The Polynomial Hierarchy

A. Antonopoulos

18/1/2010
1 Optimization Problems
- Introduction
- The Class DP
- Oracle Classes

2 The Polynomial Hierarchy
- Definition
- Basic Theorems
- BPP and PH
Introduction

TSP Versions

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

(1) \leq_P (2) \leq_P (3) \leq_P (4)
The Polynomial Hierarchy

A.Antonopoulos

Outline
Optimization Problems
Introduction
The Class DP
Oracle Classes

The Polynomial Hierarchy
Definition
Basic Theorems
BPP and PH

DP Class Definition

**Definition**

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

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

Definition
A language \( L \) is in the class \( \text{DP} \) if and only if there are two languages \( L_1 \in \text{NP} \) and \( L_2 \in \text{coNP} \) such that \( L = L_1 \cap L_2 \).

- \( \text{DP} \) is \textit{not} \( \text{NP} \cap \text{coNP} \)!
- Also, \( \text{DP} \) is a \textit{syntactic} class, and so it has complete problems.

**SAT-UNSAT** Definition
Given two Boolean expressions \( \phi, \phi' \), both in 3CNF. Is it true that \( \phi \) is satisfiable and \( \phi' \) is not?
The Polynomial Hierarchy

A. Antonopoulos

Outline

Optimization Problems

Introduction

The Class DP

Oracle Classes

The Polynomial Hierarchy

Definition

Basic Theorems

BPP and PH

Complete Problems for DP

Theorem

SAT-UNSAT is DP-complete.

Proof

Firstly, we have to show it is in DP. Let:

\[ L_1 = \{ (\phi, \phi') : \phi \text{ is satisfiable} \} \]

\[ L_2 = \{ (\phi, \phi') : \phi' \text{ is unsatisfiable} \} \]

It is easy to see, \( L_1 \in NP \) and \( L_2 \in coNP \), thus \( L \equiv L_1 \cap L_2 \in DP \).

For completeness, let \( L \in DP \). We have to show that \( L \leq P SAT-UNSAT \).

\( L \in DP \Rightarrow L = L_1 \cap L_2 \), \( L_1 \in NP \) and \( L_2 \in coNP \).

SAT NP-complete \( \Rightarrow \exists R_1 : L_1 \leq P SAT \) and \( \exists R_2 : L_2 \leq P SAT \).

Hence, \( L \leq P SAT-UNSAT \), by \( R(x) = (R_1(x), R_2(x)) \).
SAT-UNSAT is DP-complete.
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, \( L_1 \in \text{NP} \) and \( L_2 \in \text{coNP} \), thus
  
  \( L \equiv L_1 \cap L_2 \in \text{DP} \).

- For completeness, let \( L \in \text{DP} \). We have to show that
  
  \( L \leq_p \text{SAT-UNSAT} \). \( L \in \text{DP} \Rightarrow L = L_1 \cap L_2 \), \( L_1 \in \text{NP} \) and \( L_2 \in \text{coNP} \).

  SAT NP-complete \( \Rightarrow \exists R_1 : L_1 \leq_p \text{SAT} \) and \( R_2 : \bar{L}_2 \leq_p \text{SAT} \).

  Hence, \( L \leq_p \text{SAT-UNSAT} \), by \( R(x) = (R_1(x), R_2(x)) \)
EXACT TSP is DP-complete.

Proof

- \( EXACT \ TSP \in \text{DP} \), by \( L_1 \equiv TSP \in \text{NP} \) and \( L_2 \equiv TSP \ COMPLEMENT \in \text{coNP} \)

- 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 Assignment (satisfies all clauses except for one)
Complete Problems for DP

**Proof**

We define distances:

1. If \((i, j) \in E(G) \text{ or } E(G')\): \(d(i, j) \equiv 1\)
2. If \((i, j) \not\in E(G), \text{ but } i \text{ and } j \in V(G)\): \(d(i, j) \equiv 2\)
3. Otherwise: \(d(i, j) \equiv 4\)

Let \(n\) be the size of the graph.

