives

- $\text{NP}^{\text{SAT}} = \bigcup_{A \in \text{NP}} \text{NP}^A$

# $\text{NP}^{\text{NP}}$ and relatives

- $\text{NP}^{\text{SAT}} = \bigcup_{A \in \text{NP}} \text{NP}^A$ 
  - Oracle for  $A$  can be implemented using oracle for SAT in polynomial time (deterministically)

# $NP^{NP}$ and relatives

- $NP^{SAT} = \bigcup_{A \in NP} NP^A$ 
  - Oracle for  $A$  can be implemented using oracle for SAT in polynomial time (deterministically)
  - $NP^{SAT}$  also called  $NP^{NP}$

# $NP^{NP}$ and relatives

- $NP^{SAT} = \bigcup_{A \in NP} NP^A$ 
  - Oracle for  $A$  can be implemented using oracle for SAT in polynomial time (deterministically)
  - $NP^{SAT}$  also called  $NP^{NP}$
- $NP^{\Sigma_k} = \bigcup_{A \in \Sigma_k} NP^A = NP^{\Sigma_k SAT}$

# $NP^{NP}$ and relatives

- $NP^{SAT} = \bigcup_{A \in NP} NP^A$ 
  - Oracle for  $A$  can be implemented using oracle for SAT in polynomial time (deterministically)
  - $NP^{SAT}$  also called  $NP^{NP}$
- $NP^{\Sigma_k} = \bigcup_{A \in \Sigma_k} NP^A = NP^{\Sigma_k SAT}$
- Will show  $NP^{\Sigma_k} = \Sigma_{k+1}^P$  (alt. definition for  $\Sigma_{k+1}^P$ )

# $NP^{NP}$ and relatives

- $NP^{SAT} = \bigcup_{A \in NP} NP^A$ 
  - Oracle for  $A$  can be implemented using oracle for SAT in polynomial time (deterministically)
  - $NP^{SAT}$  also called  $NP^{NP}$
- $NP^{\Sigma_k} = \bigcup_{A \in \Sigma_k} NP^A = NP^{\Sigma_k SAT}$
- Will show  $NP^{\Sigma_k} = \Sigma_{k+1}^P$  (alt. definition for  $\Sigma_{k+1}^P$ )
  - In particular,  $NP^{NP} = \Sigma_2^P$

$$\Sigma_{k+1} = N P^{\Sigma_k}$$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$

- $L = \{ x \mid \exists w (x, w) \in L' \}$ , where  $L'$  in  $\Pi_k^P$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$ 
  - $L = \{ x \mid \exists w (x, w) \in L' \}$ , where  $L' \in \Pi_k^P$
  - So  $L \in \text{NP}^{L'}$  where  $L' \in \Pi_k^P$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$ 
  - $L = \{ x \mid \exists w (x, w) \in L' \}$ , where  $L'$  in  $\Pi_k^P$
  - So  $L$  in  $\text{NP}^{L'}$  where  $L'$  in  $\Pi_k^P$ 
    - But  $\text{NP}^{L'} \subseteq \text{NP}^{\Pi_k} = \text{NP}^{\Sigma_k}$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$ 
  - $L = \{ x \mid \exists w (x, w) \in L' \}$ , where  $L'$  in  $\Pi_k^P$
  - So  $L$  in  $\text{NP}^{L'}$  where  $L'$  in  $\Pi_k^P$ 
    - But  $\text{NP}^{L'} \subseteq \text{NP}^{\Pi_k} = \text{NP}^{\Sigma_k}$
- So  $\Sigma_{k+1}^P \subseteq \text{NP}^{\Sigma_k}$

