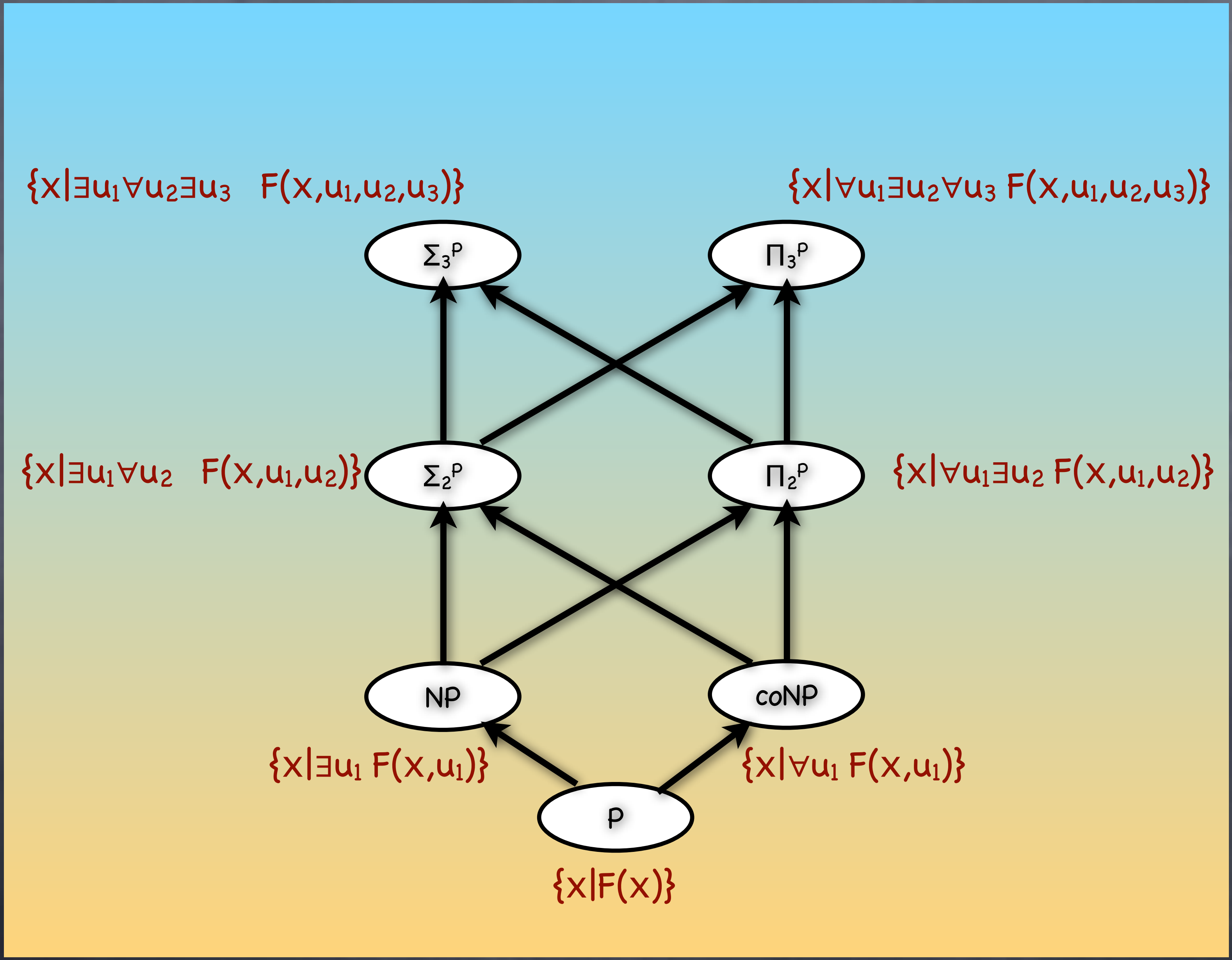
Computational Complexity

Lecture 8

More of the Polynomial Hierarchy

Oracle-based Definition

Recall PH



{x|∃u1∀u2∃u3 F(x,u1,u2,u3)}
{x|∀u1∃u2∀u3 F(x,u1,u2,u3)}
Σ3P
Π3P
{x|∃u1∀u2 F(x,u1,u2)}
Σ2P
Π2P
{x|∀u1∃u2 F(x,u1,u2)}
NP
coNP
{x|∃u1 F(x,u1)}
{x|∀u1 F(x,u1)}
P
{x|F(x)}

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. (<u>Note: oracles are deterministic!</u>)

Oracle Machines (ctd.)