1. If \(\phi\) and \(\phi'\) satisfiable, then \(\text{optCost} = n\)
2. If \(\phi\) and \(\phi'\) unsatisfiable, then \(\text{optCost} = n + 3\)
3. If \(\phi\) satisfiable and \(\phi'\) not, then \(\text{optCost} = n + 2\)
4. If \(\phi'\) satisfiable and \(\phi\) not, then \(\text{optCost} = n + 1\)

"yes" instance of \(\text{SAT-UNSAT} \iff \text{optCost} = n + 2\)

Let \(B \equiv n + 2!\)
Other DP-complete problems

Also:

- **CRITICAL SAT**: Given a Boolean expression $\phi$, is it true that it’s unsatisfiable, 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!
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**.

- $PSAT$ is the class of languages decided in pol time with a $SAT$ oracle.
  - Polynomial number of queries
  - Queries computed adaptively
- $SAT$ $NP$-complete $\Rightarrow P^{SAT} = P^{NP}$
- $FP^{NP}$ is the class of functions that can be computed by a pol-time TM with a $SAT$ oracle.
- Goal: $MAX\ OUTPUT \leq_p MAX\-WEIGHT\ SAT \leq_p SAT$
**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 \( 1^n \)?
The Polynomial Hierarchy

A. Antonopoulos

Outline
Optimization Problems
Introduction
The Class DP
Oracle Classes

The Polynomial Hierarchy
Definition
Basic Theorems
BPP and PH

\[ FP^{NP} \text{-complete Problems} \]

**MAX OUTPUT** Definition

Given NTM \( N \), with input \( 1^n \), which halts after \( O(n) \), with output a string of length \( n \). Which is the largest output, of any computation of \( N \) on \( 1^n \)?
**MAX OUTPUT** Definition

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

**Theorem**

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

**Proof**

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

Let $F : \Sigma^* \to \Sigma^* \in FP^{NP}$ $\Rightarrow \exists$ 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**

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 ($\phi_1$):

- If $z_1 = 0$ ($\phi_1$ unsat), then continue from $q_{NO}$.
- If $z_1 = 1$ ($\phi_1$ 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 $= \underbrace{z_1z_2\ldots00}_n$

(Successful computation)
$FP^{NP}$-complete Problems

**Proof**

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 $\phi_j$ 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.
**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 $\text{FP}^{\text{NP}}$-complete.

**Proof**

MAX-WEIGHT SAT is in $\text{FP}^{\text{NP}}$: 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.

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

- **$NTMN(1^n)$** → $\phi(N, m)$:
  Any satisfying truth assignment of $\phi(N, m)$ → 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 $N(1^n)$. 

And the main result:

**Theorem**

*TSP* is \( FP^{NP} \)-complete.
And the main result:

**Theorem**

\(TSP\) is \(\mathsf{FP}^{\mathsf{NP}}\)-complete.

**Corollary**

\(TSP\) \(COST\) is \(\mathsf{FP}^{\mathsf{NP}}\)-complete.
$FP^{NP}$-complete Problems

**Figure:** The overall construction (17-2)
The Class $P^{NP[\log n]}$

**Definition**

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

- $FP^{NP[\log n]}$ is the corresponding class of functions.
The Class $P^{NP[\log n]}$

**Definition**

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

- $FP^{NP[\log n]}$ 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**

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

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

**CLIQUE SIZE Definition**

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

**Theorem**

$CLIQUE SIZE$ is $FP^{NP[\log n]}$-complete.
Conclusion

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

And now,

- \( \text{P}^{\text{NP}} \rightarrow \text{NP}^{\text{NP}} \) ?
- Oracles for \( \text{NP}^{\text{NP}} \) ?
Outline

