The polynomial hierarchy

IIC3242

Motivation: Exact solutions

When studying NP we consider "existential" problems

- Is there a truth assignment satisfying a propositional formula?
- Is there a subset of a given set of numbers that sums up to a given target number?
- Is there a Hamiltonian path in a graph?

So does the complexity change once we ask **exact** questions?

Motivation: Exact solutions

When studying NP we consider "existential" problems

- Is there a truth assignment satisfying a propositional formula?
- Is there a subset of a given set of numbers that sums up to a given target number?
- Is there a Hamiltonian path in a graph?

So does the complexity change once we ask **exact** questions?

- Is there precisely one truth assignment satisfying a propositional formula?

Motivation: Exact solutions

When studying NP we consider "existential" problems

- Is there a truth assignment satisfying a propositional formula?
- Is there a subset of a given set of numbers that sums up to a given target number?
- Is there a Hamiltonian path in a graph?

So does the complexity change once we ask **exact** questions?

- Is there precisely one truth assignment satisfying a propositional formula?
- What is the complexity of this problem? Is the problem in NP?

Exact solutions: The class DP

To answer questions like that we need to define a new complexity class.

Idea behind this class: The notion of the exact solution can be reduced to two queries.

- Is it true that there is a truth assignment that satisfies the propositional formula φ?
- Is it true that there are not at least two truth assignment satisfying φ?

Exact solutions: The class DP

To answer questions like that we need to define a new complexity class.

Idea behind this class: The notion of the exact solution can be reduced to two queries.

- Is it true that there is a truth assignment that satisfies the propositional formula φ?
- Is it true that there are not at least two truth assignment satisfying φ?

Definition (The class DP)

$L \in DP$ if and only if there are two languages, $L_1 \in NP$ and $L_2 \in co\text{-}NP$, such that $L = L_1 \cap L_2$.

unique-SAT and the class DP

The first problem we will study is the following:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ is a propositional formula such that there exists precisely one truth assignment satisfying } \varphi\}$$

Exercise

Show that unique-SAT $\in$ DP.

unique-SAT and the class DP

The first problem we will study is the following:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ is a propositional formula such that there exists precisely one truth assignment satisfying } \varphi\}$$

Exercise

Show that unique-SAT $\in$ DP.

Is unique-SAT complete for DP?

unique-SAT and the class DP

The first problem we will study is the following:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ is a propositional formula such that there exists precisely one truth assignment satisfying } \varphi\}$$

Exercise

Show that unique-SAT $\in$ DP.

Is unique-SAT complete for DP?

- This is an open problem

unique-SAT and the class DP

The first problem we will study is the following:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ is a propositional formula such that there exists precisely one truth assignment satisfying } \varphi\}$$

Exercise

Show that unique-SAT $\in$ DP.

Is unique-SAT complete for DP?

- This is an open problem
- So are there complete problems for this class?

A DP-complete problem

Let 3-CNF-SAT-UNSAT be the following language:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ and } \psi \text{ are conjunctions of 3 literals, } \varphi \text{ is satisfiable and } \psi \text{ is not satisfiable}\}$$

A DP-complete problem

Let 3-CNF-SAT-UNSAT be the following language:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ and } \psi \text{ are conjunctions of 3 literals, } \varphi \text{ is satisfiable and } \psi \text{ is not satisfiable}\}$$

Theorem

3-CNF-SAT-UNSAT is DP-complete.

A DP-complete problem

Let 3-CNF-SAT-UNSAT be the following language:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ and } \psi \text{ are conjunctions of 3 literals, } \varphi \text{ is satisfiable and } \psi \text{ is not satisfiable}\}$$

Theorem

3-CNF-SAT-UNSAT is DP-complete.

Exercise

Prove the theorem.

A (natural) DP-complete problem

Notation

$$\textit{clique-number}(G) \quad = \quad \max\{k \mid G \textit{ has a clique of size } k\}$$

We use this number to define two problems about graphs:

$$\begin{aligned} \text{CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) = k\} \end{aligned}$$

A (natural) DP-complete problem

Notation

$$\textit{clique-number}(G) \quad = \quad \max\{k \mid G \textit{ has a clique of size } k\}$$

We use this number to define two problems about graphs:

$$\begin{aligned} \text{CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) = k\} \end{aligned}$$

What is the complexity of CLIQUE?

- Are the complexities of CLIQUE and exact-CLIQUE different?

A (natural) DP-complete problem

Theorem

exact-CLIQUE is DP-complete.

A (natural) DP-complete problem

Theorem

exact-CLIQUE is DP-complete.

Proof: First we need to show that exact-CLIQUE $\in$ DP

- How is this done?

Now we have to show that exact-CLIQUE is DP-hard.

A (natural) DP-complete problem

Theorem

exact-CLIQUE is DP-complete.

Proof: First we need to show that exact-CLIQUE $\in$ DP

- How is this done?

Now we have to show that exact-CLIQUE is DP-hard.

- We will reduce 3-CNF-SAT-UNSAT to exact-CLIQUE

exact-CLIQUE is DP-hard: the proof

We will use a reduction from 3-CNF-SAT to CLIQUE introduced earlier.

exact-CLIQUE is DP-hard: the proof

We will use a reduction from 3-CNF-SAT to CLIQUE introduced earlier.

- Let us recall this reduction using an example:

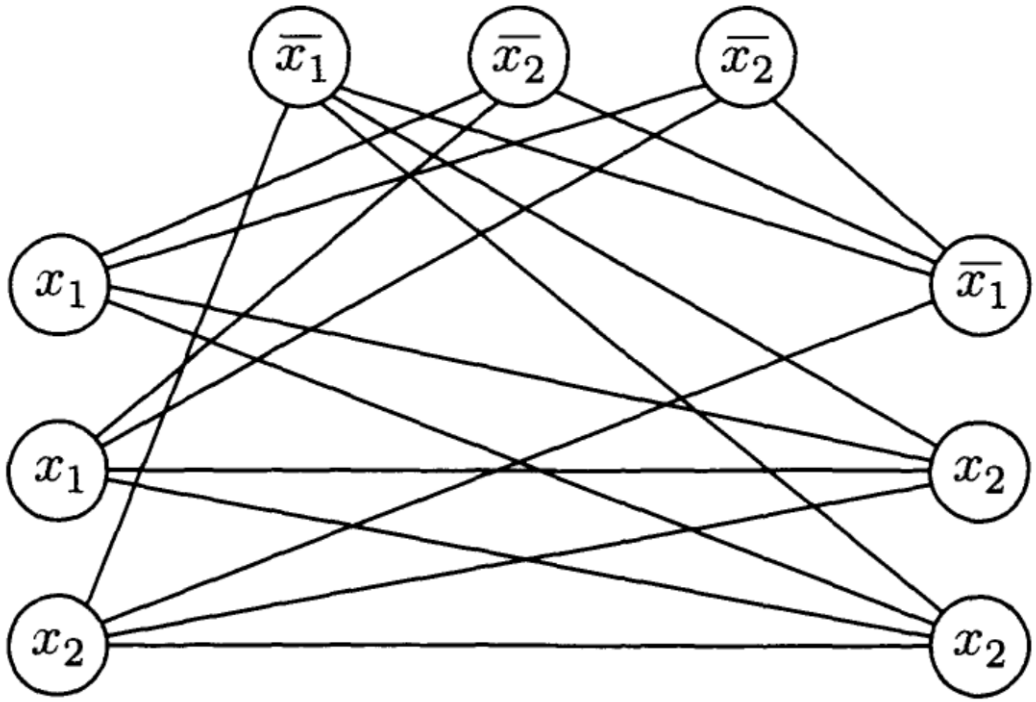
$$\beta \quad = \quad (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

exact-CLIQUE is DP-hard: the proof

We will use a reduction from 3-CNF-SAT to CLIQUE introduced earlier.

- Let us recall this reduction using an example:

$$\beta \quad = \quad (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



exact-CLIQUE is DP-hard: the proof

We have that: $\beta \in$ 3-CNF-SAT if and only if $(G_\beta, 3) \in$ CLIQUE

exact-CLIQUE is DP-hard: the proof

We have that: $\beta \in$ 3-CNF-SAT if and only if $(G_\beta, 3) \in$ CLIQUE

More generally, if $\varphi = C_1 \wedge \cdots \wedge C_n$, where each C_i is a clause, then:

$$\varphi \in \text{3-CNF-SAT} \quad \text{if and only if} \quad (G_\varphi, n) \in \text{CLIQUE}$$

exact-CLIQUE is DP-hard: the proof

We have that: $\beta \in$ 3-CNF-SAT if and only if $(G_\beta, 3) \in$ CLIQUE

More generally, if $\varphi = C_1 \wedge \cdots \wedge C_n$, where each C_i is a clause, then:

$$\varphi \in \text{3-CNF-SAT} \quad \text{if and only if} \quad (G_\varphi, n) \in \text{CLIQUE}$$

We will use this reduction in our proof.

- But first, we need to define some operations over graphs.

exact-CLIQUE is DP-hard: the proof

Given graphs $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$

exact-CLIQUE is DP-hard: the proof

Given graphs $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$

Notation

$G_1 \uplus G_2$, *the disjoint union of* G_1 *and* G_2, *is defined as*

$$G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$$

exact-CLIQUE is DP-hard: the proof

Given graphs $G_1 = (N_1, A_1)$ and $G_2 = (N_2, A_2)$

Notation

$G_1 \uplus G_2$, the disjoint union of G_1 and G_2, is defined as

$$G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$$

We are assuming that $N_1 \cap N_2 = \emptyset$.