$$\Sigma_{k+1} = \text{NP}^{\Sigma_k}$$

- Consider  $L \in \Sigma_{k+1}^P$ 
  - $L = \{ x \mid \exists w (x, w) \in L' \}$ , where  $L'$  in  $\Pi_k^P$
  - So  $L$  in  $\text{NP}^{L'}$  where  $L'$  in  $\Pi_k^P$ 
    - But  $\text{NP}^{L'} \subseteq \text{NP}^{\Pi_k} = \text{NP}^{\Sigma_k}$
- So  $\Sigma_{k+1}^P \subseteq \text{NP}^{\Sigma_k}$
- Now to show  $\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}^P$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $\text{NP}^A \subseteq \Sigma_{k+1}^P$  if  $A \in \Sigma_k^P$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $\text{NP}^A \subseteq \Sigma_{k+1}^P$  if  $A \in \Sigma_k^P$ 
  - For  $B \in \text{NP}^A$  poly-time TM  $M$  s.t.  $B = \{ x \mid \exists w M^A(x,w)=1 \}$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $\text{NP}^A \subseteq \Sigma_{k+1}^P$  if  $A \in \Sigma_k^P$ 
  - For  $B \in \text{NP}^A$  poly-time TM  $M$  s.t.  $B = \{ x \mid \exists w M^A(x,w)=1 \}$
  - i.e.,  $B = \{ x \mid \exists w \exists \text{ans } M^{\langle \text{ans} \rangle}(x,w)=1 \text{ and "ans correct"} \}$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $\text{NP}^A \subseteq \Sigma_{k+1}^P$  if  $A$  in  $\Sigma_k^P$ 
  - For  $B \in \text{NP}^A$  poly-time TM  $M$  s.t.  $B = \{ x \mid \exists w M^A(x,w)=1 \}$
  - i.e.,  $B = \{ x \mid \exists w \exists \text{ans } M^{<\text{ans}>}(x,w)=1 \text{ and "ans correct"} \}$
  - To show  $C = \{ (x,w,\text{ans}) \mid M^{<\text{ans}>}(x,w)=1 \text{ and "ans correct"} \}$  in  $\Sigma_{k+1}^P$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $\text{NP}^A \subseteq \Sigma_{k+1}^P$  if  $A$  in  $\Sigma_k^P$ 
  - For  $B \in \text{NP}^A$  poly-time TM  $M$  s.t.  $B = \{ x \mid \exists w M^A(x,w)=1 \}$
  - i.e.,  $B = \{ x \mid \exists w \exists \text{ans } M^{<\text{ans}>}(x,w)=1 \text{ and "ans correct"} \}$
  - To show  $C = \{ (x,w,\text{ans}) \mid M^{<\text{ans}>}(x,w)=1 \text{ and "ans correct"} \}$  in  $\Sigma_{k+1}^P$ 
    - Then  $B$  also in  $\Sigma_{k+1}^P$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z=Z(x,w)$ .  $ans$  is a single bit saying if  $z$  in  $A$  or not

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z=Z(x,w)$ .  $ans$  is a single bit saying if  $z$  in  $A$  or not
  - "ans correct":  $(ans=1 \wedge z \in A) \text{ or } (ans=0 \wedge z \notin A)$

$$\text{NP}^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z = Z(x, w)$ .  $\text{ans}$  is a single bit saying if  $z$  in  $A$  or not
  - "ans correct":  $(\text{ans} = 1 \wedge z \in A) \text{ or } (\text{ans} = 0 \wedge z \notin A)$
  - $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge \exists u_1 \forall u_2 \dots Q_k u_k F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge \forall v_1 \exists v_2 \dots Q'_k v_k F(z, v_1, \dots) = 0)]\}$

$$NP^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z = Z(x, w)$ .  $\text{ans}$  is a single bit saying if  $z$  in  $A$  or not
  - "ans correct":  $(\text{ans} = 1 \wedge z \in A) \text{ or } (\text{ans} = 0 \wedge z \notin A)$
  - $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge \exists u_1 \forall u_2 \dots Q_k u_k F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge \forall v_1 \exists v_2 \dots Q'_k v_k F(z, v_1, \dots) = 0)]\}$
  - $C = \{(x, w, \text{ans}) \mid \exists u_1 \forall u_2 \forall v_1 \exists u_3 \forall v_2 \dots Q_k u_k \forall v_{k-1} Q'_k v_k M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge F(z, v_1, \dots) = 0)]\}$

$$NP^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z = Z(x, w)$ .  $\text{ans}$  is a single bit saying if  $z$  in  $A$  or not
  - "ans correct":  $(\text{ans} = 1 \wedge z \in A) \text{ or } (\text{ans} = 0 \wedge z \notin A)$
  - $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge \exists u_1 \forall u_2 \dots Q_k u_k F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge \forall v_1 \exists v_2 \dots Q'_k v_k F(z, v_1, \dots) = 0)]\}$
  - $C = \{(x, w, \text{ans}) \mid \exists u_1 \forall u_2 \forall v_1 \exists u_3 \forall v_2 \dots Q_k u_k \forall v_{k-1} Q'_k v_k M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge F(z, v_1, \dots) = 0)]\}$ 

