The Polynomial Hierarchy

Antonis Antonopoulos

Computation and Reasoning Laboratory
National Technical University of Athens

December 2009 - January 2010

Outline

Outline

Introduction

TSP Versions

1. *TSP (D)*
2. *EXACT TSP*
3. *TSP COST*
4. *TSP*

$(1) \leq_P (2) \leq_P (3) \leq_P (4)$

DP Class Definition

Definition

A language L is in the class **DP** if and only if there are two languages $L_1 \in \mathbf{NP}$ and $L_2 \in co\mathbf{NP}$ such that $L = L_1 \cap L_2$.

- **DP** is *not* $\mathbf{NP} \cap co\mathbf{NP}$!
- Also, **DP** is a *syntactic* class, and so it has complete problems.

***SAT-UNSAT* Definition**

Given two Boolean expressions ϕ, ϕ', both in 3CNF. Is it true that ϕ is satisfiable and ϕ' is not?

Complete Problems for DP

Theorem

SAT-UNSAT is **DP**-complete.

Proof

- Firstly, we have to show it is in **DP**.
  So, let:
  $L_1 = \{(\phi, \phi'): \ \phi \text{ is satisfiable}\}$.
  $L_2 = \{(\phi, \phi'): \ \phi' \text{ is unsatisfiable}\}$.
  It is easy to see, $L_1 \in \mathbf{NP}$ and $L_2 \in co\mathbf{NP}$, thus
  $L \equiv L_1 \cap L_2 \in \mathbf{DP}$.
- For completeness, let $L \in \mathbf{DP}$. We have to show that $L \leq_P SAT\text{-}UNSAT$. $L \in \mathbf{DP} \Rightarrow L = L_1 \cap L_2$, $L_1 \in \mathbf{NP}$ and $L_2 \in co\mathbf{NP}$.
  SAT **NP**-complete$\Rightarrow \exists R_1{:}L_1 \leq_P SAT$ and $R_2{:}\overline{L_2} \leq_P SAT$.
  Hence, $L \leq_P SAT\text{-}UNSAT$,by $R(x) = (R_1(x), R_2(x))$

Complete Problems for DP

Theorem

EXACT TSP is **DP**-complete.

Proof

- *EXACT TSP* $\in$ **DP**, by $L_1 \equiv$*TSP* $\in$ **NP** and $L_2 \equiv$*TSP COMPLEMENT* $\in$ *co***NP**
- Completeness: we'll show that *SAT-UNSAT*$\leq_P$*EXACT TSP*.
  3SAT$\leq_P$*HP*: $(\phi, \phi') \rightarrow (G, G')$
  Broken Hamilton Path (*2 node-disjoint paths that cover all nodes*)
  Almost Satisfying Truth Assignement (*satisfies all clauses except for one*)

Complete Problems for DP

Proof

We define distances:

1. If $(i,j) \in$ E(G) or E(G'): $d(i,j) \equiv 1$
2. If $(i,j) \notin$ E(G), but i and j $\in$ V(G): $d(i,j) \equiv 2$
3. Otherwise: $d(i,j) \equiv 4$

Let n be the size of the graph.

1. If ϕ and ϕ' satisfiable, then $optCost = n$
2. If ϕ and ϕ' **un**satisfiable, then $optCost = n + 3$
3. If ϕ satisfiable and ϕ' not, then $optCost = n + 2$
4. If ϕ' satisfiable and ϕ not, then $optCost = n + 1$

"**yes**" instance of *SAT-UNSAT* $\Leftrightarrow optCost = n + 2$

Let $B \equiv n + 2$!

Other DP-complete problems

Also:

- *CRITICAL SAT*: Given a Boolean expression ϕ, is it true that it's **un**satisfiable, but deleting any clause makes it satisfiable?
- *CRITICAL HAMILTON PATH*: Given a graph, is it true that it has **no** Hamilton path, but addition of any edge creates a Hamilton path?
- *CRITICAL 3-COLORABILITY*: Given a graph, is it true that it is **not** 3-colorable, but deletion of any node makes it 3-colorable?

are **DP**-complete!

Oracle TMs and Oracle Classes

Definition

A Turing Machine $M^?$ with *oracle* is a multi-string deterministic TM that has a special string, called **query string**, and three special states: $q_?$ (*query state*), and q_{YES}, q_{NO} (*answer states*). Let $A \subseteq \Sigma^*$ be an arbitrary language. The computation of oracle machine M^A proceeds like an ordinary TM except for transitions from the query state:
From the $q_?$ moves to either q_{YES}, q_{NO}, depending on whether the current query string is in A or not.