If $N_1 \cap N_2 \neq \emptyset$: we can rename the nodes of G_2 in such a way that they are disjoint from N_1

exact-CLIQUE is DP-hard: the proof

Notation

$G_1 \times G_2 = (N_1 \times N_2, A)$, *where:*

$$\begin{aligned} A \;=\; \{((a_1,a_2),(b_1,b_2)) \mid\; & (a_1,b_1) \in A_1 \text{ and } (a_2,b_2) \in A_2, \text{ or} \\ & a_1 = b_1 \text{ and } (a_2,b_2) \in A_2, \text{ or} \\ & (a_1,b_1) \in A_1 \text{ y } a_2 = b_2\} \end{aligned}$$

exact-CLIQUE is DP-hard: the proof

Notation

$G_1 \times G_2 = (N_1 \times N_2, A)$, *where:*

$$A = \{((a_1, a_2), (b_1, b_2)) \mid (a_1, b_1) \in A_1 \text{ and } (a_2, b_2) \in A_2, \text{ or } a_1 = b_1 \text{ and } (a_2, b_2) \in A_2, \text{ or } (a_1, b_1) \in A_1 \text{ y } a_2 = b_2\}$$

Exercise

If G_1 and G_2 have cliques with n_1 and n_2 elements, respectively, than what do we know about cliques in $G_1 \times G_2$?

exact-CLIQUE is DP-hard: the proof

Now we have all the ingredients needed to reduce 3-CNF-SAT-UNSAT to exact-CLIQUE.

Given (φ, ψ), we will construct $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ such that:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{if and only if}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

exact-CLIQUE is DP-hard: the proof

Now we have all the ingredients needed to reduce 3-CNF-SAT-UNSAT to exact-CLIQUE.

Given (φ, ψ), we will construct $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ such that:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{if and only if}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

Assume that φ and ψ consist of m and n clauses, respectively, where $m \geq 2$, $n \geq 2$ and $m \neq n$.

exact-CLIQUE is DP-hard: the proof

Now we have all the ingredients needed to reduce 3-CNF-SAT-UNSAT to exact-CLIQUE.

Given (φ, ψ), we will construct $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ such that:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{if and only if}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

Assume that φ and ψ consist of m and n clauses, respectively, where $m \geq 2$, $n \geq 2$ and $m \neq n$.

- Why can we assume that $m \geq 2$, $n \geq 2$ and $m \neq n$?

exact-CLIQUE is DP-hard: the proof

Let:

$$\begin{aligned} G_{(\varphi,\psi)} &= (G_\varphi \uplus K_{m-1}) \times (G_\psi \uplus K_{n-1}) \\ k_{(\varphi,\psi)} &= m \cdot (n-1) \end{aligned}$$

exact-CLIQUE is DP-hard: the proof

Let:

$$\begin{aligned} G_{(\varphi,\psi)} &= (G_\varphi \uplus K_{m-1}) \times (G_\psi \uplus K_{n-1}) \\ k_{(\varphi,\psi)} &= m \cdot (n-1) \end{aligned}$$

Then the following holds:

φ	ψ	The size of the largest clique in $G_{(\varphi,\psi)}$
CNF-SAT	CNF-SAT	$m \cdot n$
CNF-SAT	$\overline{\text{CNF-SAT}}$	$m \cdot (n-1)$
$\overline{\text{CNF-SAT}}$	CNF-SAT	$(m-1) \cdot n$
$\overline{\text{CNF-SAT}}$	$\overline{\text{CNF-SAT}}$	$(m-1) \cdot (n-1)$

exact-CLIQUE is DP-hard: the proof

Since $m \neq n$: $m \cdot (n-1) \neq (m-1) \cdot n$

Therefore, since $(m-1) \cdot (n-1) < m \cdot (n-1) < m \cdot n$, we can conclude that:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{if and only if}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

exact-CLIQUE is DP-hard: the proof

Since $m \neq n$: $m \cdot (n-1) \neq (m-1) \cdot n$

Therefore, since $(m-1) \cdot (n-1) < m \cdot (n-1) < m \cdot n$, we can conclude that:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{if and only if}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

To finish the proof we need to check that the reduction is doable in PTIME (or LOGSPACE if you prefer).

- How can we show this?

Relation between NP and DP

It is easy to see that NP $\subseteq$ DP and co-NP $\subseteq$ DP.

- How can we show this?

What is the relation between NP and DP?

- Is NP $\neq$ DP?

Relation between NP and DP

It is easy to see that NP $\subseteq$ DP and co-NP $\subseteq$ DP.

- How can we show this?

What is the relation between NP and DP?

- Is NP $\neq$ DP?
- This is an open problem, and there is a very good reason for this!

Relation between NP and DP

Theorem

NP = DP if and only if NP = co-NP.

Relation between NP and DP

Theorem

NP = DP if and only if NP = co-NP.

Proof: ($\Leftarrow$) Assume first that NP = co-NP.

If $L \in$ DP: $L = L_1 \cap L_2$, with $L_1 \in$ NP and $L_2 \in$ co-NP.

- By the assumption: $L_2 \in$ NP

Therefore: $L = L_1 \cap L_2$, where $L_1, L_2 \in$ NP.

But NP is closed under intersection.

- How does this follow?
- Thus, we have that $L \in$ NP

Relation between NP and DP

($\Rightarrow$) Assume that NP = DP.

Since co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Therefore we have that NP $\subseteq$ co-NP:

Relation between NP and DP

($\Rightarrow$) Assume that NP = DP.

Since co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Therefore we have that NP $\subseteq$ co-NP:

$L \in$ NP

Relation between NP and DP

($\Rightarrow$) Assume that NP = DP.

Since co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Therefore we have that NP $\subseteq$ co-NP:

$$L \in \text{NP} \quad \Rightarrow \quad \overline{L} \in \text{co-NP}$$

Relation between NP and DP

($\Rightarrow$) Assume that NP = DP.

Since co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Therefore we have that NP $\subseteq$ co-NP:

$$\begin{aligned} L \in \text{NP} &\Rightarrow \overline{L} \in \text{co-NP} \\ &\Rightarrow \overline{L} \in \text{NP} \qquad \text{(we showed that co-NP} \subseteq \text{NP)} \end{aligned}$$

Relation between NP and DP

($\Rightarrow$) Assume that NP = DP.

Since co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Therefore we have that NP $\subseteq$ co-NP:

$$
\begin{array}{lcll}
L \in \text{NP} & \Rightarrow & \overline{L} \in \text{co-NP} & \\
 & \Rightarrow & \overline{L} \in \text{NP} & \text{(we showed that co-NP} \subseteq \text{NP)} \\
 & \Rightarrow & L \in \text{co-NP} &
\end{array}
$$

Practical motivation: Optimisation problems

A tour π in a graph G is a sequence of edges (a_1, a_2), ..., (a_{k-1}, a_k), (a_k, a_1) in G such that:

- $a_i \neq a_j$ for each $i \neq j$,
- $\{a_1, \ldots, a_k\}$ is the set of all nodes in G.
- (Recall Hamiltonian paths)

Notation

- *A cost function for a graph $G = (N, A)$ is a function $cost : A \to \mathbb{N}$.*
- *The cost of a tour π in G is defined as:*

$$cost(\pi) \quad = \quad \left(\sum_{i=1}^{k-1} cost((a_i, a_{i+1})) \right) + cost((a_k, a_1))$$

Practical motivation: Optimisation problems

The travelling salesman problem is defined as:

$$\text{TSP} = \{(G, \text{cost}, k) \mid \text{there is a tour } \pi \text{ in } G \text{ with } \text{cost}(\pi) \leq k\}$$

What is the complexity of TSP?

Practical motivation: Optimisation problems

The travelling salesman problem is defined as:

$$\text{TSP} = \{(G, \text{cost}, k) \mid \text{there is a tour } \pi \text{ in } G \text{ with } \text{cost}(\pi) \leq k\}$$

What is the complexity of TSP?

Theorem

TSP is NP-complete.

Practical motivation: Optimisation problems

TSP is a decision problem.

Practical motivation: Optimisation problems

TSP is a decision problem.

- There is also an optimisation variant of TSP: Find the smallest k such that $(G, \text{cost}, k) \in \text{TSP}$

Practical motivation: Optimisation problems

TSP is a decision problem.

- There is also an optimisation variant of TSP: Find the smallest k such that $(G, \text{cost}, k) \in \text{TSP}$
- The optimisation variant of the problem is at least as difficult as the decision variant

Practical motivation: Optimisation problems

TSP is a decision problem.

- There is also an optimisation variant of TSP: Find the smallest k such that $(G, \text{cost}, k) \in \text{TSP}$
- The optimisation variant of the problem is at least as difficult as the decision variant
- If we can solve TSP in polynomial time, can we also solve its optimisation variant in polynomial time?

Practical motivation: Optimisation problems

TSP is a decision problem.

- There is also an optimisation variant of TSP: Find the smallest k such that $(G, \text{cost}, k) \in \text{TSP}$
- The optimisation variant of the problem is at least as difficult as the decision variant
- If we can solve TSP in polynomial time, can we also solve its optimisation variant in polynomial time?
  - TSP has to be used polynomially many times

Practical motivation: Optimisation problems

a tour is optimal if there is no other tour with a smaller cost.

- We can have more than one optimal tour

Another version of TSP:

$$\text{unique-TSP} = \{(G, \text{cost}) \mid \text{there is just one optimal tour for } G\}$$

This is a decision problem, and it seems more difficult than TSP.

Practical motivation: Optimisation problems

a tour is optimal if there is no other tour with a smaller cost.

- We can have more than one optimal tour

Another version of TSP:

$$\text{unique-TSP} = \{(G, \text{cost}) \mid \text{there is just one optimal tour for } G\}$$

This is a decision problem, and it seems more difficult than TSP.

- If we can solve TSP in polynomial time, can we also solve unique-TSP in polynomial time?

Practical motivation: Optimisation problems

a tour is optimal if there is no other tour with a smaller cost.

- We can have more than one optimal tour

Another version of TSP:

$$\text{unique-TSP} = \{(G, \text{cost}) \mid \text{there is just one optimal tour for } G\}$$

This is a decision problem, and it seems more difficult than TSP.

- If we can solve TSP in polynomial time, can we also solve unique-TSP in polynomial time?
  - Again, TSP has to be used polynomially many times

Oracles make a comeback

What do all these problems have in common?

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times
  - In the case of DP: two times

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times
  - In the case of DP: two times
- If we find a polynomial algorithm for some NP-complete problem, then these problems can be solved in polynomial time

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times
  - In the case of DP: two times
- If we find a polynomial algorithm for some NP-complete problem, then these problems can be solved in polynomial time
- An NP-complete problem can be seen as an *oracle* for these problems

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times
  - In the case of DP: two times
- If we find a polynomial algorithm for some NP-complete problem, then these problems can be solved in polynomial time
- An NP-complete problem can be seen as an *oracle* for these problems

So we will recall the notion of an oracle machine.