*in  $\Sigma_{k+1}^P$*

$$NP^{\Sigma_k} \subseteq \Sigma_{k+1}$$

- To show  $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \text{ and "ans correct"}\}$  in  $\Sigma_{k+1}^P$ 
  - Suppose  $M$  makes only one query  $z = Z(x, w)$ .  $\text{ans}$  is a single bit saying if  $z$  in  $A$  or not
  - "ans correct":  $(\text{ans} = 1 \wedge z \in A) \text{ or } (\text{ans} = 0 \wedge z \notin A)$
  - $C = \{(x, w, \text{ans}) \mid M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge \exists u_1 \forall u_2 \dots Q_k u_k F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge \forall v_1 \exists v_2 \dots Q'_k v_k F(z, v_1, \dots) = 0)]\}$
  - $C = \{(x, w, \text{ans}) \mid \exists u_1 \forall u_2 \forall v_1 \exists u_3 \forall v_2 \dots Q_k u_k \forall v_{k-1} Q'_k v_k M^{\langle \text{ans} \rangle}(x, w) = 1 \wedge [(\text{ans} = 1 \wedge F(z, u_1, \dots) = 1) \text{ or } (\text{ans} = 0 \wedge F(z, v_1, \dots) = 0)]\}$  in  $\Sigma_{k+1}^P$
  - Changes for  $t$  queries:  $z = Z(x, w) \rightarrow (z^{(1)} \dots z^{(t)}) = Z(x, w, \text{ans})$ ,  
 $u_i \rightarrow u_i^{(1)} \dots u_i^{(t)}$ ,  $v_i \rightarrow v_i^{(1)} \dots v_i^{(t)}$ , and use conjunction of  $t$  checks  
(for  $j = 1, \dots, t$ ) of the form  $[(\text{ans}^{(j)} = 1 \wedge F(z^{(j)}, u_1^{(j)}, \dots) = 1) \text{ or } (\text{ans}^{(j)} = 0 \wedge F(z^{(j)}, v_1^{(j)}, \dots) = 0)]$

# Oracle Version

# Oracle Version

•  $\Sigma_{k+1}^P = \text{NP}^{\Sigma_k}$  (with  $\Sigma_0^P = P$ )

# Oracle Version

- $\Sigma_{k+1}^P = \text{NP}^{\Sigma_k}$  (with  $\Sigma_0^P = P$ )
- $\Pi_{k+1}^P = \text{co-NP}^{\Pi_k}$  (with  $\Pi_0^P = P$ )

# Oracle Version

- $\Sigma_{k+1}^P = \text{NP}^{\Sigma_k}$  (with  $\Sigma_0^P = P$ )
- $\Pi_{k+1}^P = \text{co-NP}^{\Pi_k}$  (with  $\Pi_0^P = P$ )
  - $\Pi_{k+1}^P = \text{co}-(\text{NP}^{\Sigma_k}) = \text{co-NP}^{\Sigma_k} = \text{co-NP}^{\Pi_k}$