- Non-deterministic oracle machine
  - Can make non-deterministic choices and make oracle queries. (<u>Note: oracles are deterministic!)</u>
  - 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 ∃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!)
  - Said to accept if ~~any thread reaches~~ all threads reach 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 ~~∃w such that~~ ∀w (x,w) takes it to accepting state

NP^A

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 poly(|x|) time and |w|=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 poly(|x|) time and |w|=poly(|x|)
    - i.e., $B = \{x \mid \exists w\ (x,w) \in L\}$, where L in P^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 poly(|x|) time and |w|=poly(|x|)
    - i.e., $B = \{x \mid \exists w\ (x,w) \in L\}$, where L in P^A
- $\text{co-}(NP^A) = (\text{co-}NP)^A$

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 poly(|x|) time and |w|=poly(|x|)
    - i.e., $B = \{x \mid \exists w\ (x,w) \in L\}$, where L in P^A
- co-(NP^A) = (co-NP)^A
  - languages of the form $\{x \mid \forall w\ (x,w) \in L\}$, where L in P^A

NP^A

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! $NP^A = NP^{A^c}$
  - $NP \cup co\text{-}NP \subseteq NP^{SAT}$

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! $NP^A = NP^{A^c}$
  - $NP \cup co\text{-}NP \subseteq NP^{SAT}$
    - Can we better characterize NP^{SAT}?

NP^{NP} and relatives

NP^{NP} and relatives

- $NP^{SAT} = \cup_{A \in NP} NP^A$

NP^{NP} and relatives

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

NP^{NP} and relatives

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

NP^{NP} and relatives

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

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 Σ_{k+1}^P)

NP^{NP} and relatives

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

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

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

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

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

- Consider $L \in \Sigma_{k+1}^P$
  - $L = \{ x | \exists w \ (x,w) \in L' \}$, where L' in Π_k^P

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- To show C = {(x,w,ans) | $M^{\langle ans \rangle}(x,w)=1$ and "ans correct"} in Σ_{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

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

- To show C = {(x,w,ans) | $M^{\langle ans \rangle}(x,w)=1$ and "ans correct"} in Σ_{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) or (ans=0 $\wedge$ z $\notin$ A)

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

- To show C = {(x,w,ans) | $M^{<ans>}(x,w)=1$ and "ans correct"} in Σ_{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) or (ans=0 $\wedge$ z $\notin$ A)
  - C={(x,w,ans) | $M^{<ans>}(x,w)=1$ $\wedge$ [(ans=1 $\wedge$ $\exists u_1 \forall u_2 \ldots Q_k u_k$ F(z,u_1,...)=1) or (ans=0 $\wedge$ $\forall v_1 \exists v_2 \ldots Q'_k v_k$ F(z,v_1,...)=0)] }

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

- To show C = {(x,w,ans) | $M^{\langle ans \rangle}(x,w)=1$ and "ans correct"} in Σ_{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) or (ans=0 $\wedge$ z $\notin$ A)
  - $C=\{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge \exists u_1 \forall u_2 \ldots Q_k u_k\, F(z,u_1,\ldots)=1) \text{ or } (ans=0 \wedge \forall v_1 \exists v_2 \ldots Q'_k v_k\, F(z,v_1,\ldots)=0)]\}$
  - $C=\{(x,w,ans) \mid \exists u_1 \forall u_2 v_1 \exists u_3 v_2 \ldots Q_k u_k v_{k-1} Q'_k v_k\ \ M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge F(z,u_1,\ldots)=1) \text{ or } (ans=0 \wedge F(z,v_1,\ldots)=0)]\}$

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

- To show $C = \{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1$ and "ans correct"$\}$ in Σ_{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": $(ans=1 \wedge z \in A)$ or $(ans=0 \wedge z \notin A)$
  - $C=\{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge \exists u_1 \forall u_2 \ldots Q_k u_k\ F(z,u_1,\ldots)=1)$ or $(ans=0 \wedge \forall v_1 \exists v_2 \ldots Q'_k v_k\ F(z,v_1,\ldots)=0)]\}$
  - $C=\{(x,w,ans) \mid \exists u_1 \forall u_2 v_1 \exists u_3 v_2 \ldots Q_k u_k v_{k-1} Q'_k v_k\ \ M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge F(z,u_1,\ldots)=1)$ or $(ans=0 \wedge F(z,v_1,\ldots)=0)]\}$

in Σ_{k+1}^P

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

- To show $C = \{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1$ and "ans correct"$\}$ in Σ_{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": $(ans=1 \wedge z \in A)$ or $(ans=0 \wedge z \notin A)$
  - $C=\{(x,w,ans) \mid M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge \exists u_1 \forall u_2 \ldots Q_k u_k\ F(z,u_1,\ldots)=1)$ or $(ans=0 \wedge \forall v_1 \exists v_2 \ldots Q'_k v_k\ F(z,v_1,\ldots)=0)]\ \}$
  - $C=\{(x,w,ans) \mid \exists u_1 \forall u_2 v_1 \exists u_3 v_2 \ldots Q_k u_k v_{k-1} Q'_k v_k\ \ M^{\langle ans \rangle}(x,w)=1 \wedge [(ans=1 \wedge F(z,u_1,\ldots)=1)$ or $(ans=0 \wedge F(z,v_1,\ldots)=0)]\ \}$

    in Σ_{k+1}^P
  - Changes for t queries: $z=Z(x,w) \rightarrow (z^{(1)} \ldots z^{(t)}) = Z(x,w,ans)$, $u_i \rightarrow u_i^{(1)} \ldots u_i^{(t)}$, $v_i \rightarrow v_i^{(1)} \ldots v_i^{(t)}$, and use conjunction of t checks (for $j=1,\ldots,t$) of the form $[\ (ans^{(j)}=1 \wedge F(z^{(j)},u_1^{(j)},\ldots)=1)$ or $(ans^{(j)}=0 \wedge F(z^{(j)},v_1^{(j)},\ldots)=0)\ ]$