Oracles make a comeback

What do all these problems have in common?

- We can solve the using an answer to a NP problem polynomially many times
  - In the case of DP: two times
- If we find a polynomial algorithm for some NP-complete problem, then these problems can be solved in polynomial time
- An NP-complete problem can be seen as an *oracle* for these problems

So we will recall the notion of an oracle machine.

- We will show that oracles can be used to characterise these problems, and many more interesting problems

Oracle machines

Definition

A deterministic TM with input tape and an oracle for $A \subseteq \Sigma^*$:
$M^A = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

- Q a finite set of states with $q_?, q_{yes}, q_{no} \in Q$
- Σ a finite alphabet with $\mathtt{B} \notin \Sigma$
- Γ a finite alphabet with $\Sigma \cup \{\mathtt{B}, \vdash\} \subseteq \Gamma$
- $q_0, q_{accept}, q_{reject} \in Q$ as before
- δ is a partial function:

$$\delta \quad : \quad Q \times \Gamma \times \Gamma^2 \to Q \times \{\leftarrow, \square, \rightarrow\} \times \Gamma^2 \times \{\leftarrow, \square, \rightarrow\}^2$$

The third tape is the **query tape**

A is called the **oracle**

How do oracle machines work?

Easy to extend the previous definition to multiple work tapes and to non-determinism.

An oracle machine M^A work as an ordinary TM, but when it enters the state $q_?$:

- The query tape has: $w\text{BB}\cdots$, for some $w \in \Sigma^*$
- M uses the oracle for A, and its next state is q_{yes} or q_{no}
  - The state is q_{yes} iff $w \in A$

Oracle machines: examples

The execution time is defined as for ordinary TMs

- **A query to the oracle counts as one step**

For our initial examples:

Oracle machines: examples

The execution time is defined as for ordinary TMs

- **A query to the oracle counts as one step**

For our initial examples:

3-CNF-SAT-UNSAT : is accepted by M^{SAT} that runs in time $O(n)$

Oracle machines: examples

The execution time is defined as for ordinary TMs

- **A query to the oracle counts as one step**

For our initial examples:

3-CNF-SAT-UNSAT	:	is accepted by M^{SAT} that runs in time $O(n)$
exact-CLIQUE	:	is accepted by M^{CLIQUE} that runs in time $O(n)$

Oracle machines: examples

The execution time is defined as for ordinary TMs

- **A query to the oracle counts as one step**

For our initial examples:

3-CNF-SAT-UNSAT	:	is accepted by M^{SAT} that runs in time $O(n)$
exact-CLIQUE	:	is accepted by M^{CLIQUE} that runs in time $O(n)$
unique-TSP	:	is accepted by M^{TSP} that runs in time $O(n^k)$

Complexity classes with an oracle

A complexity class defined using oracle machines:

Definition

$PTIME^A$: All languages L such that there exists an oracle machine M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$.

Complexity classes with an oracle

A complexity class defined using oracle machines:

Definition

$PTIME^A$: All languages L such that there exists an oracle machine M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$.

We have already seen that:

Complexity classes with an oracle

A complexity class defined using oracle machines:

Definition

$PTIME^A$: All languages L such that there exists an oracle machine M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$.

We have already seen that:

- 3-CNF-SAT-UNSAT, exact-CLIQUE and unique-TSP are in $\text{PTIME}^{\text{SAT}}$

Complexity classes with an oracle

A complexity class defined using oracle machines:

Definition

$PTIME^A$: All languages L such that there exists an oracle machine M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$.

We have already seen that:

- 3-CNF-SAT-UNSAT, exact-CLIQUE and unique-TSP are in $\text{PTIME}^{\text{SAT}}$
  - They are also in PTIME^A, for any NP-complete problem A

Complexity classes with an oracle

A complexity class defined using oracle machines:

Definition

$PTIME^A$: All languages L such that there exists an oracle machine M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$.

We have already seen that:

- 3-CNF-SAT-UNSAT, exact-CLIQUE and unique-TSP are in $\text{PTIME}^{\text{SAT}}$
  - They are also in PTIME^A, for any NP-complete problem A
- NP, co-NP and DP are contained in $\text{PTIME}^{\text{SAT}}$

Complexity classes with an oracle

A more general definition:

Definition

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

Complexity classes with an oracle

A more general definition:

Definition

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

But in reality, this is not more general than:

Proposition

$PTIME^{NP} = PTIME^{SAT}$

Complexity classes with an oracle

A more general definition:

Definition

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

But in reality, this is not more general than:

Proposition

$PTIME^{NP} = PTIME^{SAT}$

Exercise

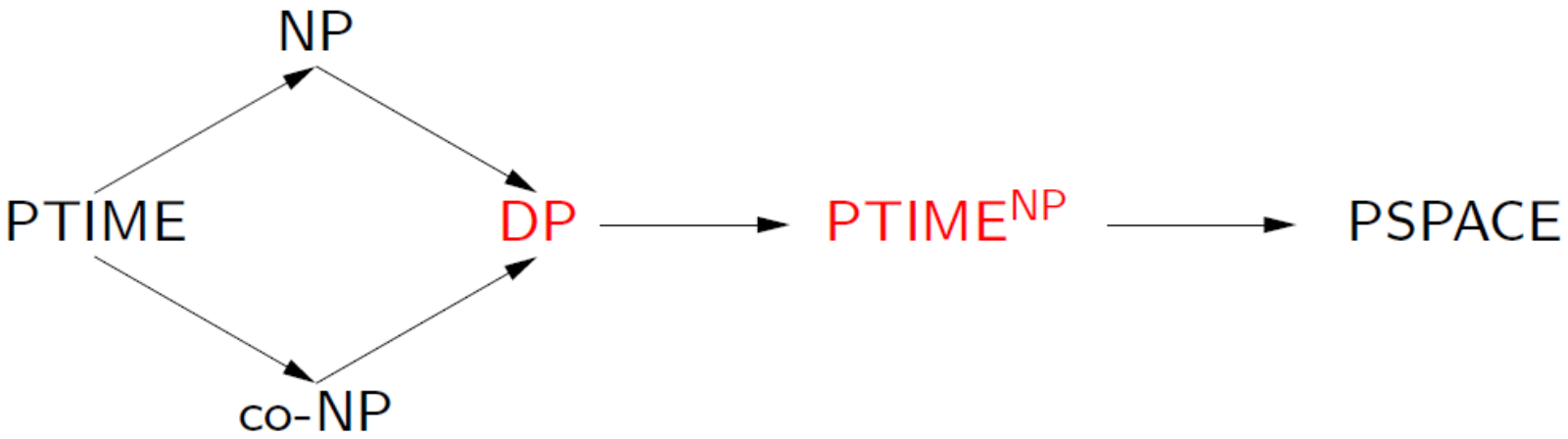
Prove the proposition.

Relationships between complexity classes

What is the relations between PTIME^{NP} and the classes we defined so far?

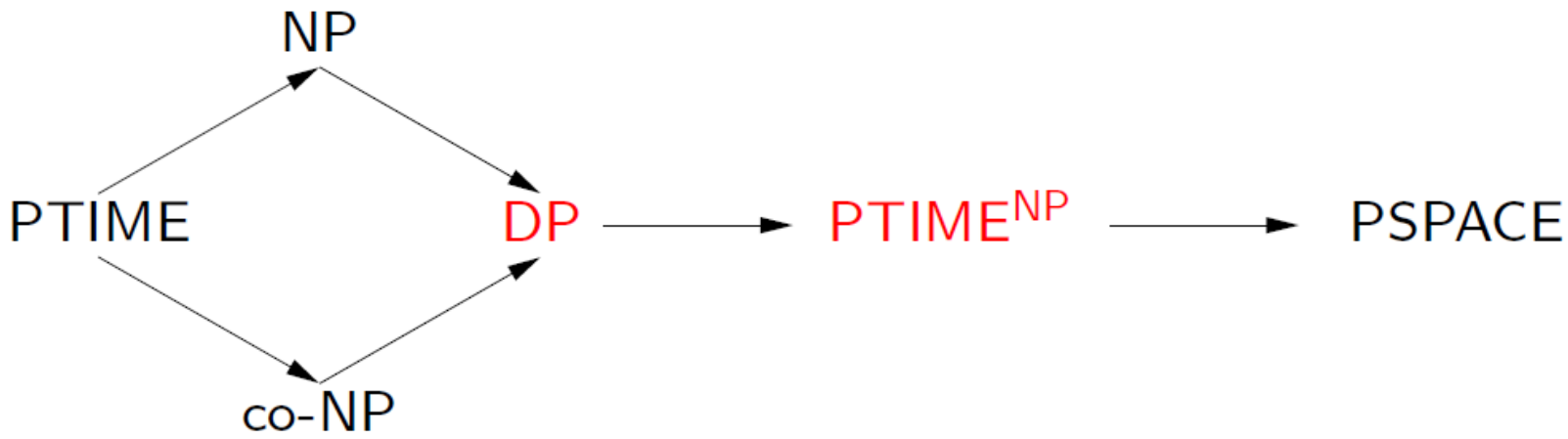
Relationships between complexity classes

What is the relations between PTIME^{NP} and the classes we defined so far?