$$\Delta_k^p$$

$$\Delta_k^p$$

- $\Delta_{k+1}^p = p^{\Sigma_k} = p^{\Pi_k}$

$$\Delta_k^p$$

- $\Delta_{k+1}^p = p^{\Sigma_k} = p^{\Pi_k}$

- $\Delta_1^p = p$

$$\Delta_k^P$$

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$

- $\Delta_1^P = P$

- $\Delta_2^P = P^{NP}$

$$\Delta_k^P$$

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$

- $\Delta_1^P = P$

- $\Delta_2^P = P^{NP}$

- Note that  $\Delta_2^P = \text{co-}\Delta_2^P$

$$\Delta_k^P$$

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$

- $\Delta_1^P = P$

- $\Delta_2^P = P^{NP}$

- Note that  $\Delta_2^P = \text{co-}\Delta_2^P$

- $\Delta_{k+1}^P \supseteq \Sigma_k^P \cup \Pi_k^P$

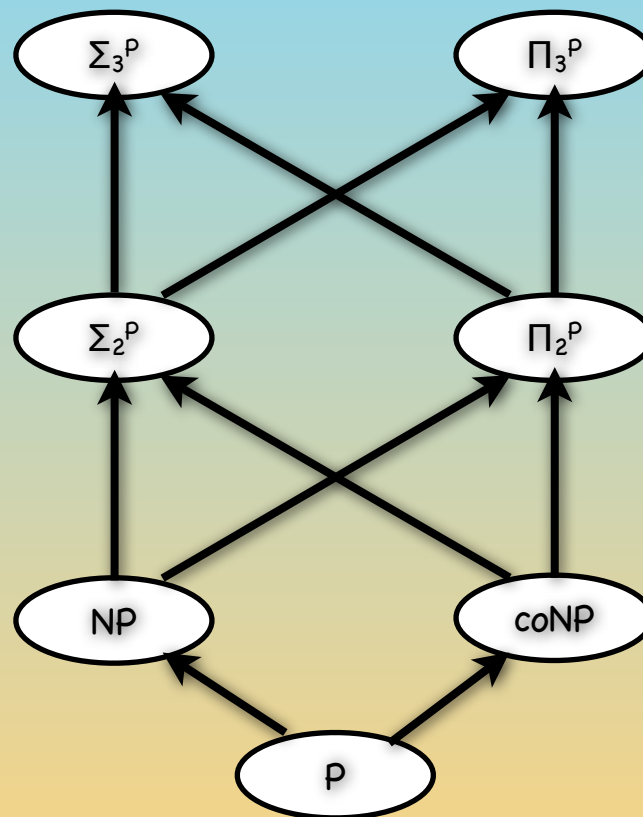
$$\Delta_k^P$$

- $\Delta_{k+1}^P = \Sigma_k^P = \Pi_k^P$ 
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- Note that  $\Delta_2^P = \text{co-}\Delta_2^P$
- $\Delta_{k+1}^P \supseteq \Sigma_k^P \cup \Pi_k^P$
- $\Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P$  (why?)

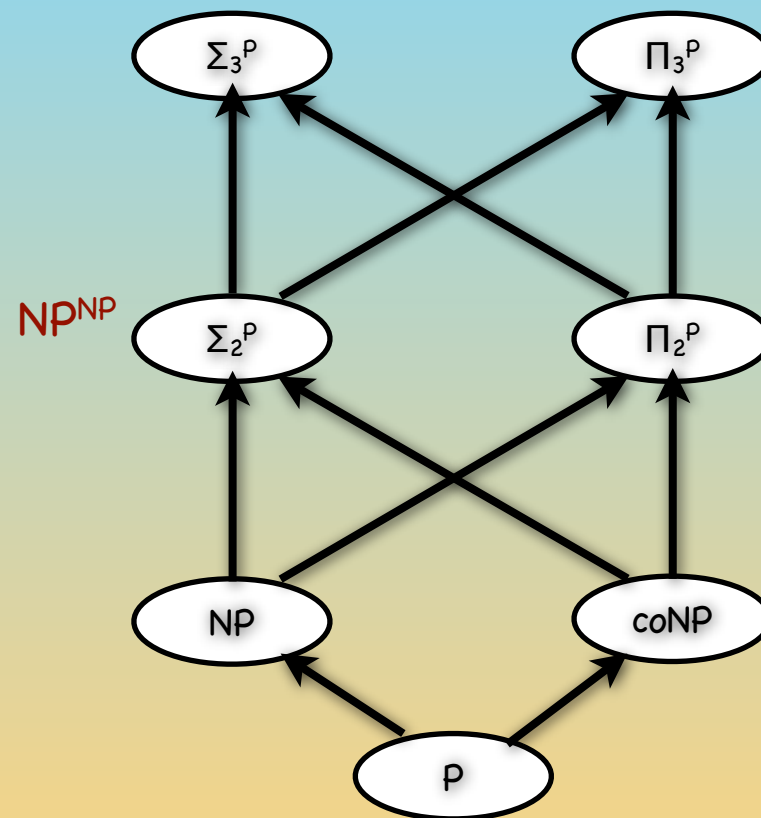
$$\Delta_k^P$$

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$ 
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- Note that  $\Delta_2^P = \text{co-}\Delta_2^P$
- $\Delta_{k+1}^P \supseteq \Sigma_k^P \cup \Pi_k^P$
- $\Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P$  (why?)
  - $P^{\Sigma_k} \subseteq NP^{\Sigma_k} \cap \text{coNP}^{\Sigma_k}$