- The answer states allow the machine to use this answer to its further computation.
- The computation of $M^?$ with oracle A on iput x is denoted as $M^A(x)$.

Oracle TMs and Oracle Classes

Definition

Let $\mathcal{C}$ be a time complexity class (deterministic or nondeterministic).
Define $\mathcal{C}^A$ to be the class of all languages decided by machines of the same sort and time bound as in $\mathcal{C}$, only that the machines have now oracle A.

Theorem

There exists an oracle A for which $\mathbf{P}^A = \mathbf{NP}^A$

Proof

Take A to be a **PSPACE**-complete language. Then:
$\mathbf{PSPACE} \subseteq \mathbf{P}^A \subseteq \mathbf{NP}^A \subseteq \mathbf{NPSPACE} \subseteq \mathbf{PSPACE}$.

Theorem

There exists an oracle B for which $\mathbf{P}^B \neq \mathbf{NP}^B$

The Classes P^{NP} and FP^{NP}

Alternative DP Definition

DP is the class of languages that can be decided by an oracle machine which makes 2 queries to a *SAT* oracle, and accepts iff the 1st answer is **yes**, and the 2nd is **no**.

- $\mathbf{P}^{SAT}$ is the class of languages decided in pol time with a *SAT* oracle.
  - Polynomial number of queries
  - Queries computed adaptively
- *SAT* **NP**-complete $\Rightarrow \mathbf{P}^{SAT} = \mathbf{P}^{\mathbf{NP}}$
- $\mathbf{FP}^{\mathbf{NP}}$ is the class of functions that can be computed by a pol-time TM with a *SAT* oracle.
- Goal: $MAX\ OUTPUT \leq_P MAX\text{-}WEIGHT\ SAT \leq_P SAT$

FP^{NP}-complete Problems

MAX OUTPUT Definition

Given NTM N, with input 1^n, which halts after $\mathcal{O}(n)$,with output a string of length n. Which is the largest output,of any computation of N on 1^n?

Theorem

MAX OUTPUT is $\mathbf{FP^{NP}}$-complete.

Proof

MAX OUTPUT $\in \mathbf{FP^{NP}}$.

Let $F : \Sigma^* \to \Sigma^* \in \mathbf{FP^{NP}} \Rightarrow \exists$ pol-time TM $M^?$, s.t. $M^{SAT}(x) = F(x)$. We'll show: $F \leq$*MAX OUTPUT*!

Reductions R and S (*log space computable*) s.t.:

- $\forall x, R(x)$ is a instance of *MAX OUTPUT*
- $S(\text{max output of } R(x)) \to F(x)$

FP^{NP}-complete Problems

Proof (cont.)

NTM N:

Let $n = p^2(|x|)$, $p(\cdot)$, is the pol bound of SAT.

$N(1^n)$ generates x on a string.

M^{SAT} query state (ϕ_1):

- If $z_1 = 0$ (ϕ_1 unsat), then continue from q_{NO}.
- If $z_1 = 1$ (ϕ_1 sat), then guess assignment T_1:
    - If test succeeds, continue from q_{YES}.
    - If test fails, output$=0^n$ and **halt**. (Unsuccessful computation)

Continue to all guesses (z_i), and **halt**, with output$= \underbrace{z_1 z_2 \ldots .00}_{n}$

(Successful computation)

FP^{NP}-complete Problems

Proof (cont.)

We claim that the successful computation that outputs the largest integer, correspond to a correct simulation:
Let j the smallest integer,s.t.: $z_j = 0$, while ϕ_j was satisfiable.
Then, $\exists$ another successful computation of N, s.t.: $z_j = 1$.
The computations agree to the first $j-1$ digits,$\Rightarrow$ the 2^{nd} represents a larger number.
The S part: $F(x)$ can be read off the end of the largest output of N.

FP^{NP}-complete Problems

MAX-WEIGHT SAT Definition

Given a set of clauses, each with an integer weight, find the truth assignment that satisfies a set of clauses with the most total weight.

FP^{NP}-complete Problems

MAX-WEIGHT SAT Definition

Given a set of clauses, each with an integer weight, find the truth assignment that satisfies a set of clauses with the most total weight.

Theorem

MAX-WEIGHT SAT is $\mathbf{FP^{NP}}$-complete.