Oracle Version

Oracle Version

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

Oracle Version

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

Oracle Version

- **Σ_{k+1}^P = NP^{Σ_k}** (with $\Sigma_0^P = P$)
- **Π_{k+1}^P = co-NP^{Π_k}** (with $\Pi_0^P = P$)
  - Π_{k+1}^P = co-(NP^{Σ_k}) = co-NP^{Σ_k} = co-NP^{Π_k}

Δ_k^P

Δ_k^P

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

Δ_k^P

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1^P = P$

$\Delta_k{}^P$

- $\Delta_{k+1}{}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1{}^P = P$
  - $\Delta_2{}^P = P^{NP}$

Δ_k^P

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- Note that $\Delta_2^P = \text{co-}\Delta_2^P$

Δ_k^P

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- Note that $\Delta_2^P = \text{co-}\Delta_2^P$
- $\Delta_{k+1}^P \supseteq \Sigma_k^P \cup \Pi_k^P$

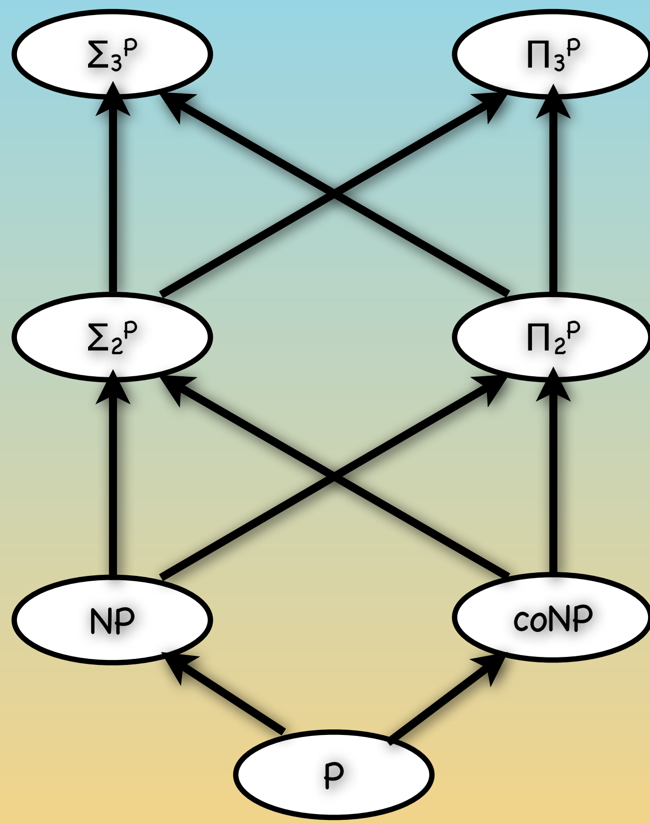
Δ_k^P

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- 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?)

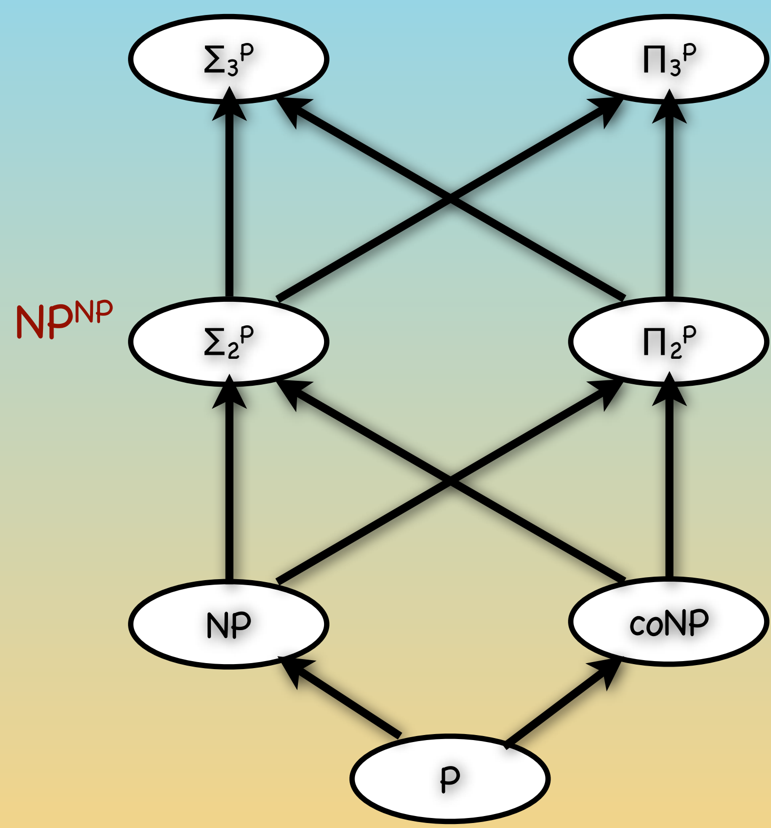
Δ_k^P

- $\Delta_{k+1}^P = P^{\Sigma_k} = P^{\Pi_k}$
  - $\Delta_1^P = P$
  - $\Delta_2^P = P^{NP}$
- 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?)
  - $P^{\Sigma_k} \subseteq NP^{\Sigma_k} \cap coNP^{\Sigma_k}$