Relationships between complexity classes

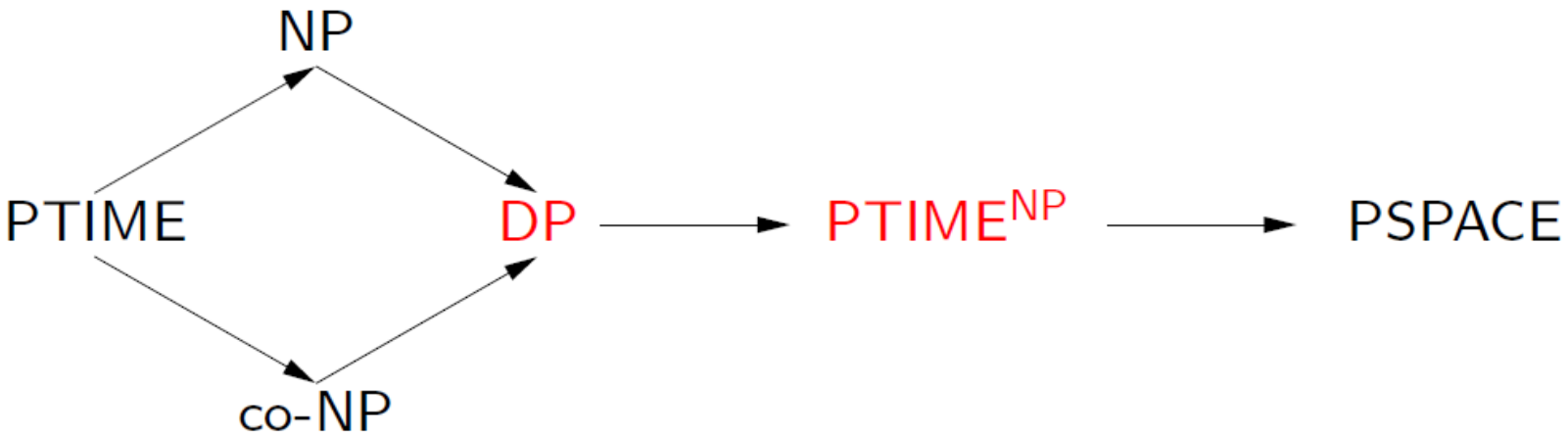
What is the relations between PTIME^{NP} and the classes we defined so far?



Why does it hold that $\text{PTIME}^{\text{NP}} \subseteq \text{PSPACE}$? (Recall limits of diagonalisation)

Relationships between complexity classes

What is the relations between PTIME^{NP} and the classes we defined so far?



Why does it hold that $\text{PTIME}^{\text{NP}} \subseteq \text{PSPACE}$? (Recall limits of diagonalisation)

Do we have enough complexity classes in our life?

- Sorry, but there are many other natural problems between PTIME^{NP} and PSPACE

A problem from Artificial Intelligence: Knowledge revision

A fundamental problem: Knowledge update.

- Given a knowledge base Σ and a new fact φ, we want to update the knowledge in Σ with respect to the new fact φ

A problem from Artificial Intelligence: Knowledge revision

A fundamental problem: Knowledge update.

- Given a knowledge base Σ and a new fact φ, we want to update the knowledge in Σ with respect to the new fact φ
- We want to update what is *necessary*

A problem from Artificial Intelligence: Knowledge revision

A fundamental problem: Knowledge update.

- Given a knowledge base Σ and a new fact φ, we want to update the knowledge in Σ with respect to the new fact φ
- We want to update what is *necessary*
- We do not want to lose the information that we do not need to eliminate

A problem from Artificial Intelligence: Knowledge revision

A fundamental problem: Knowledge update.

- Given a knowledge base Σ and a new fact φ, we want to update the knowledge in Σ with respect to the new fact φ
- We want to update what is *necessary*
- We do not want to lose the information that we do not need to eliminate

Assumptions:

- Σ is a set of propositional formulas
- φ is a propositional formula

Knowledge revision

Given Σ and φ: we want to generate a formula that describes the update of Σ with respect to φ.

Notation

$\Sigma \circ \varphi$

Knowledge revision

Given Σ and φ: we want to generate a formula that describes the update of Σ with respect to φ.

Notation

$\Sigma \circ \varphi$

How can we do this?

- What would be $\{p, p \rightarrow q\} \circ \neg q$?

Knowledge revision

A possible solution: Belief Revision

Knowledge revision

A possible solution: Belief Revision

Notation

- *models*(Σ): *the set of valuations* σ *that satisfy* Σ
- $\Delta(\sigma_1, \sigma_2)$: *the set of propositional variables* p *such that* $\sigma_1(p) \neq \sigma_2(p)$
  - *We assume all the valuation to have the same domain*

Knowledge revision

A possible solution: Belief Revision

Notation

- *models*(Σ): *the set of valuations* σ *that satisfy* Σ
- $\Delta(\sigma_1, \sigma_2)$: *the set of propositional variables* p *such that* $\sigma_1(p) \neq \sigma_2(p)$
  - *We assume all the valuation to have the same domain*

Example

If $\sigma_1(p) = 1$, $\sigma_1(q) = 1$, $\sigma_2(p) = 1$ and $\sigma_2(q) = 0$, then $\Delta(\sigma_1, \sigma_2) = \{q\}$

- $\Delta(\sigma_1, \sigma_2)$ measures the *distance* between σ_1 and σ_2

Knowledge revision

To update Σ with respect to φ: We update the models of Σ with respect to φ.

For any σ such that $\sigma(\Sigma) = 1$, we want to select the models σ_1 of φ that are at a minimal distance from σ.

Knowledge revision

To update Σ with respect to φ: We update the models of Σ with respect to φ.

For any σ such that $\sigma(\Sigma) = 1$, we want to select the models σ_1 of φ that are at a minimal distance from σ.

Notation

$$min(\sigma, \varphi) = \{\sigma_1 \mid \sigma_1(\varphi) = 1 \text{ and there is no } \sigma_2 \text{ such that } \sigma_2(\varphi) = 1 \text{ and } \Delta(\sigma, \sigma_2) \subsetneq \Delta(\sigma, \sigma_1)\}$$

Knowledge revision

We define the models of $\Sigma \circ \varphi$ as the models of φ that are *closest* to the models of Σ:

$$models(\Sigma \circ \varphi) \quad = \quad \bigcup_{\sigma \,:\, \sigma(\Sigma)=1} min(\sigma, \varphi)$$

and we define $\Sigma \circ \varphi$ as any formula ψ such that $models(\psi) = models(\Sigma \circ \varphi)$.

Knowledge revision

We define the models of $\Sigma \circ \varphi$ as the models of φ that are *closest* to the models of Σ:

$$models(\Sigma \circ \varphi) \quad = \quad \bigcup_{\sigma \,:\, \sigma(\Sigma)=1} min(\sigma, \varphi)$$

and we define $\Sigma \circ \varphi$ as any formula ψ such that $models(\psi) = models(\Sigma \circ \varphi)$.

- Does this formula always exist? Is it unique?

Knowledge revision

We define the models of $\Sigma \circ \varphi$ as the models of φ that are *closest* to the models of Σ:

$$models(\Sigma \circ \varphi) \quad = \quad \bigcup_{\sigma \,:\, \sigma(\Sigma)=1} min(\sigma, \varphi)$$

and we define $\Sigma \circ \varphi$ as any formula ψ such that $models(\psi) = models(\Sigma \circ \varphi)$.

- Does this formula always exist? Is it unique?
- What is the complexity of determining if $\sigma \in models(\Sigma \circ \varphi)$?

Knowledge revision

Example

$\Sigma = \{p, p \to q\}$ and $\varphi = \neg q$

$$
\begin{array}{rcll}
models(\Sigma) & = & \{\sigma\}, & \sigma(p) = \sigma(q) = 1 \\
models(\varphi) & = & \{\sigma_1, \sigma_2\}, & \sigma_1(p) = 1,\ \sigma_1(q) = 0 \\
 & & & \sigma_2(p) = 0,\ \sigma_2(q) = 0
\end{array}
$$

Minimal models:

$$
\begin{array}{rcl}
\Delta(\sigma, \sigma_1) & = & \{q\} \\
\Delta(\sigma, \sigma_2) & = & \{p, q\} \\
min(\sigma, \varphi) & = & \{\sigma_1\} \\
models(\Sigma \circ \varphi) & = & \{\sigma_1\}
\end{array}
$$

Result: $\{p, p \to q\} \circ \neg q \quad = \quad p \wedge \neg q$

Knowledge revision: complexity

The main problem in the area: compute *certain answers*.

- ψ is a certain answer to $\Sigma \circ \varphi$ if for every $\sigma \in \mathit{models}(\Sigma \circ \varphi)$, it is true that σ satisfies ψ

Example

p is a certain answer to $\{p, p \rightarrow q\} \circ \neg q$.

Knowledge revision: complexity

The main problem in the area: compute *certain answers*.

- ψ is a certain answer to $\Sigma \circ \varphi$ if for every $\sigma \in \mathit{models}(\Sigma \circ \varphi)$, it is true that σ satisfies ψ

Example

p is a certain answer to $\{p, p \rightarrow q\} \circ \neg q$.

The problem we need to study:

$$\text{CERTAIN-ANSWERS} = \{(\Sigma, \varphi, \psi) \mid \psi \text{ is a certain answer to } \Sigma \circ \varphi\}$$

Knowledge revision: complexity

What is the complexity of CERTAIN-ANSWERS?

Knowledge revision: complexity

What is the complexity of CERTAIN-ANSWERS?

- Is it in NP? In co-NP?

Knowledge revision: complexity

What is the complexity of CERTAIN-ANSWERS?

- Is it in NP? In co-NP?
- Is it in PTIME^{NP}?

Knowledge revision: complexity

What is the complexity of CERTAIN-ANSWERS?

- Is it in NP? In co-NP?
- Is it in PTIME^{NP}?
- Or at least in PSPACE?

Knowledge revision: complexity

What is the complexity of CERTAIN-ANSWERS?

- Is it in NP? In co-NP?
- Is it in PTIME^{NP}?
- Or at least in PSPACE?

Again, the notion of an oracle can help us to understand the complexity of this problem.

Another complexity class defined with respect to oracles

Definition

- NP^A: *Languages L for which there is a nondeterministic TM M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

Another complexity class defined with respect to oracles

Definition

- NP^A: *Languages L for which there is a nondeterministic TM M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

With this we can show where CERTAIN-ANSWERS is.

Another complexity class defined with respect to oracles

Definition

- NP^A: *Languages L for which there is a nondeterministic TM M^A with $L = L(M^A)$ and M^A runs in time $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

With this we can show where CERTAIN-ANSWERS is.