Proof

MAX-WEIGHT SAT is in $\mathbf{FP^{NP}}$: By binary search, and a *SAT* oracle, we can find the largest possible total weight of satisfied clauses, and then, by setting the variables 1-1, the truth assignment that achieves it.

$MAX\ OUTPUT \leq MAX\text{-}WEIGHT\ SAT$:

FP^{NP}-complete Problems

Proof (cont.)

- $NTMN(1^n) \rightarrow \phi(N,m)$:
  Any satisfying truth assignment of $\phi(N,m) \rightarrow$ legal comp. of $N(1^n)$
- Clauses are given a huge weight (2^n), so that any t.a. that aspires to be optimum satisfy all clauses of $\phi(N,m)$.
- Add more clauses: (y_i): $i = 1, ..n$ with weight 2^{n-i}.
- Now, optimum t.a. must *not* represent any legal computation, but this which produces the *largest* possible output value.
- S part: From optimum t.a. of the resulting expression (or the weight), we can recover the optimum output of $N(1^n)$.

FP^{NP}-complete Problems

And the main result:

Theorem

TSP is $\mathbf{FP}^{\mathbf{NP}}$-complete.

FP^{NP}-complete Problems

And the main result:

Theorem

TSP is $\mathbf{FP}^{\mathbf{NP}}$-complete.

Corollary

TSP COST is $\mathbf{FP}^{\mathbf{NP}}$-complete.

FP^{NP}-complete Problems

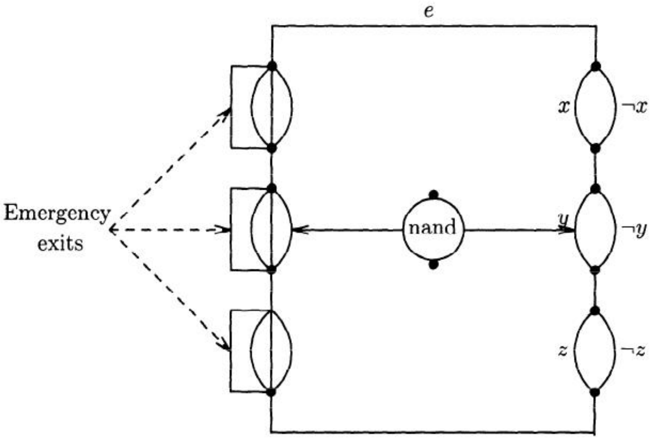


Figure: The overall construction (17-2)

The Class $P^{NP[\log n]}$

Definition

$\mathbf{P}^{\mathbf{NP[logn]}}$ is the class of all languages decided by a polynomial time oracle machine, which on input x asks a total of $\mathcal{O}(\log |x|)$ SAT queries.

- $\mathbf{FP}^{\mathbf{NP[logn]}}$ is the corresponding class of functions.

The Class $P^{NP[\log n]}$

Definition

$\mathbf{P}^{\mathbf{NP[logn]}}$ is the class of all languages decided by a polynomial time oracle machine, which on input x asks a total of $\mathcal{O}(\log |x|)$ *SAT* queries.

- $\mathbf{FP}^{\mathbf{NP[logn]}}$ is the corresponding class of functions.

CLIQUE SIZE Definition

Given a graph, determine the size of his *largest* clique.

The Class $P^{NP[\log n]}$

Definition

$\mathbf{P^{NP[logn]}}$ is the class of all languages decided by a polynomial time oracle machine, which on input x asks a total of $\mathcal{O}(\log|x|)$ *SAT* queries.

- $\mathbf{FP^{NP[logn]}}$ is the corresponding class of functions.

CLIQUE SIZE Definition

Given a graph, determine the size of his *largest* clique.

Theorem

CLIQUE SIZE is $\mathbf{FP^{NP[logn]}}$-complete.

Conclusion

1. *TSP (D)* is **NP**-complete.
2. *EXACT TSP* is **DP**-complete.
3. *TSP COST* is $\mathbf{FP}^{\mathbf{NP}}$-complete.
4. *TSP* is $\mathbf{FP}^{\mathbf{NP}}$-complete.

And now,

- $\mathbf{P}^{\mathbf{NP}} \rightarrow \mathbf{NP}^{\mathbf{NP}}$?
- Oracles for $\mathbf{NP}^{\mathbf{NP}}$?