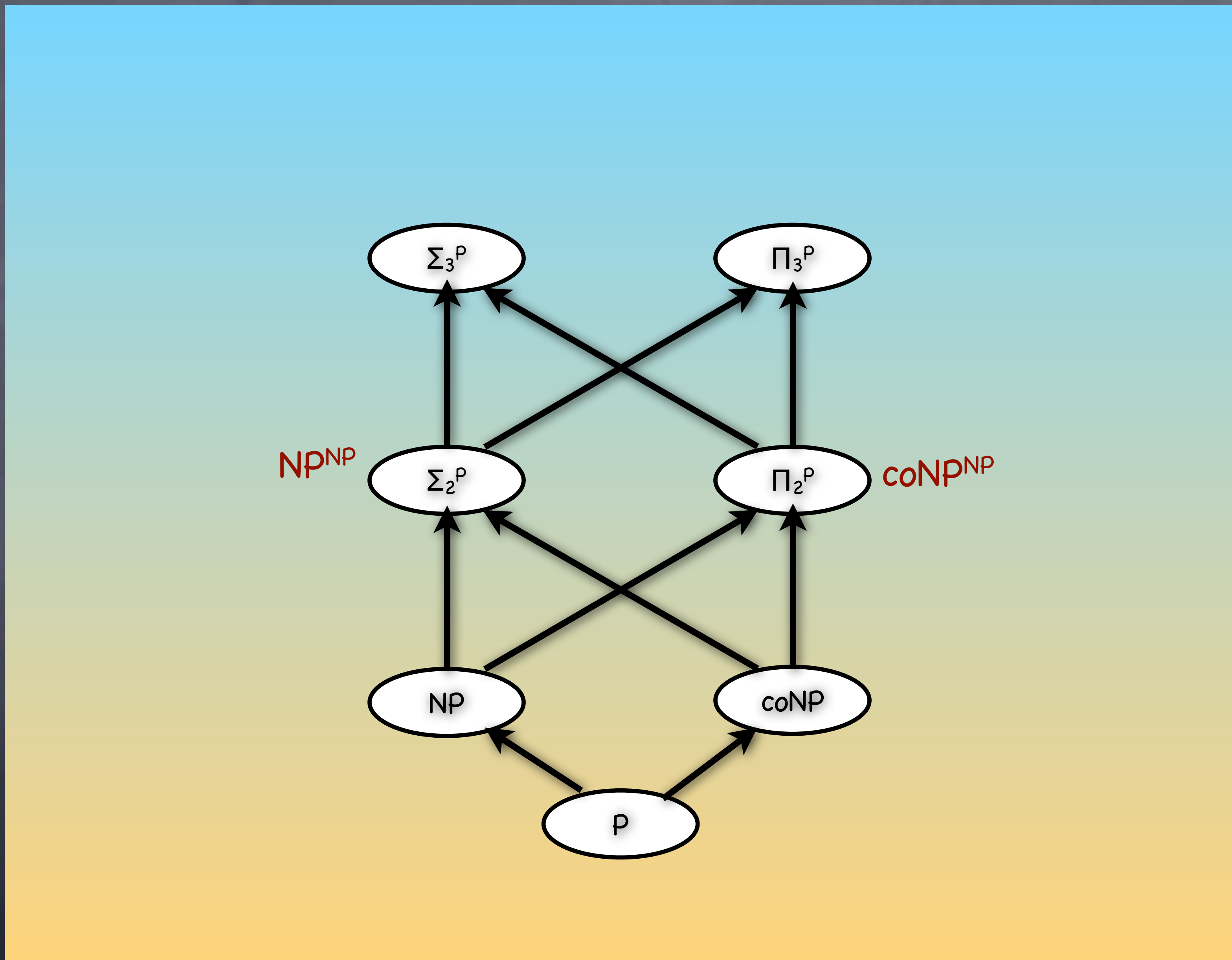
PH



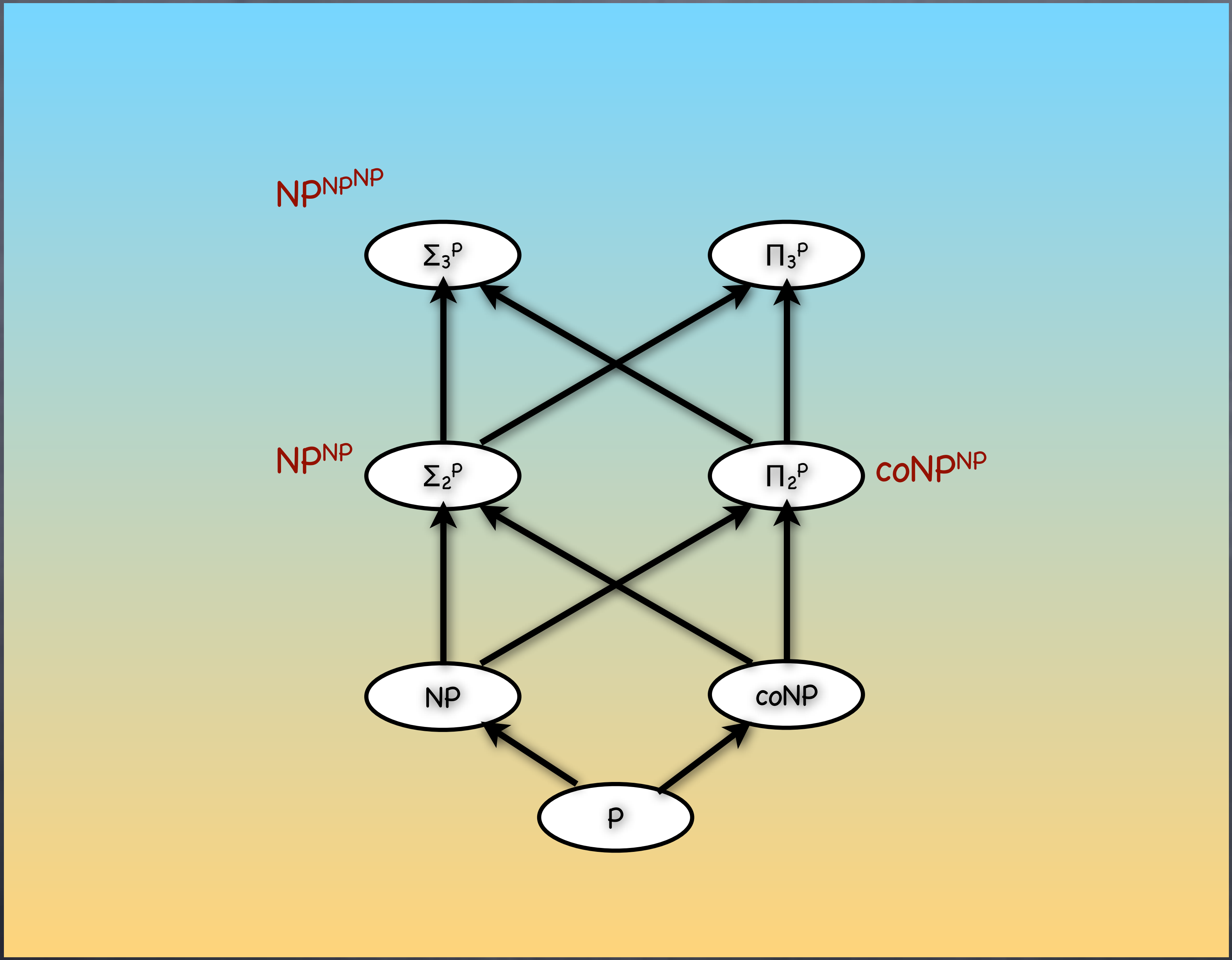
PH