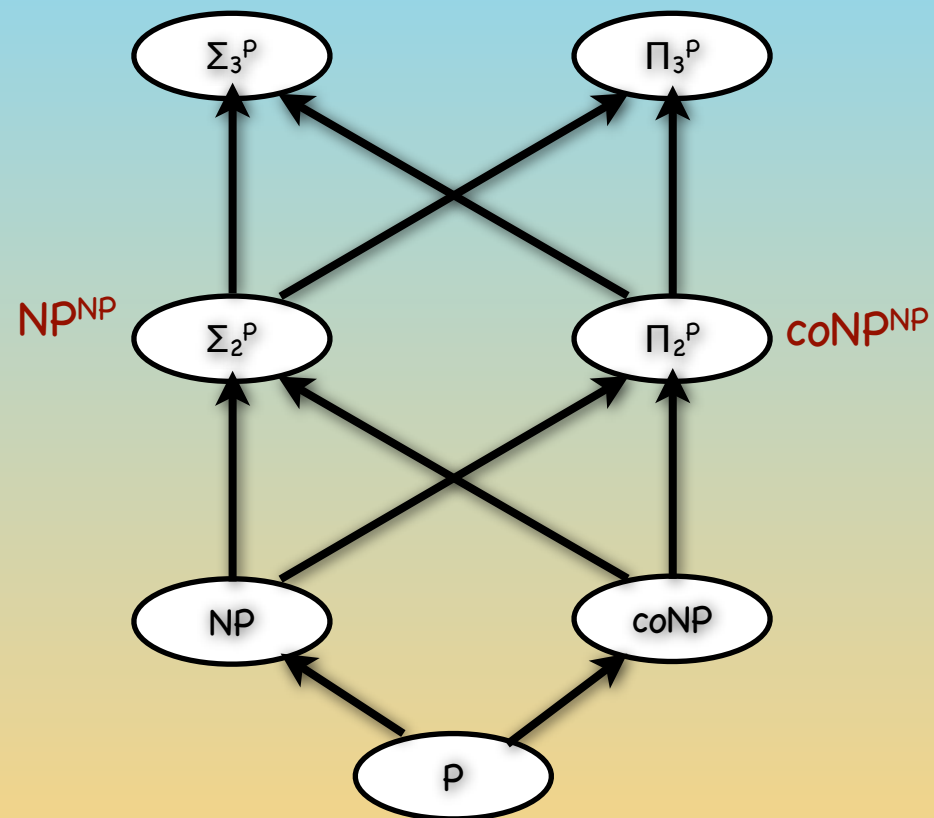
# PH



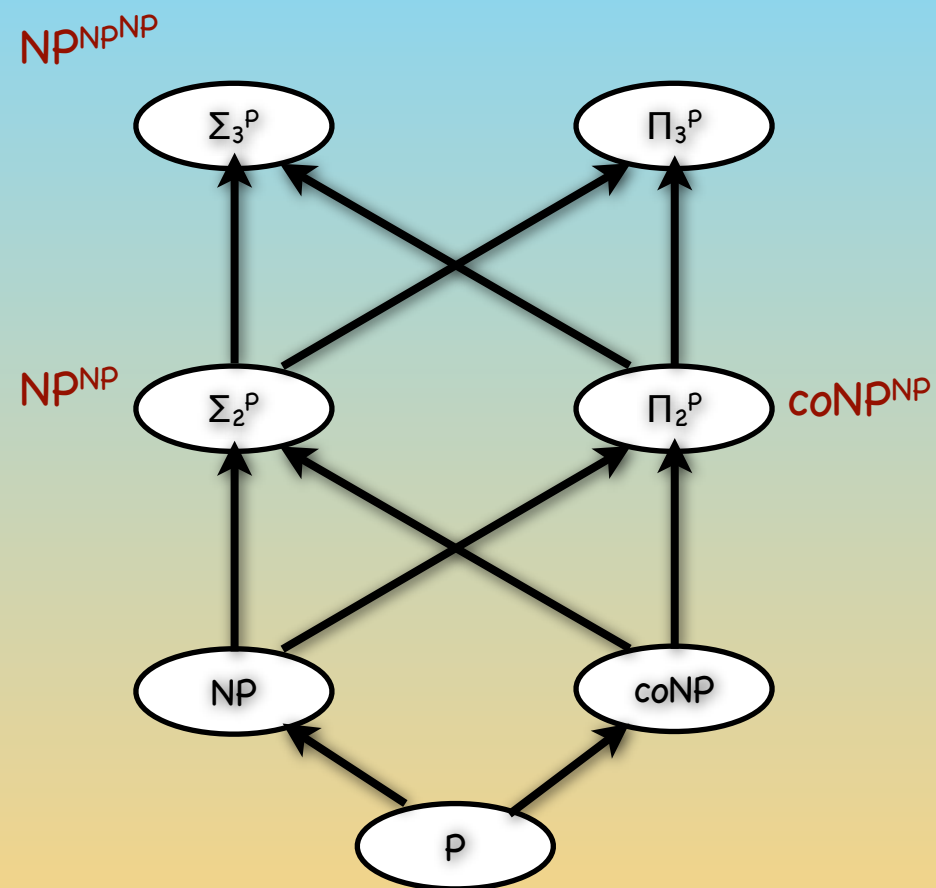
# PH



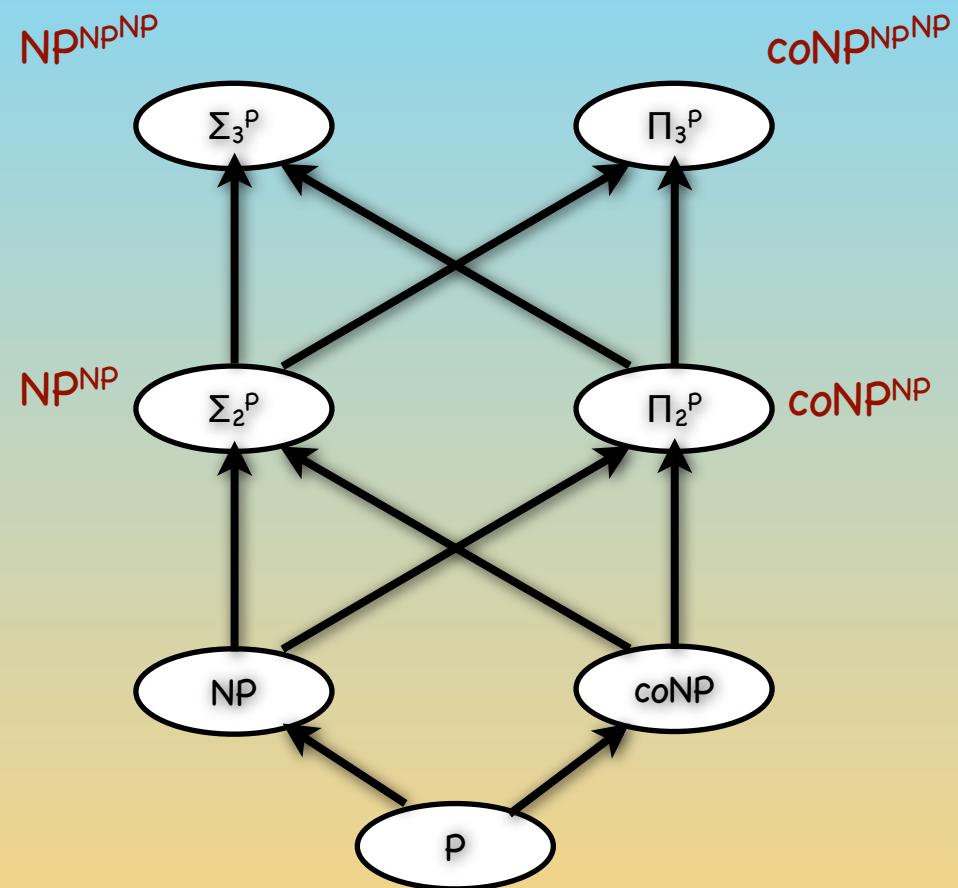
# PH



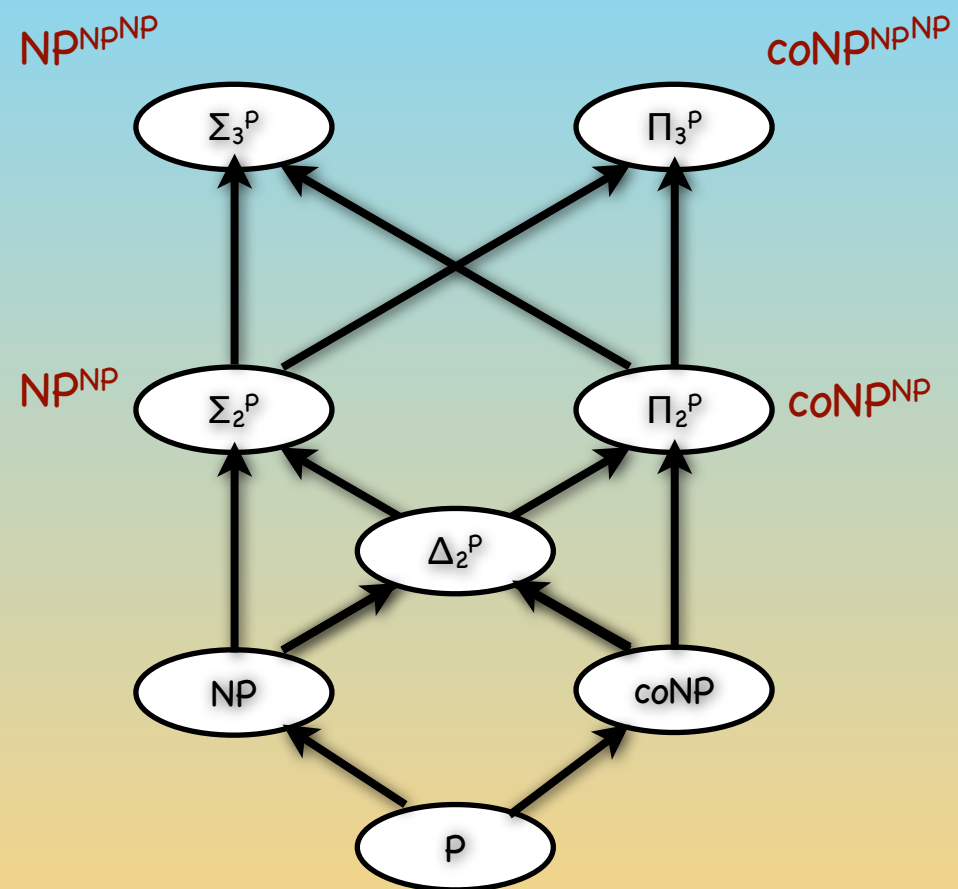
# PH



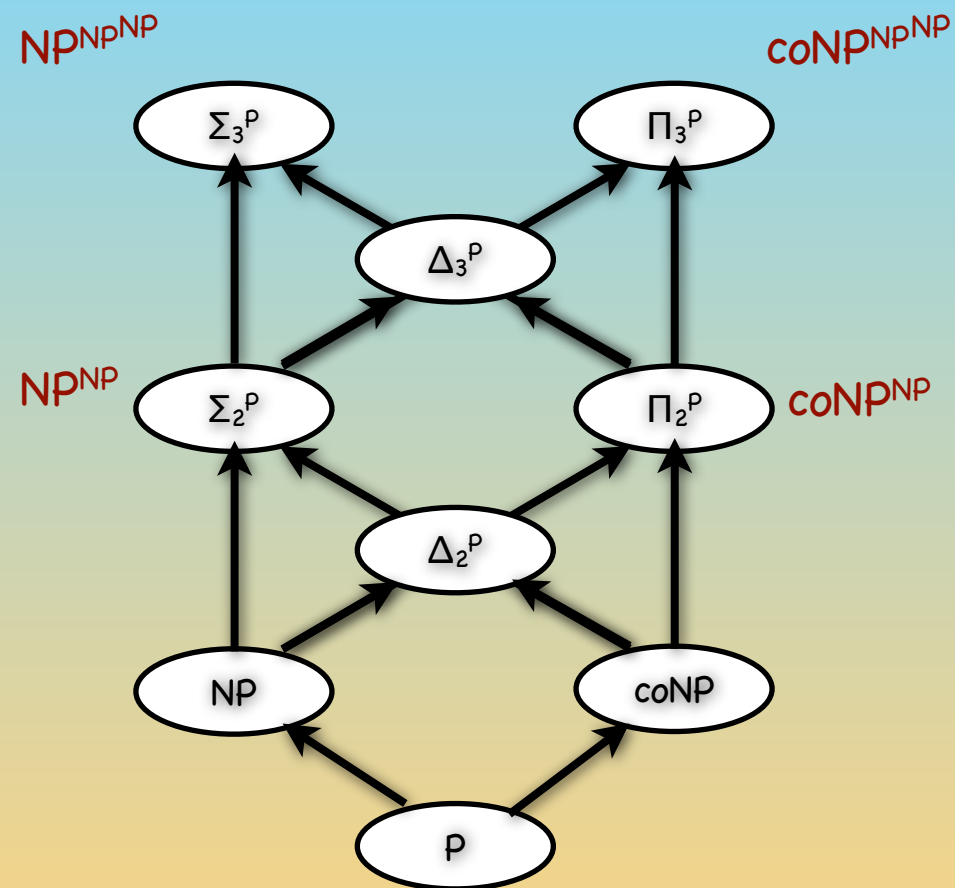