Outline

The Polynomial Hierarchy

Polynomial Hierarchy Definition

- $\Delta_0\mathbf{P} = \Sigma_0\mathbf{P} = \Pi_0\mathbf{P} = \mathbf{P}$
- $\Delta_{i+1}\mathbf{P} = \mathbf{P}^{\Sigma_i\mathbf{P}}$
- $\Sigma_{i+1}\mathbf{P} = \mathbf{NP}^{\Sigma_i\mathbf{P}}$
- $\Pi_{i+1}\mathbf{P} = co\mathbf{NP}^{\Sigma_i\mathbf{P}}$
- $$\mathbf{PH} \equiv \bigcup_{i \geqslant 0} \Sigma_i\mathbf{P}$$

- $\Sigma_0\mathbf{P} = \mathbf{P}$
- $\Delta_1\mathbf{P} = \mathbf{P}$, $\Sigma_1\mathbf{P} = \mathbf{NP}$, $\Pi_1\mathbf{P} = co\mathbf{NP}$
- $\Delta_2\mathbf{P} = \mathbf{P}^{\mathbf{NP}}$, $\Sigma_2\mathbf{P} = \mathbf{NP}^{\mathbf{NP}}$, $\Pi_2\mathbf{P} = co\mathbf{NP}^{\mathbf{NP}}$

Basic Theorems

Theorem

Let L be a language , and $i \geq 1$. $L \in \Sigma_i\mathbf{P}$ iff there is a polynomially balanced relation R such that the language $\{x; y : (x, y) \in R\}$ is in $\Pi_{i-1}\mathbf{P}$ and

$$L = \{x : \exists y, s.t. : (x, y) \in R\}$$

Proof (by Induction)

- For $i = 1$
  $\{x; y : (x, y) \in R\} \in \mathbf{P}$,so $L = \{x | \exists y : (x, y) \in R\} \in \mathbf{NP}$
- For $i > 1$
  If $\exists R \in \Pi_{i-1}\mathbf{P}$, we must show that $L \in \Sigma_i\mathbf{P} \Rightarrow$
  $\exists$ NTM with $\Sigma_{i-1}\mathbf{P}$ oracle: NTM(x) guesses a y and asks $\Pi_{i-1}\mathbf{P}$ oracle whether $(x, y) \notin R$.

Basic Theorems

Proof (cont.)

- If $L \in \Sigma_i\mathbf{P}$, we must show the existence or R.
  $L \in \Sigma_i\mathbf{P} \Rightarrow \exists$ NTM M^K, $K \in \Sigma_{i-1}\mathbf{P}$, which decides L.
  $K \in \Sigma_{i-1}\mathbf{P} \Rightarrow \exists S \in \Pi_{i-2}\mathbf{P} : (z \in K \Leftrightarrow \exists w : (z, w) \in S)$
  We must describe a relation R (we know: $x \in L \Leftrightarrow$ accepting comp of $M^K(x)$)
  Query Steps: "yes" $\rightarrow z_i$ has a certificate w_i st $(z_i, w_i) \in S$.
  So, $R(x) =$"$(x, y) \in R$ *iff y records an accepting computation of* $M^?$ *on* x *, together with a certificate* w_i *for each* **yes** *query* z_i *in the computation.*"
  We must show $\{x; y : (x, y) \in R\} \in \Pi_{i-1}\mathbf{P}$.

Basic Theorems

Corollary

Let L be a language , and $i \geq 1$. $L \in \Pi_i\mathbf{P}$ iff there is a polynomially balanced relation R such that the language $\{x; y : (x,y) \in R\}$ is in $\Sigma_{i-1}\mathbf{P}$ and

$$L = \{x : \forall y, |y| \leq |x|^k, s.t. : (x,y) \in R\}$$

Corollary

Let L be a language , and $i \geq 1$. $L \in \Sigma_i\mathbf{P}$ iff there is a polynomially balanced, polynomially-time decicable $(i+1)$-ary relation R such that:

$$L = \{x : \exists y_1 \forall y_2 \exists y_3 ... Q y_i, s.t. : (x, y_1, ..., y_i) \in R\}$$

where the i^{th} quantifier Q is $\forall$, if i is even, and $\exists$, if i is odd.

Basic Theorems

Theorem

If for some $i \geq 1$, $\Sigma_i\mathbf{P} = \Pi_i\mathbf{P}$, then for all $j > i$:

$$\Sigma_j\mathbf{P} = \Pi_j\mathbf{P} = \Delta_j\mathbf{P} = \Sigma_i\mathbf{P}$$

Or, the polynomial hierarchy *collapses* to the i^{th} level.