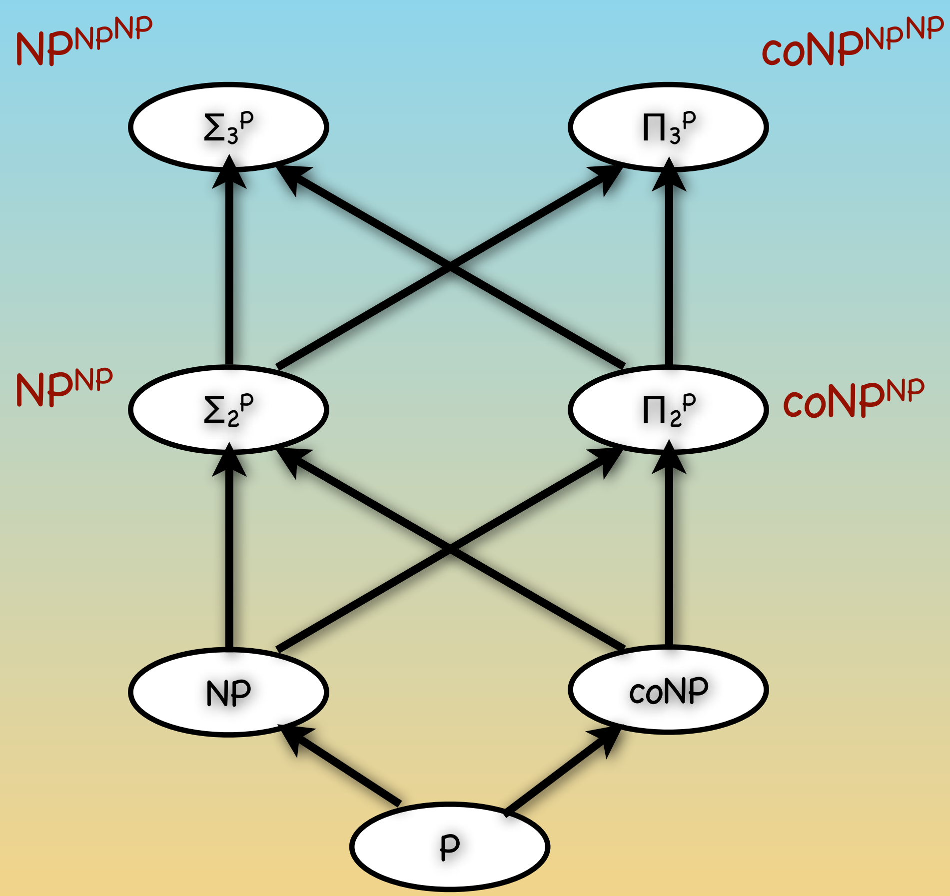
PH



PH

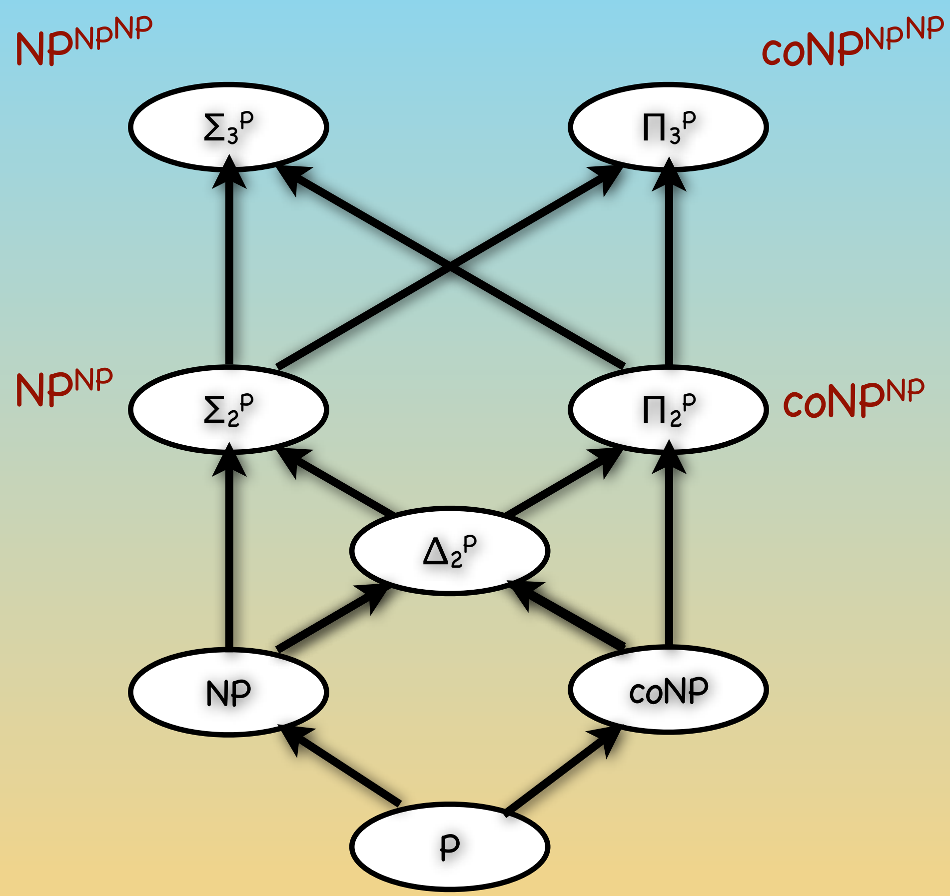


PH



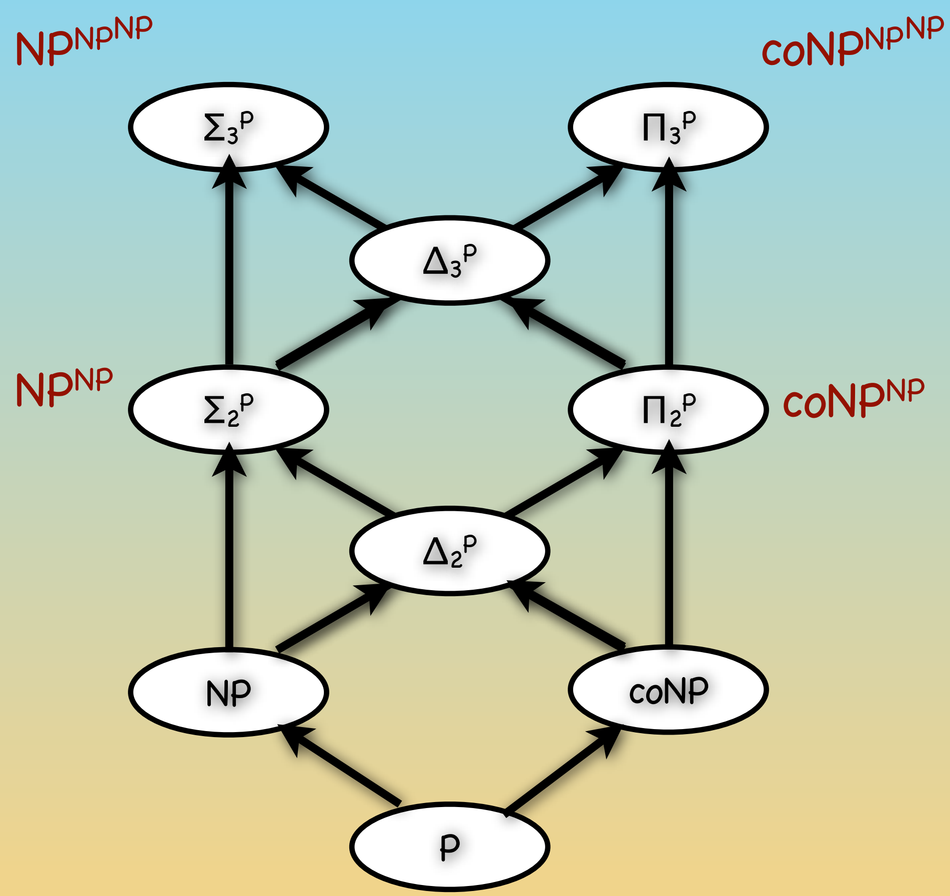