1 Optimization Problems
   - Introduction
   - The Class DP
   - Oracle Classes

2 The Polynomial Hierarchy
   - Definition
   - Basic Theorems
   - BPP and PH
The Polynomial Hierarchy

Polynomial Hierarchy Definition

- $\Delta_0 P = \Sigma_0 P = \Pi_0 P = P$
- $\Delta_{i+1} P = P^{\Sigma_i P}$
- $\Sigma_{i+1} P = NP^{\Sigma_i P}$
- $\Pi_{i+1} P = coNP^{\Sigma_i P}$
- $PH \equiv \bigcup_{i \geq 0} \Sigma_i P$
The Polynomial Hierarchy

**Polynomial Hierarchy Definition**

- $\Delta_0 P = \Sigma_0 P = \Pi_0 P = P$
- $\Delta_{i+1} P = P^{\Sigma_i P}$
- $\Sigma_{i+1} P = NP^{\Sigma_i P}$
- $\Pi_{i+1} P = coNP^{\Sigma_i P}$
- $\mathbf{PH} \equiv \bigcup_{i \geq 0} \Sigma_i P$
The Polynomial Hierarchy

### Polynomial Hierarchy Definition

- $\Delta_0 P = \Sigma_0 P = \Pi_0 P = P$
- $\Delta_{i+1} P = P^{\Sigma_i P}$
- $\Sigma_{i+1} P = NP^{\Sigma_i P}$
- $\Pi_{i+1} P = coNP^{\Sigma_i P}$

$$PH \equiv \bigcup_{i \geq 0} \Sigma_i P$$

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

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

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

Proof (by Induction)

- For \( i = 1 \)
  \(
  \{x; y : (x, y) \in R\} \in P, \text{so } L = \{x|\exists y : (x, y) \in R\} \in NP
  \)

- For \( i > 1 \)
  If \( \exists R \in \Pi_{i-1} P \), we must show that \( L \in \Sigma_i P \Rightarrow \exists \text{ NTM with } \Sigma_{i-1} P \text{ oracle: NTM}(x) \text{ guesses a } y \text{ and asks } \Sigma_{i-1} P \text{ oracle whether } (x, y) \notin R \).
Basic Theorems

**Proof**

If $L \in \Sigma_i P$, we must show the existence of $R$.

$L \in \Sigma_i P \Rightarrow \exists$ NTM $M^K$, $K \in \Sigma_{i-1} P$, which decides $L$.

$K \in \Sigma_{i-1} P \Rightarrow \exists S \in \Pi_{i-2} P : (z \in K \iff \exists w : (z, w) \in S)$

We must describe a relation $R$ (we know: $x \in L \iff$ 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 $\{x; y : (x, y) \in R\} \in \Pi_{i-1} P$. 
Basic Theorems

Corollary

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

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

Corollary

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

\[
L = \{x : \exists y_1 \forall y_2 \exists y_3 \ldots Qy_i, \text{s.t.} : (x, y_1, \ldots, y_i) \in R\}
\]

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

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 \( \Pi_i \mathbf{P} = \Sigma_i \mathbf{P} \Rightarrow R \in \Sigma_i \mathbf{P} \)

\( (x, y) \in R \iff \exists z : (x, y, z) \in S, S \in \Pi_{i-1} \mathbf{P} \).

Thus, \( x \in L \iff \exists y; z : (x, y, z) \in S, S \in \Pi_{i-1} \mathbf{P} \), which means \( L \in \Sigma_i \mathbf{P} \).
Basic Theorems

**Corollary**

If $P=NP$, or even $NP=coNP$, the Polynomial Hierarchy collapses to the first level.
Basic Theorems

Corollary
If $P=NP$, or even $NP=coNP$, 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.
Corollary

If $P=NP$, or even $NP=coNP$, 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.
Basic Theorems

Corollary
If $P = NP$, or even $NP = coNP$, 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 $\Pi_2 P$, and not known to be in any class below that.
- It is open whether $MINIMUM\ CIRCUIT$ is $\Pi_2 P$-complete.
**QSAT\(_i\) Definition**