Proof

It suffices to show that: $\Sigma_i\mathbf{P} = \Pi_i\mathbf{P} \Rightarrow \Sigma_{i+1}\mathbf{P} = \Sigma_i\mathbf{P}$
Let $L \in \Sigma_{i+1}\mathbf{P} \Rightarrow \exists R \in \Pi_i\mathbf{P}$: $L = \{x | \exists y : (x, y) \in R\}$
Since $\Pi_i\mathbf{P} = \Sigma_i\mathbf{P} \Rightarrow R \in \Sigma_i\mathbf{P}$
$(x, y) \in R \Leftrightarrow \exists z : (x, y, z) \in S$, $S \in \Pi_{i-1}\mathbf{P}$.
Thus, $x \in L \Leftrightarrow \exists y; z : (x, y, z) \in S$, $S \in \Pi_{i-1}\mathbf{P}$, which means $L \in \Sigma_i\mathbf{P}$.

Basic Theorems

Corollary

If **P**=**NP**, or even **NP**=co**NP**, the Polynomial Hierarchy collapses to the first level.

Basic Theorems

Corollary

If **P**=**NP**, or even **NP**=co**NP**, the Polynomial Hierarchy collapses to the first level.

MINIMUM CIRCUIT Definition

Given a Boolean Circuit C, is it true that there is no circuit with fewer gates that computes the same Boolean function

- *MINIMUM CIRCUIT* is in $\Pi_2\mathbf{P}$, and not known to be in any class below that.
- It is open whether *MINIMUM CIRCUIT* is $\Pi_2\mathbf{P}$-complete.

Theorem

If *SAT* has Polynomial Circuits, then the Polynomial Hierarchy collapses to the second level.

Basic Theorems

$QSAT_i$ Definition

Given expression ϕ, with Boolean variables partitioned into i sets X_i,is ϕ satisfied by the overall truth assignment of the expression:

$$\exists X_1 \forall X_2 \exists X_3 \ldots\ldots Q X_i \phi$$

, where Q is $\exists$ if i is *odd*, and $\forall$ if i is even.

Theorem

For all $i \geq 1$ $QSAT_i$ is $\Sigma_i\mathbf{P}$-complete.

Basic Theorems

Theorem

If there is a **PH**-complete problem, then the polynomial hierarchy collapses to some finite level.

Proof

Let L is **PH**-complete.
Since $L \in \mathbf{PH}$, $\exists i \geq 0 : L \in \Sigma_i \mathbf{P}$.
But any $L' \in \Sigma_{i+1}\mathbf{P}$ reduces to L. Since PH is closed under reductions, we imply that $L' \in \Sigma_i \mathbf{P}$, so $\Sigma_i \mathbf{P} = \Sigma_{i+1}\mathbf{P}$.

Basic Theorems

Theorem

If there is a **PH**-complete problem, then the polynomial hierarchy collapses to some finite level.

Proof

Let L is **PH**-complete.
Since $L \in \mathbf{PH}$, $\exists i \geq 0 : L \in \Sigma_i\mathbf{P}$.
But any $L' \in \Sigma_{i+1}\mathbf{P}$ reduces to L. Since PH is closed under reductions, we imply that $L' \in \Sigma_i\mathbf{P}$, so $\Sigma_i\mathbf{P} = \Sigma_{i+1}\mathbf{P}$.

Theorem

$\mathbf{PH} \subseteq \mathbf{PSPACE}$

- $\mathbf{PH} \stackrel{?}{=} \mathbf{PSPACE}$ (Open). If it was, then **PH** has complete problems, so it collapses to some finite level.

BPP and PH

Theorem

$\mathbf{BPP} \subseteq \Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P}$

Proof

Because $co\mathbf{BPP} = \mathbf{BPP}$,we prove only $\mathbf{BPP} \subseteq \Sigma_2\mathbf{P}$.
Let $L \in \mathbf{BPP}$ (L is accepted by "clear majority").
For $|x| = n$, let $A(x) \subseteq \{0,1\}^{p(n)}$ be the set of *accepting* computations.
We have:

- $x \in L \Rightarrow |A(x)| \geq 2^{p(n)} \left(1 - \frac{1}{2^n}\right)$
- $x \notin L \Rightarrow |A(x)| \leq 2^{p(n)} \left(\frac{1}{2^n}\right)$

Let U be the set of all bit strings of length $p(n)$.
For $a, b \in U$, let $a \oplus b$ be the XOR:
$a \oplus b = c \Leftrightarrow c \oplus b = a$, so "$\oplus b$" is 1-1.