$NP^{NP^{NP}}$
$coNP^{NP^{NP}}$
Σ_3^P
Π_3^P
NP^{NP}
Σ_2^P
Π_2^P
$coNP^{NP}$
NP
coNP
P

PH



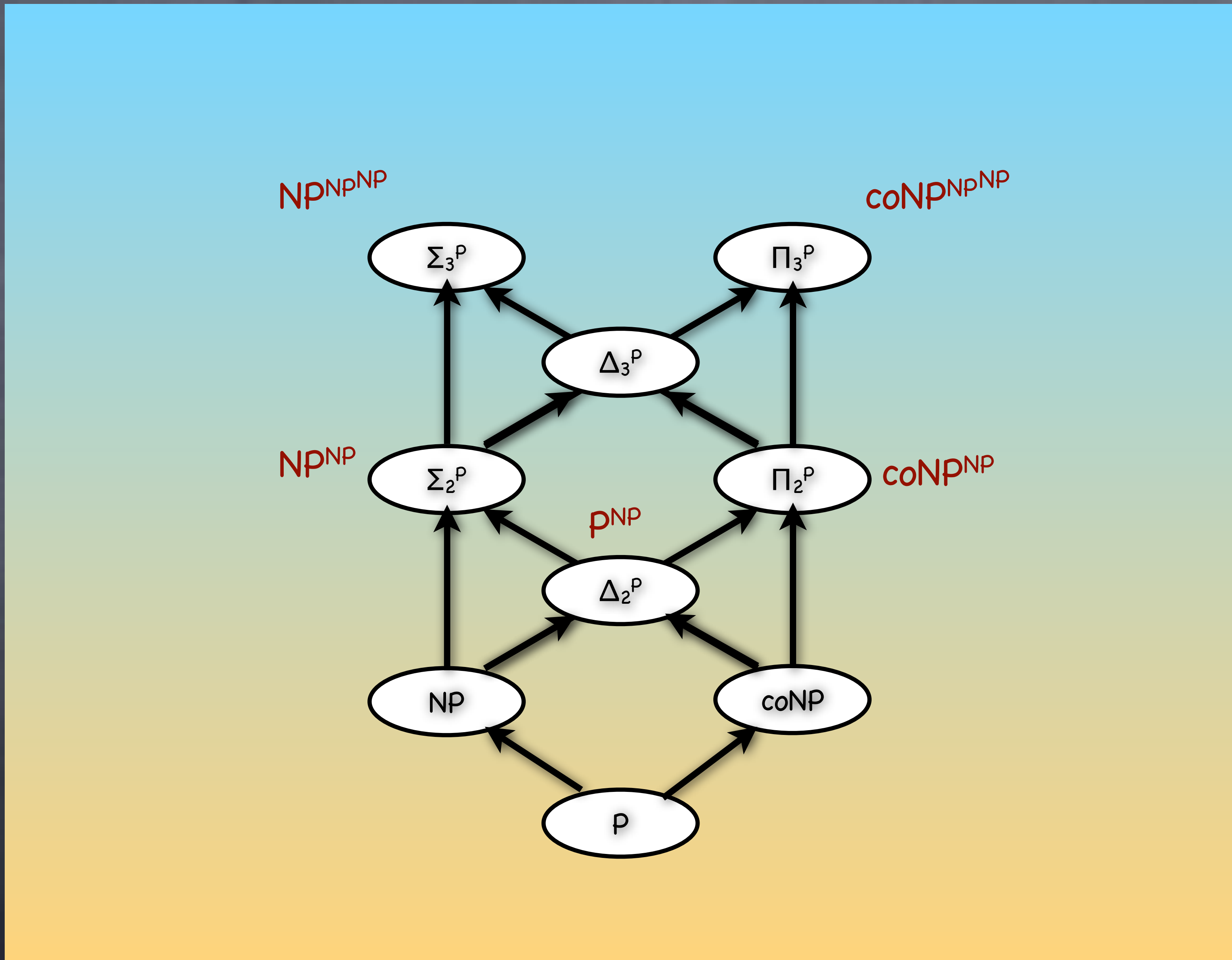
NP^NP^NP
coNP^NP^NP
Σ_3^P
Π_3^P
NP^NP
Σ_2^P
Π_2^P
coNP^NP
Δ_2^P
NP
coNP
P