Exercise

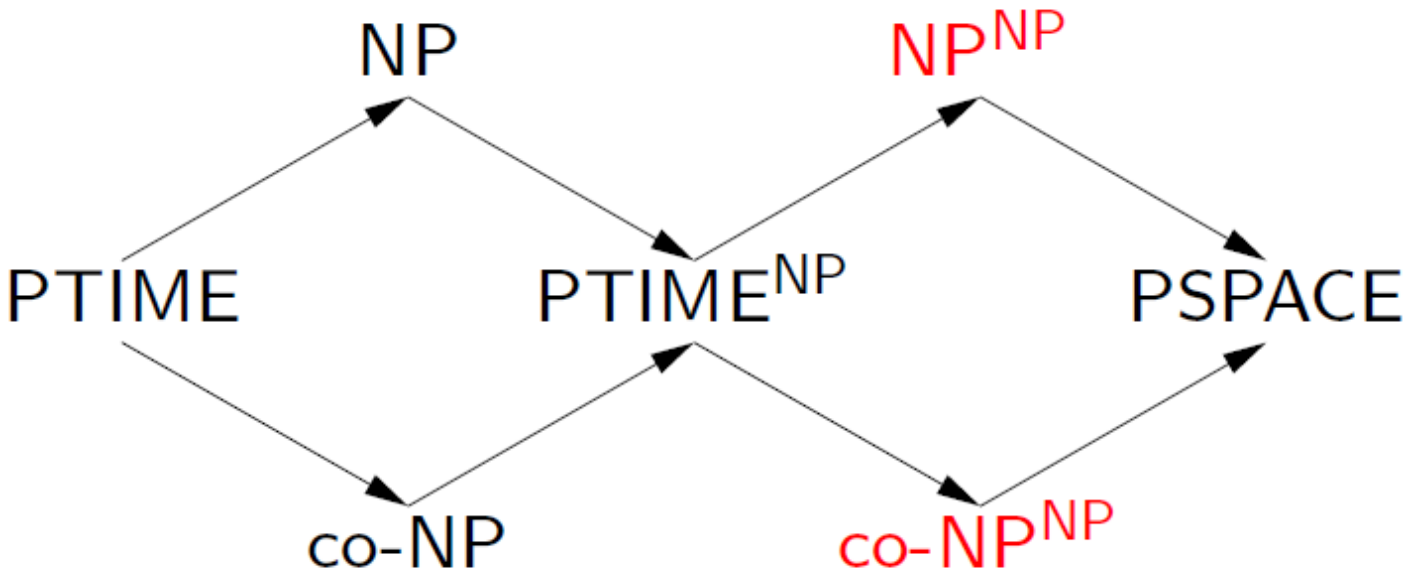
Show that CERTAIN-ANSWERS $\in$ co-NP^{NP}. (Hint: to show that ψ is **not** a certain answer, we guess a valuation that confirms this. But to check that this is a model of the update we need NP.)

Where are we now?

What is the relationship between NP^{NP} and the other classes we defined?

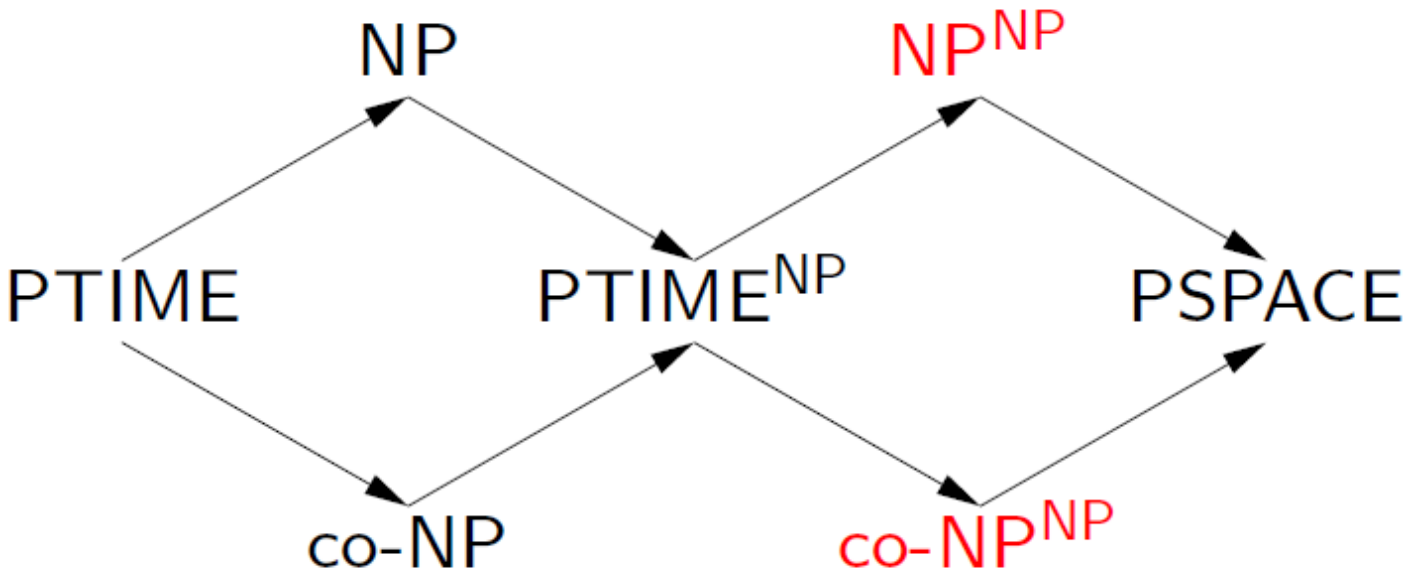
Where are we now?

What is the relationship between NP^{NP} and the other classes we defined?



Where are we now?

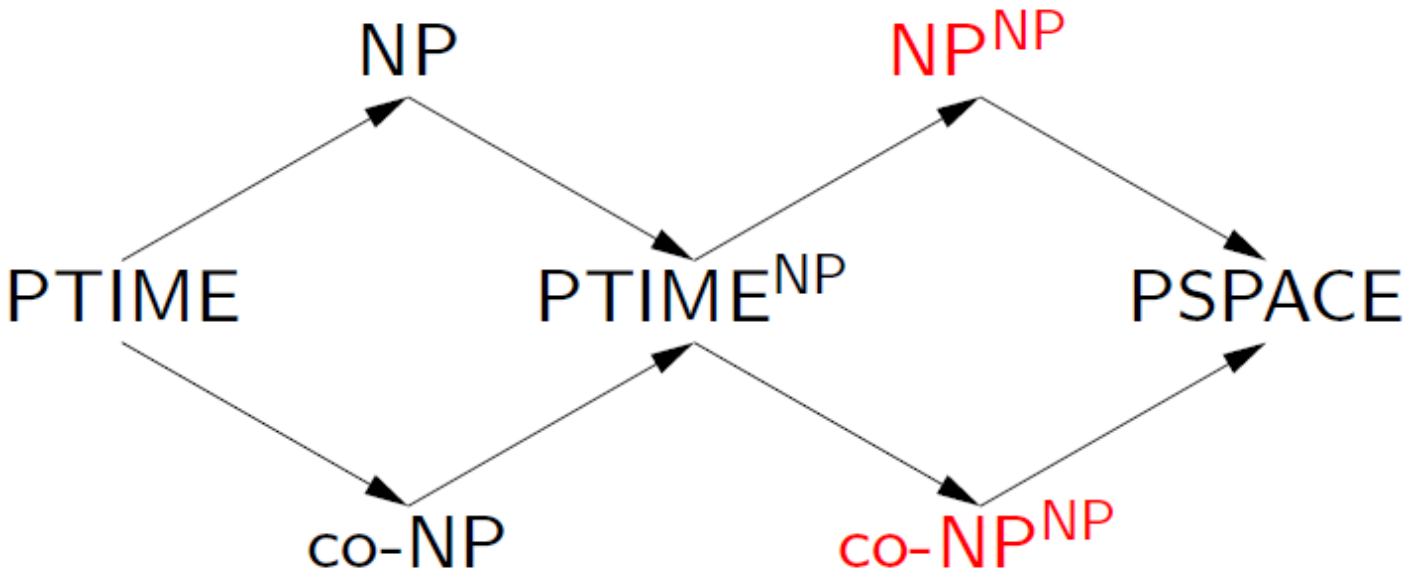
What is the relationship between NP^{NP} and the other classes we defined?



Why is it true that $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$?

Where are we now?

What is the relationship between NP^{NP} and the other classes we defined?



Why is it true that $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$?

We can generalise this construction.

- So we define the polynomial hierarchy

The polynomial hierarchy

We can generalise the previous definitions with respect to any complexity class $\mathcal{C}$.

The polynomial hierarchy

We can generalise the previous definitions with respect to any complexity class $\mathcal{C}$.

Definition

- $PTIME^{\mathcal{C}}$: $\bigcup_{A \in \mathcal{C}} PTIME^{A}$
- $NP^{\mathcal{C}}$: $\bigcup_{A \in \mathcal{C}} NP^{A}$

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

Definition (The polynomial hierarchy (PH))

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

Definition (The polynomial hierarchy (PH))

$$\Sigma_0^P \quad = \quad PTIME$$

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

Definition (The polynomial hierarchy (PH))

$$\Sigma_0^P = PTIME$$

$$\Sigma_{n+1}^P = NP^{\Sigma_n^P} \qquad n \geq 0$$

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

Definition (The polynomial hierarchy (PH))

$$\begin{aligned} \Sigma_0^P &= PTIME \\ \Sigma_{n+1}^P &= NP^{\Sigma_n^P} \qquad n \geq 0 \\ \Delta_{n+1}^P &= PTIME^{\Sigma_n^P} \qquad n \geq 0 \end{aligned}$$

The polynomial hierarchy

Using this generalisation we define the polynomial hierarchy.