BPP and PH

Proof (cont.)
For $t \in U$, $A(x) \oplus t = \{a \oplus t : a \in A(x)\}$ (translation of $A(x)$ by t). We imply that: $|A(x) \oplus t| = |A(x)|$
If $x \in L$, consider a *random* (drawing $p^2(n)$ bits) sequence of translations: $t_1, t_2, .., t_{p(n)} \in U$.
For $b \in U$, these translations *cover* b, if $b \in A(x) \oplus t_j$, $j \leq p(n)$.
$b \in A(x) \oplus t_j \Leftrightarrow b \oplus t_j \in A(x) \Rightarrow \mathbf{Pr}[b \notin A(x) \oplus t_j] = \frac{1}{2^n}$
$\mathbf{Pr}$[b is **not** covered by any t_j]$=2^{-np(n)}$
$\mathbf{Pr}[\exists$ point that is **not** covered$] \leq 2^{-np(n)}|U| = 2^{-(n-1)p(n)}$

BPP and PH

Proof (cont.)

So, $T = (t_1, .., t_{p(n)})$ has a positive probability that it covers all of U.

If $x \notin L, |A(x)|$ is exp small,and (for large n) there's not T that cover all U.

$(x \in L) \Leftrightarrow (\exists T$ that cover all $U)$

So,

$$L = \{x | \exists (T \in \{0,1\}^{p^2(n)}) \forall (b \in U) \exists (j \leq p(n)) : b \oplus t_j \in A(x)\}$$

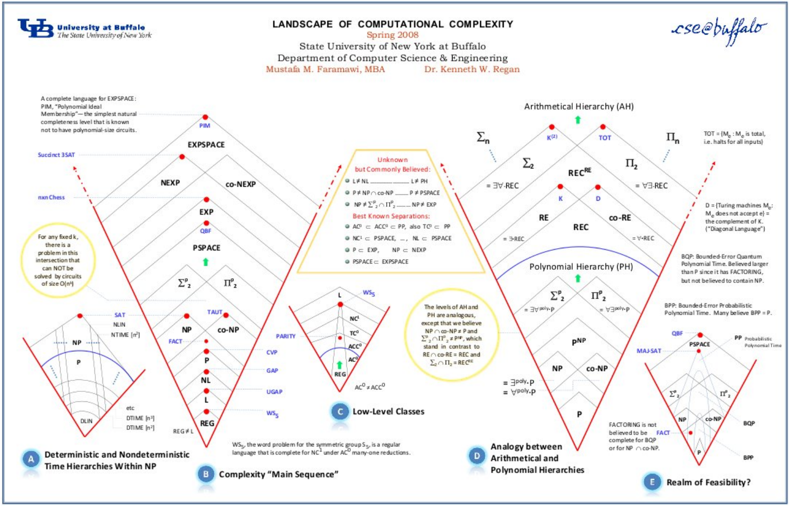
which is precisely the form of languages in $\Sigma_2\mathbf{P}$.

The last existential quantifier $(\exists (j \leq p(n))...)$ affects only polynomially many possibilities,so it doesn't "count" (can by tested in polynomial time by trying all t_j's). $\square$

Extra Properties

- $\Sigma_i\mathbf{P}, \Pi_i\mathbf{P} \subseteq \Sigma_{i+1}\mathbf{P}$
- $\Sigma_i\mathbf{P} \cup \Pi_i\mathbf{P} \subseteq \Delta_{i+1}\mathbf{P} \subseteq \Sigma_{i+1}\mathbf{P} \cap \Pi_{i+1}\mathbf{P}$
- $A, B \in \Pi_i\mathbf{P} \Rightarrow A \cup B \in \Pi_i\mathbf{P}$, $A \cap B \in \Pi_i\mathbf{P}$ and $\overline{A} \in \Sigma_i\mathbf{P}$
- $A, B \in \Delta_i\mathbf{P} \Rightarrow A \cup B$, $A \cap B$ and $\overline{A} \in \Delta_i\mathbf{P}$
- $\mathbf{NP}^{\Sigma_i\mathbf{P} \cap \Pi_i\mathbf{P}} = \Sigma_i\mathbf{P}$

The Complexity Universe



References

The slides were based mainly on:

- *Computational Complexity*, Christos H. Papadimitriou, Addison-Wesley, 1994
  Chapters 14 & 17

And also on:

- *The Theory of Computational Complexity*, Ding Zhu-Du, Ker I Ko, John Wiley and Sons, 2000
  Chapter 3 (The extra properties)

Thank You!