PH



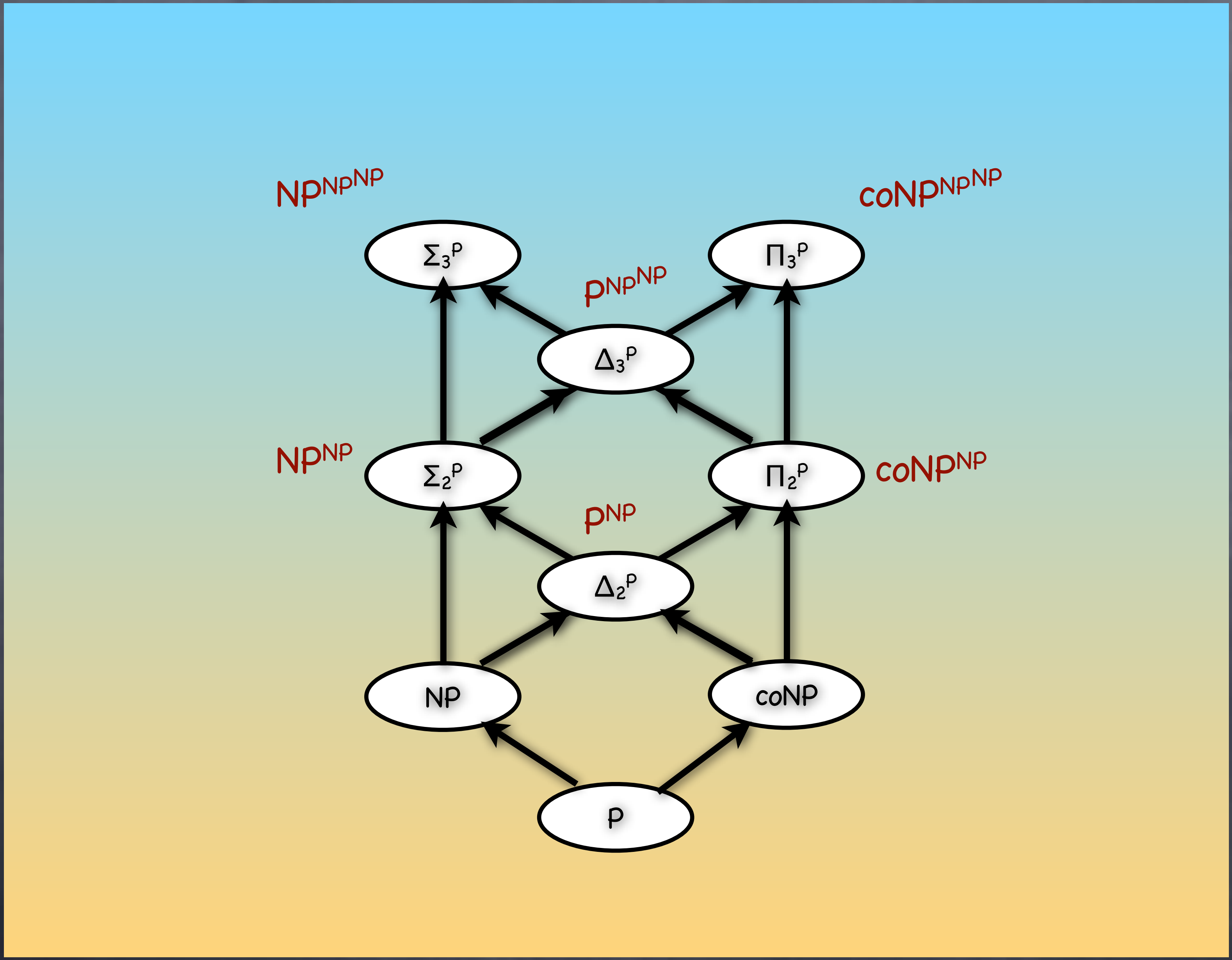
NP^NP^NP
coNP^NP^NP
Σ_3^P
Π_3^P
Δ_3^P
NP^NP
Σ_2^P
Π_2^P
coNP^NP
Δ_2^P
NP
coNP
P

PH



NP^NP^NP
coNP^NP^NP
Σ3P
Π3P
Δ3P
NP^NP
Σ2P
Π2P
coNP^NP
P^NP
Δ2P
NP
coNP
P

PH



NP^NP^NP
coNP^NP^NP
Σ_3^P
Π_3^P
P^NP^NP
Δ_3^P
NP^NP
Σ_2^P
Π_2^P
coNP^NP
P^NP
Δ_2^P
NP
coNP
P

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