Definition (The polynomial hierarchy (PH))

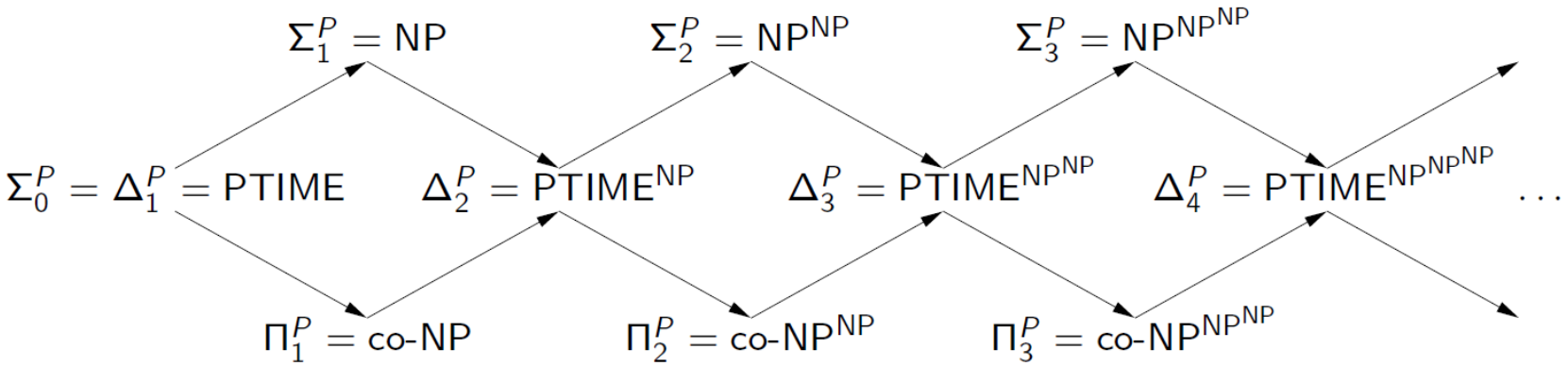
$$
\begin{aligned}
\Sigma_0^P &= PTIME \\
\Sigma_{n+1}^P &= NP^{\Sigma_n^P} && n \geq 0 \\
\Delta_{n+1}^P &= PTIME^{\Sigma_n^P} && n \geq 0 \\
\Pi_{n+1}^P &= co\text{-}\Sigma_{n+1}^P && n \geq 0
\end{aligned}
$$

The polynomial hierarchy

How can we depict polynomial hierarchy graphically?

The polynomial hierarchy

How can we depict polynomial hierarchy graphically?



The polynomial hierarchy and PSPACE

Definition

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

The polynomial hierarchy and PSPACE

Definition

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

What is the relationship between PH and PSPACE?

The polynomial hierarchy and PSPACE

Definition

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

What is the relationship between PH and PSPACE?

- PH $\subseteq$ PSPACE

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

- This would imply that PH has complete problems

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

- This would imply that PH has complete problems

What happens if PH has complete problems?

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

- This would imply that PH has complete problems

What happens if PH has complete problems?

- Each class Σ_k^P is closed under $\leq_{\mathrm{m}}^{\log}$ (or $\leq_P$)

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

- This would imply that PH has complete problems

What happens if PH has complete problems?

- Each class Σ_k^P is closed under $\leq_{\mathrm{m}}^{\log}$ (or $\leq_P$)
- If PH has a complete problem, then the polynomial hierarchy collapses to some finite level.

The polynomial hierarchy and PSPACE

Can it hold that PSPACE $\subseteq$ PH?

- This would imply that PH has complete problems

What happens if PH has complete problems?

- Each class Σ_k^P is closed under $\leq_{\mathrm{m}}^{\log}$ (or $\leq_P$)
- If PH has a complete problem, then the polynomial hierarchy collapses to some finite level.

Proposition

If the polynomial hierarchy does not collapse to a finite level, then it holds that PH $\subsetneq$ PSPACE.

A bit of intuition about PH

Exercise

An integer expression is defined recursively as:

- Every $a \in \mathbb{N}$ is an integer expression.
- If E and F are integer expressions, then $(E + F)$ and $(E \cup F)$ are also integer expressions.

The set of natural numbers represented by an integer expressions is defined recursively as:

- $L(a) = \{a\}$
- $L(E + F) = \{a + b \mid a \in L(E) \text{ y } b \in L(F)\}$
- $L(E \cup F) = L(E) \cup L(F)$

What is the complexity of the problem of determining if $L(E) = L(F)$?

Alternative characterisation of levels in PH

Notation

$\Pi_0^P = PTIME$

Proposition

For every $k \geq 1$: a language L over the alphabet Σ is in Σ_k^P if and only if there exists $A \in \Pi_{k-1}^P$ and a polynomial $p(n)$ such that for all $w \in \Sigma^*$:

$w \in L$ if and only if

there exists $z \in \Sigma^*$ such that $|z| \leq p(|w|)$ and $(w, z) \in A$.

Alternative characterisation of levels in PH

Notation

$\Pi_0^P = PTIME$

Proposition

For every $k \geq 1$: a language L over the alphabet Σ is in Σ_k^P if and only if there exists $A \in \Pi_{k-1}^P$ and a polynomial $p(n)$ such that for all $w \in \Sigma^*$:

$w \in L$ *if and only if*

there exists $z \in \Sigma^*$ such that $|z| \leq p(|w|)$ and $(w, z) \in A$.

Proof: Induction on k

Alternative characterisation of levels in PH

For $k = 1$: We need to prove that $L \in \text{NP}$ if and only if $A \in \text{PTIME}$ and a polynomial $p(n)$ such that:

$$w \in L$$
$$\Updownarrow$$
there exists z such that $|z| \leq p(|w|)$ and $(w, z) \in A$.

Alternative characterisation of levels in PH

For $k = 1$: We need to prove that $L \in \text{NP}$ if and only if $A \in \text{PTIME}$ and a polynomial $p(n)$ such that:

$$w \in L$$
$$\Updownarrow$$
$$\text{there exists } z \text{ such that } |z| \leq p(|w|) \text{ and } (w, z) \in A.$$

How do we prove this?

- The direction $\Leftarrow$ is easy (why?)
- For the other direction which A can we use? (Recall the alternative definition of NP)

Alternative characterisation of levels in PH

Assume that the statements holds for some k. We will prove that it also holds for $k + 1$.

($\Leftarrow$) Assume that there is a language $A \in \Pi_k^P$ and a polynomial $p(n)$ such that:

$$w \in L \text{ if and only if}$$
$$\text{there exists } z \text{ such that } |z| \leq p(|w|) \text{ and } (w, z) \in A.$$

Alternative characterisation of levels in PH

Assume that the statements holds for some k. We will prove that it also holds for $k+1$.

($\Leftarrow$) Assume that there is a language $A \in \Pi_k^P$ and a polynomial $p(n)$ such that:

$$w \in L \text{ if and only if}$$
$$\text{there exists } z \text{ such that } |z| \leq p(|w|) \text{ and } (w, z) \in A.$$

Therefore we have that $L \in \mathrm{NP}^A$

- So we conclude that: $L \in \Sigma_{k+1}^P$
- (This follows since $\mathrm{NP}^A \subseteq \mathrm{NP}^{\overline{A}}$)

Alternative characterisation of levels in PH

($\Rightarrow$) Assume now that $L \in \Sigma^P_{k+1}$.

It holds that $L = L(M^A)$, where

- $A \in \Sigma^P_k$
- M^A is polynomial time nondeterministic TM with an oracle for A

Since $A \in \Sigma^P_k$, by the induction hypothesis, there exists $C \in \Pi^P_{k-1}$ and a polynomial $p(n)$ such that:

$$w \in A \text{ if and only if}$$
$$\text{there exists } z \text{ such that } |z| \leq p(|w|) \text{ and } (w, z) \in C.$$

Alternative characterisation of levels in PH

Let Γ be the tape alphabet of M^A and assume that $\# \notin \Gamma$.

Define the language D as the set of all pairs (w, z) such that $w \in \Sigma^*$, $z = \alpha_0 \# \beta_0 \# \cdots \# \alpha_{m-1} \# \beta_{m-1} \# \alpha_m$, and:

(a) α_0 is the initial configuration of M^A with input w

(b) α_m an accepting configuration of M^A

(c) for $i \in [0, m-1]$: If the state of α_i is **not** $q_?$, then α_{i+1} is a configuration that follows from α_i in M^A and $\beta_i = \varepsilon$

Alternative characterisation of levels in PH

(d) for $i \in [0, m-1]$: If the state of α_i **is** $q_?$ and $u \in \Sigma^*$ is the word on the query tape of α_i, then α_{i+1} differs from α_i only in the state, and:

(d.1) the state of α_{i+1} is q_{NO}, $u \notin A$ and $\beta_i = \varepsilon$; or

(d.2) the state of α_{i+1} is q_{YES}, $|\beta_i| \leq p(|u|)$ and $(u, \beta_i) \in C$

Alternative characterisation of levels in PH

(d) for $i \in [0, m-1]$: If the state of α_i **is** $q_?$ and $u \in \Sigma^*$ is the word on the query tape of α_i, then α_{i+1} differs from α_i only in the state, and:

(d.1) the state of α_{i+1} is q_{NO}, $u \notin A$ and $\beta_i = \varepsilon$; or

(d.2) the state of α_{i+1} is q_{YES}, $|\beta_i| \leq p(|u|)$ and $(u, \beta_i) \in C$

Lemma

There is a polynomial $q(n)$ such that for each $w \in \Sigma^*$:

$w \in L$ if and only if

there is a $z \in \Sigma^*$ such that $|z| \leq q(|w|)$ and $(w, z) \in D$.

Alternative characterisation of levels in PH

It remains to prove that $D \in \Pi_k^P$.

Let M^C be a Turing machine with the oracle C which on input (w, z) works as follows:

(1) Verify in polynomial time if (w, z) does not satisfy (a), (b) or (c), and if this is the case M^C accepts (w, z)

(2) If (1) is not true, then M^C does the following for each configuration α_i with the state $q_?$:

(2.1) If the state of α_{i+1} is neither q_{NO} nor q_{YES}, if α_i and α_{i+1} differ in a symbol not corresponding to a state, or if the state of α_{i+1} is q_{NO} and $\beta_i \neq \varepsilon$, then M^C accepts (w, z)

Alternative characterisation of levels in PH

(2.2) If (2.1) does not hold, the state of α_{i+1} is q_{NO} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C guesses v such that $|v| \leq p(|u|)$, and then queries the oracle C with input (u, v). If the oracle replies with YES, then M^C accepts (w, z)

(2.3) If (2.1) and (2.2) do not hold, the state of α_{i+1} is q_{YES}, $u \in \Sigma^*$ is the word on the query tape of α_i and $|\beta_i| > p(|u|)$, then M^C accepts (w, z)

(2.4) If (2.1), (2.2) and (2.3) do not hold, the state of α_{i+1} is q_{YES} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C queries the oracle C with input (u, β_i). If the oracle replies NO, the M^C accepts (w, z)

Alternative characterisation of levels in PH

(2.2) If (2.1) does not hold, the state of α_{i+1} is q_{NO} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C guesses v such that $|v| \leq p(|u|)$, and then queries the oracle C with input (u, v). If the oracle replies with YES, then M^C accepts (w, z)

(2.3) If (2.1) and (2.2) do not hold, the state of α_{i+1} is q_{YES}, $u \in \Sigma^*$ is the word on the query tape of α_i and $|\beta_i| > p(|u|)$, then M^C accepts (w, z)

(2.4) If (2.1), (2.2) and (2.3) do not hold, the state of α_{i+1} is q_{YES} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C queries the oracle C with input (u, β_i). If the oracle replies NO, the M^C accepts (w, z)

It is easy to see that $\overline{D} = L(M^C)$.