# PH



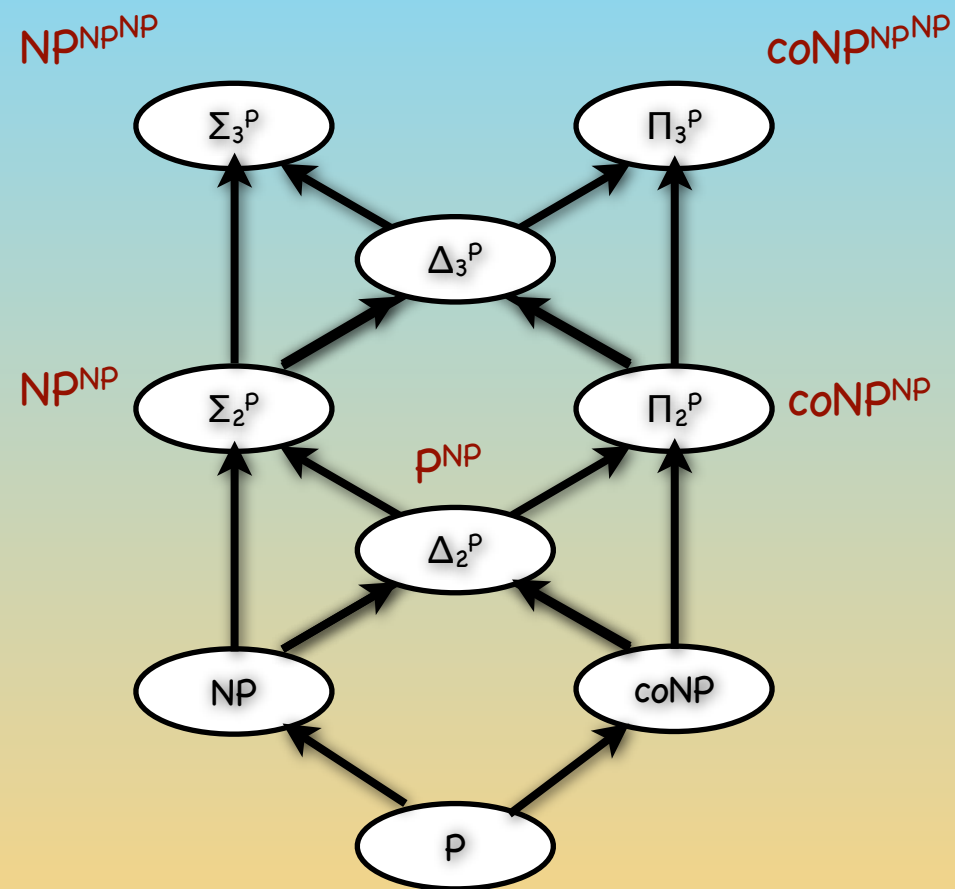
# PH



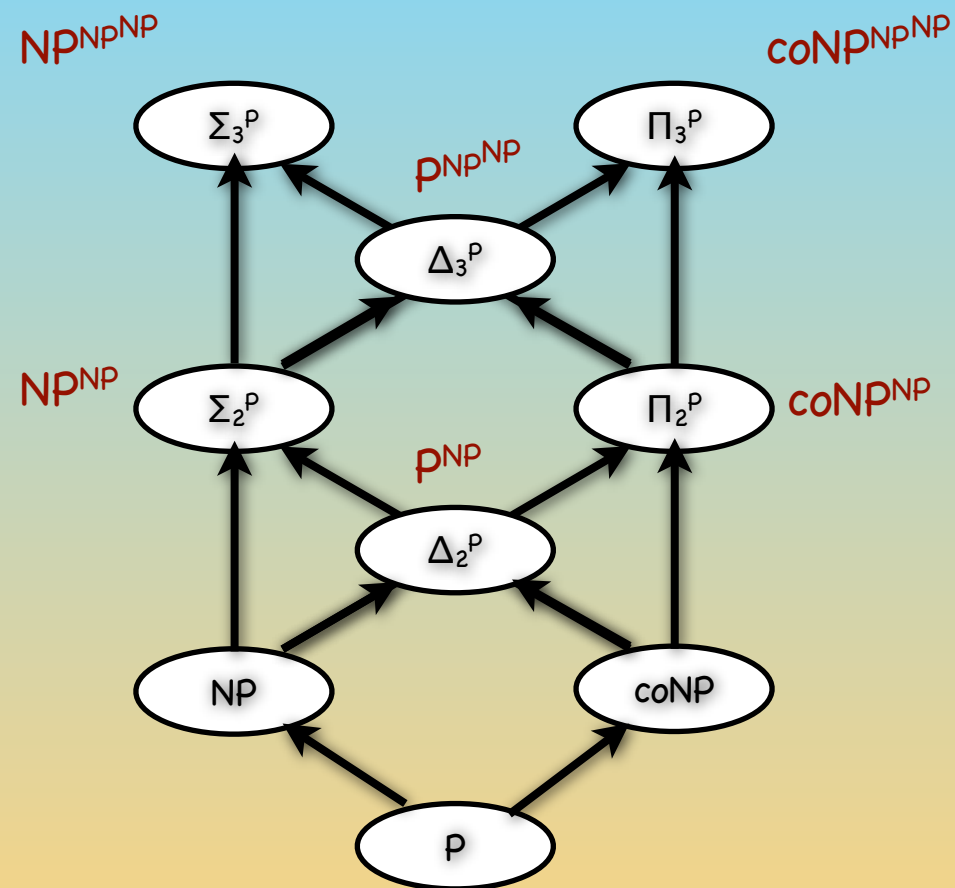
# PH



# PH



# PH



# Today

# Today

- Today, more PH

# Today

- Today, more PH
  - Oracle-based definitions (in particular  $NP^{NP} = \Sigma_2^P$ )

# Today

- Today, more PH
  - Oracle-based definitions (in particular  $NP^{NP} = \Sigma_2^P$ )
- Next lecture, more PH

# Today

- Today, more PH
  - Oracle-based definitions (in particular  $NP^{NP} = \Sigma_2^P$ )
- Next lecture, more PH
  - Alternating TM-based definitions

# Today

- Today, more PH
  - Oracle-based definitions (in particular  $NP^{NP} = \Sigma_2^P$ )
- Next lecture, more PH
  - Alternating TM-based definitions
  - Time-Space tradeoffs