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

\[
\exists X_1 \forall X_2 \exists X_3 \ldots QX_i \phi
\]

, where \(Q\) is \(\exists\) if \(i\) is odd, and \(\forall\) if \(i\) is even.
Basic Theorems

**QSAT}_i Definition**

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

$$\exists X_1 \forall X_2 \exists X_3 \ldots \text{Q}X_i \phi$$

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

**Theorem**

For all $i \geq 1$ QSAT}_i is $\Sigma_i \text{P}$-complete.
Basic Theorems

Theorem
If there is a \( \text{PH} \)-complete problem, then the polynomial hierarchy collapses to some finite level.

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

Theorem
If there is a \textsc{ph}-complete problem, then the polynomial hierarchy collapses to some finite level.

Proof
Let \(L\) is \textsc{ph}-complete.
Since \(L \in \textsc{ph}, \exists i \geq 0 : L \in \Sigma_i \text{P}\).
But any \(L' \in \Sigma_{i+1} \text{P}\) reduces to \(L\). Since \textsc{ph} is closed under reductions, we imply that \(L' \in \Sigma_i \text{P}\), so \(\Sigma_i \text{P} = \Sigma_{i+1} \text{P}\).

Theorem
\(\textsc{ph} \subseteq \text{pspace}\)
Basic Theorems

**Theorem**

If there is a \( \text{PH} \)-complete problem, then the polynomial hierarchy collapses to some finite level.

**Proof**

Let \( L \) is \( \text{PH} \)-complete.

Since \( L \in \text{PH} \), \( \exists i \geq 0 : L \in \Sigma_i \text{P} \).

But any \( L' \in \Sigma_{i+1} \text{P} \) reduces to \( L \). Since \( \text{PH} \) is closed under reductions, we imply that \( L' \in \Sigma_i \text{P} \), so \( \Sigma_i \text{P} = \Sigma_{i+1} \text{P} \).

**Theorem**

\( \text{PH} \subseteq \text{PSPACE} \)
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 \( L \in PH \), \( \exists i \geq 0 : L \in \Sigma_i P \).
But any \( L' \in \Sigma_{i+1} P \) reduces to \( L \). Since PH is closed under reductions, we imply that \( L' \in \Sigma_i P \), so \( \Sigma_i P = \Sigma_{i+1} P \).

Theorem
\( PH \subseteq PSPACE \)

\( \text{PH} \quad ? \quad \text{PSPACE} \) (Open). If it was, then PH has complete problems, so it collapses to some finite level.
Theorem

\[ \text{BPP} \subseteq \Sigma_2^P \cap \Pi_2^P \]

Proof

Because \( \text{coBPP} = \text{BPP} \), we prove only \( \text{BPP} \subseteq \Sigma_2^P \).

Let \( L \in \text{BPP} \) (\( L \) 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:

\( a \oplus b = c \iff c \oplus b = a \), so “\( \oplus b \)” is 1-1.
BPP and PH

**Proof**

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: \( t_1, t_2, \ldots, t_{p(n)} \in U \).

For \( b \in U \), these translations *cover* \( b \), if \( b \in A(x) \oplus t_j, \ j \leq p(n) \).

\[
\begin{align*}
\Pr[b \not\in A(x) \oplus t] &= \frac{1}{2^n} \\
\Pr[b \text{ is not covered by any } t_j] &= 2^{-np(n)} \\
\Pr[\exists \text{ point that is not covered}] &\leq 2^{-np(n)}|U| = 2^{-(n-1)p(n)}
\end{align*}
\]
Proof
So, \( T = (t_1, \ldots, t_{p(n)}) \) 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 \( U \).
\( (x \in L) \iff (\exists T \text{ that cover all } U) \)
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 \( \Sigma_2 P \).
The last existential quantifier (\( \exists (j \leq p(n)) \ldots \)) affects only \underline{polynomially} many possibilities, so it doesn’t “count” (can by tested in polynomial time by trying all \( t_j \)’s).