Alternative characterisation of levels in PH

(2.2) If (2.1) does not hold, the state of α_{i+1} is q_{NO} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C guesses v such that $|v| \leq p(|u|)$, and then queries the oracle C with input (u, v). If the oracle replies with YES, then M^C accepts (w, z)

(2.3) If (2.1) and (2.2) do not hold, the state of α_{i+1} is q_{YES}, $u \in \Sigma^*$ is the word on the query tape of α_i and $|\beta_i| > p(|u|)$, then M^C accepts (w, z)

(2.4) If (2.1), (2.2) and (2.3) do not hold, the state of α_{i+1} is q_{YES} and $u \in \Sigma^*$ is the word on the query tape of α_i, then M^C queries the oracle C with input (u, β_i). If the oracle replies NO, the M^C accepts (w, z)

It is easy to see that $\overline{D} = L(M^C)$.

- How do we prove this?

Alternative characterisation of levels in PH

We conclude that $D \in \Pi_k^P$.

- This follows because M^C is a polynomial-time nondeterministic TM with an oracle $C \in \Pi_{k-1}^P$

Alternative characterisation of levels in PH

We conclude that $D \in \Pi_k^P$.

- This follows because M^C is a polynomial-time nondeterministic TM with an oracle $C \in \Pi_{k-1}^P$

To finish we just need to consider the alphabet of D.

- The input for D is of the form (w, z),where $z \in (\Gamma \cup \{\#\})^*$

Alternative characterisation of levels in PH

We conclude that $D \in \Pi_k^P$.

- This follows because M^C is a polynomial-time nondeterministic TM with an oracle $C \in \Pi_{k-1}^P$

To finish we just need to consider the alphabet of D.

- The input for D is of the form (w, z),where $z \in (\Gamma \cup \{\#\})^*$

But the use of the alphabet $\Gamma \cup \{\#\}$ is not essential for defining D.

- We can assume that $z \in \Sigma^*$. Why?

Alternative characterisation of Σ_k^P

Theorem

For any $k \geq 1$: a language L over the alphabet Σ is in Σ_k^P if and only if there exists $A \in PTIME$ and a polynomial $p(n)$ such that for all $w \in \Sigma^*$:

$$w \in L \text{ if and only if}$$
$$(\exists z_1 \in \Sigma^*,\ |z_1| \leq p(|w|))\ (\forall z_2 \in \Sigma^*,\ |z_2| \leq p(|w|))\ \cdots$$
$$(Q_k z_k \in \Sigma^*,\ |z_k| \leq p(|w|))\ (w, z_1, z_2, \ldots, z_k) \in A,$$

where $Q_k = \exists$ if k is odd and $Q_k = \forall$ if k is even.

Alternative characterisation of Σ_k^P

Theorem

For any $k \geq 1$: a language L over the alphabet Σ is in Σ_k^P if and only if there exists $A \in PTIME$ and a polynomial $p(n)$ such that for all $w \in \Sigma^*$:

$$w \in L \text{ if and only if}$$
$$(\exists z_1 \in \Sigma^*,\ |z_1| \leq p(|w|))\ (\forall z_2 \in \Sigma^*,\ |z_2| \leq p(|w|))\ \cdots$$
$$(Q_k z_k \in \Sigma^*,\ |z_k| \leq p(|w|))\ (w, z_1, z_2, \ldots, z_k) \in A,$$

where $Q_k = \exists$ if k is odd and $Q_k = \forall$ if k is even.

Exercise

Prove this (hint: how do we characterise Π_k^P like in the previous theorem).

Collapsing PH

The previous characterisation will allow us to construct complete problems for each Σ_k^P.

- All of these problems are extensions of SAT (and restrictions of TQBF)

Collapsing PH

The previous characterisation will allow us to construct complete problems for each Σ_k^P.

- All of these problems are extensions of SAT (and restrictions of TQBF)

But before we will show one fundamental result about PH

- We will talk about possible collapse of PH
- In the proof we will assume that complete problems for each Σ_k^P exist

The collapse of the Polynomial hierarchy

Theorem

For all $k \geq 1$*:*

(a) *If* $\Sigma_k^P = \Pi_k^P$, *then* $PH = \Sigma_k^P$

(b) *If* $\Sigma_k^P = \Delta_k^P$, *then* $PH = \Delta_k^P$

The collapse of the Polynomial hierarchy

Theorem

For all $k \geq 1$:

(a) *If $\Sigma_k^P = \Pi_k^P$, then $PH = \Sigma_k^P$*

(b) *If $\Sigma_k^P = \Delta_k^P$, then $PH = \Delta_k^P$*

Proof: We will need the following lemma:

Lemma

For all $k \geq 1$: $NP^{\Sigma_k^P \cap \Pi_k^P} = \Sigma_k^P$

The collapse of the Polynomial hierarchy

Proof of the lemma:

($\supseteq$) IF $L \in \Sigma_k^P$, then $L \in \text{NP}^A$, for some $A \in \Sigma_{k-1}^P$.

Since $\Sigma_{k-1}^P \subseteq \Delta_k^P \subseteq \Sigma_k^P \cap \Pi_k^P$.

- We can conclude that $L \in \text{NP}^{\Sigma_k^P \cap \Pi_k^P}$

($\subseteq$) Assume now that $L \in \text{NP}^{\Sigma_k^P \cap \Pi_k^P}$.

Then $L \in \text{NP}^A$, for some $A \in \Sigma_k^P \cap \Pi_k^P$.

The collapse of the Polynomial hierarchy

Since $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ and $\overline{A} = L(M_2^B)$, where:

- B is some complete problem for Σ_{k-1}^P
- M_1^B and M_2^B are polynomial-time nondeterministic TMs with an oracle for B

The collapse of the Polynomial hierarchy

Since $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ and $\overline{A} = L(M_2^B)$, where:

- B is some complete problem for Σ_{k-1}^P
- M_1^B and M_2^B are polynomial-time nondeterministic TMs with an oracle for B

Therefore: $L \in \mathrm{NP}^B$

The collapse of the Polynomial hierarchy

Since $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ and $\overline{A} = L(M_2^B)$, where:

- B is some complete problem for Σ_{k-1}^P
- M_1^B and M_2^B are polynomial-time nondeterministic TMs with an oracle for B

Therefore: $L \in \mathrm{NP}^B$

- Idea: Replace each call of M^A to the oracle A with a simultaneous call to M_1^B and M_2^B

The collapse of the Polynomial hierarchy

Since $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ and $\overline{A} = L(M_2^B)$, where:

- B is some complete problem for Σ_{k-1}^P
- M_1^B and M_2^B are polynomial-time nondeterministic TMs with an oracle for B

Therefore: $L \in \mathrm{NP}^B$

- Idea: Replace each call of M^A to the oracle A with a simultaneous call to M_1^B and M_2^B

We conclude that $L \in \Sigma_k^P$. □

The collapse of the Polynomial hierarchy

Now we can complete the proof of the theorem.

(a) Assume that $\Sigma_k^P = \Pi_k^P$.

We will prove by induction on $j \geq k$ that $\Sigma_j^P = \Pi_j^P = \Sigma_k^P$.

For $j = k$ the property holds by the hypothesis.

Assume that the property holds for some $j \geq k$, we next prove that the property also holds for $j + 1$.

The collapse of the Polynomial hierarchy

We have that:

$$\begin{aligned}
\Sigma^P_{j+1} &= \mathrm{NP}^{\Sigma^P_j} \\
&= \mathrm{NP}^{\Sigma^P_j \cap \Pi^P_j} && \text{since } \Sigma^P_j = \Pi^P_j \\
&= \Sigma^P_j && \text{by the lemma } (j \geq k \geq 1) \\
&= \Sigma^P_k && \text{by the induction hypothesis}
\end{aligned}$$

Furthermore, we have:

$$\begin{aligned}
\Pi^P_{j+1} &= \text{co-}\Sigma^P_{j+1} \\
&= \text{co-}\Sigma^P_k && \text{by the above proof} \\
&= \Sigma^P_k && \text{by the hypothesis}
\end{aligned}$$

The collapse of the Polynomial hierarchy

(b) Assume that $\Delta_k^P = \Sigma_k^P$.

Since Δ_k^P is closed under complement, we have that $\Sigma_k^P = \Pi_k^P$.

- From (a) it follows that $\text{PH} = \Sigma_k^P$

We conclude that $\text{PH} = \Delta_k^P$. □

The collapse of the Polynomial hierarchy

Corollary

- *If $PTIME = NP$, then $PH = PTIME$*
- *If $NP =$ co-NP, then $PH = NP$*

Complete problems for levels of PH

The language QBF_i ($i \geq 1$) is defined as the set of all true quantified boolean formulas that are of the form:

$$\begin{aligned} &\exists x_{1,1} \cdots \exists x_{1,m_1} \\ &\forall x_{2,1} \cdots \forall x_{2,m_2} \\ &\exists x_{3,1} \cdots \exists x_{3,m_3} \\ &\quad \cdots \\ &Q_i x_{i,1} \cdots Q_i x_{i,m_i}\ \varphi \end{aligned}$$

where:

- $Q_i = \exists$ if i is odd, and $Q_i = \forall$ if i is even
- φ is a propositional formula using variables $x_{1,1}, \ldots, x_{1,m_1}, \ldots, x_{i,1}, \ldots, x_{i,m_i}$

A complete problem for Σ_k^P

The sequence of problems $\{\text{QBF}_i\}_{i \geq 1}$ is enough to represent the polynomial hierarchy.

Theorem

For every $k \geq 1$, QBF_k is Σ_k^P-complete.

A complete problem for Σ_k^P

The sequence of problems $\{\text{QBF}_i\}_{i \geq 1}$ is enough to represent the polynomial hierarchy.

Theorem

For every $k \geq 1$, QBF_k is Σ_k^P-complete.

Proof: First we have to show that $\text{QBF}_k \in \Sigma_k^P$.

- How is this done?

A complete problem for Σ_k^P

Next, we need to prove that QBF_k is Σ_k^P-hard.

- Take any $L \in \Sigma_k^P$. We have to show that L is reducible to QBF_k (in LOGSPACE)

A complete problem for Σ_k^P

Next, we need to prove that QBF_k is Σ_k^P-hard.

- Take any $L \in \Sigma_k^P$. We have to show that L is reducible to QBF_k (in LOGSPACE)

To simplify the proof we will assume that:

A complete problem for Σ_k^P

Next, we need to prove that QBF_k is Σ_k^P-hard.

- Take any $L \in \Sigma_k^P$. We have to show that L is reducible to QBF_k (in LOGSPACE)

To simplify the proof we will assume that:

- L is defined over the alphabet $\{0, 1\}$

A complete problem for Σ_k^P

Next, we need to prove that QBF_k is Σ_k^P-hard.

- Take any $L \in \Sigma_k^P$. We have to show that L is reducible to QBF_k (in LOGSPACE)

To simplify the proof we will assume that:

- L is defined over the alphabet $\{0, 1\}$
- k is odd

A complete problem for Σ_k^P

Next, we need to prove that QBF_k is Σ_k^P-hard.

- Take any $L \in \Sigma_k^P$. We have to show that L is reducible to QBF_k (in LOGSPACE)

To simplify the proof we will assume that:

- L is defined over the alphabet $\{0, 1\}$
- k is odd

The proof is similar for an arbitrary alphabet and k even.

A complete problem for Σ_k^P

We need to prove that there is a function $f : \{0,1\}^* \to \{0,1\}^*$ such that

- f is computable in LOGSPACE and;
- for every $w \in \{0,1\}^*$: $w \in L$ if and only if $f(w) \in \text{QBF}_k$

Notación

$f(w) = \varphi_w$

A complete problem for Σ_k^P

Since $L \in \Sigma_k^P$, there exists $A \in \text{PTIME}$ and a polynomial $p(n)$ such that for every $w \in \{0,1\}^*$:

$w \in L$ if and only if

$$(\exists z_1 \in \{0,1\}^*, |z_1| = p(|w|))$$
$$(\forall z_2 \in \{0,1\}^*, |z_2| = p(|w|))$$
$$\dots$$
$$(\exists z_k \in \{0,1\}^*, |z_k| = p(|w|))\ w\#z_1\#z_2\#\dots\#z_k \in A$$

This follows from the characterisation of Σ_k^P.

- How do we achieve $|z_1| = |z_2| = \cdots = |z_k| = p(|w|)$?

A complete problem for Σ_k^P

Since $A \in \text{PTIME}$, there exists a deterministic Turing machine $M = (Q = \{q_0, \ldots, q_m\}, \Sigma = \{0, 1, \#\}, \Gamma = \{0, 1, \#, \text{B}, \vdash\}, q_0, \delta, q_{accept}, q_{reject})$ where:

- M uses a single tape
- M halts on every input
- $L = L(M)$
- $t_M(n)$ is $O(n^c)$, for some natural number c
- M uses $\vdash$ to mark the left end of the tape (no transition can move left if it is reading this symbol and the symbol can not be rewritten)

A complete problem for Σ_k^P

Without the loss of generality we assume that:

- $w = a_1 \cdots a_n$, with each $a_i \in \{0, 1\}$
- For each $a \in \Gamma$, the function $\delta(q_{accept}, a)$ is not defined
- For each $a \in \Gamma$, the function $\delta(q_{reject}, a)$ is not defined

A complete problem for Σ_k^P

Let $\ell = n + k \cdot (1 + p(n))$

- ℓ is the length of the input $w\#z_1\#z_2\#\cdots\#z_k$ of the TM M

A complete problem for Σ_k^P

Let $\ell = n + k \cdot (1 + p(n))$

- ℓ is the length of the input $w\#z_1\#z_2\#\cdots\#z_k$ of the TM M

To define φ_w we use the following propositional variables:

$$
\begin{array}{lcl}
z_{i,j} & : & i \in [1,k] \text{ and } j \in [1, p(n)] \\
s_{t,p,a} & : & t \in [0, t_M(\ell)],\ p \in [0, t_M(\ell)+1] \text{ and } a \in \{0,1,\#,\mathtt{B},\vdash\} \\
c_{t,p} & : & t \in [0, t_M(\ell)] \text{ and } p \in [0, t_M(\ell)+1] \\
e_{t,q} & : & t \in [0, t_M(\ell)] \text{ and } q \in Q
\end{array}
$$

A complete problem for Σ_k^P

The formula φ_w is defined as:

$$\exists z_{1,1} \cdots \exists z_{1,p(n)}$$

$$\forall z_{2,1} \cdots \forall z_{2,p(n)}$$

$$\exists z_{3,1} \cdots \exists z_{3,p(n)}$$

$$\cdots$$

$$\exists z_{k,1} \cdots \exists z_{k,p(n)}$$

$$\exists s_{0,0,0}\, \exists s_{0,0,1}\, \exists s_{0,0,\#}\, \exists s_{0,0,\mathtt{B}}\, \exists s_{0,0,\vdash} \cdots \exists s_{t_M(\ell),t_M(\ell)+1,0}$$

$$\exists s_{t_M(\ell),t_M(\ell)+1,1}\, \exists s_{t_M(\ell),t_M(\ell)+1,\#}\, \exists s_{t_M(\ell),t_M(\ell)+1,\mathtt{B}}\, \exists s_{t_M(\ell),t_M(\ell)+1,\vdash}$$

$$\exists c_{0,0} \cdots \exists c_{t_M(\ell),t_M(\ell)+1}\, \exists e_{0,q_0} \cdots \exists e_{0,q_m}\, \exists e_{t_M(\ell),q_0} \cdots \exists e_{t_M(\ell),q_m}$$

$$\Big(\varphi_I \wedge \varphi_C \wedge \varphi_\delta \wedge \varphi_A \Big)$$

A complete problem for Σ_k^P

φ_I: the initial state

$$c_{0,1} \wedge e_{0,q_0} \wedge s_{0,0,\vdash} \wedge \left(\bigwedge_{p=1}^{n} s_{0,p,a_p} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} s_{0,(n+1)+(i-1)\cdot(p(n)+1),\#} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{p(n)} \neg z_{i,j} \rightarrow s_{0,(n+1)+(i-1)\cdot(p(n)+1)+j,0} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{p(n)} z_{i,j} \rightarrow s_{0,(n+1)+(i-1)\cdot(p(n)+1)+j,1} \right) \wedge$$

$$\left(\bigwedge_{p=(n+1)+k\cdot(p(n)+1)}^{t_M(\ell)+1} s_{0,p,\mathtt{B}} \right)$$

A complete problem for Σ_k^P

φ_C: The machine works correctly

φ_C is a conjunction of four formulas. First, each cell can contain only one symbol (recall Cook-Levin and tableaus):

$$\bigwedge_{t=0}^{t_M(\ell)} \bigwedge_{p=0}^{t_M(\ell)+1} \left(\bigvee_{a \in \Gamma} \left(s_{t,p,a} \wedge \bigwedge_{b \in (\Gamma \setminus \{a\})} \neg s_{t,p,b} \right) \right)$$

A complete problem for Σ_k^P

Second, the machine is always in a unique state:

$$\bigwedge_{t=0}^{t_M(\ell)} \left(\bigvee_{q \in Q} \left(e_{t,q} \wedge \bigwedge_{q' \in (Q \setminus \{q\})} \neg e_{t,q'} \right) \right)$$

Third, the head of the machine is always in a unique position:

$$\bigwedge_{t=0}^{t_M(\ell)} \left(\bigvee_{p=0}^{t_M(\ell)+1} \left(c_{t,p} \wedge \bigwedge_{p' \in ([0, t_M(\ell)+1] \setminus \{p\})} \neg c_{t,p'} \right) \right)$$

A complete problem for Σ_k^P

Fourth, the value of a cell on the work tape of the machine is not changed is it was not under the head of the machine (in the previous time point)

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)+1} \left((\neg c_{t,p}) \rightarrow \bigwedge_{a \in \Gamma} (s_{t+1,p,a} \leftrightarrow s_{t,p,a}) \right)$$

A complete problem for Σ_k^P

φ_δ: the relation δ defines how the machine moves from one configuration to another (we represent $\leftarrow$ as -1, $\square$ as 0 and $\rightarrow$ as 1)

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)} \left[\bigwedge_{\delta(q,a)=(q',b,X)} \left((e_{t,q} \wedge c_{t,p} \wedge s_{t,p,a}) \rightarrow (e_{t+1,q'} \wedge c_{t+1,p+X} \wedge s_{t+1,p,b}) \right) \right] \wedge$$

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)} \left[\bigwedge_{\delta(q,a) \text{ is not defined}} \left((e_{t,q} \wedge c_{t,p} \wedge s_{t,p,a}) \rightarrow (e_{t+1,q} \wedge c_{t+1,p} \wedge s_{t+1,p,a}) \right) \right]$$

A complete problem for Σ_k^P

φ_A: The machine accepts w

$$e_{t_M(\ell), q_{accept}}$$

A complete problem for Σ_k^P

φ_A: The machine accepts w

$$e_{t_M(\ell), q_{accept}}$$

To finish the proof we need:

A complete problem for Σ_k^P

φ_A: The machine accepts w

$$e_{t_M(\ell),q_{accept}}$$

To finish the proof we need:

- $w \in L$ if and only if φ_w is true

A complete problem for Σ_k^P

φ_A: The machine accepts w

$$e_{t_M(\ell), q_{accept}}$$

To finish the proof we need:

- $w \in L$ if and only if φ_w is true
- φ_w can be constructed in logarithmic space

A complete problem for Σ_k^P

φ_A: The machine accepts w

$$e_{t_M(\ell), q_{accept}}$$

To finish the proof we need:

- $w \in L$ if and only if φ_w is true
- φ_w can be constructed in logarithmic space

Why does this hold? □