Variations on P and NP

Shuangjun Zhang

Fudan University

zhangsj17@fudan.edu.cn

June 16, 2019

Overview

1. The Polynomial-Time Hierarchy (PH)
   - Alternation of Quantifiers
   - Collapse of the Polynomial-time Hierarchy
   - Complete problems for levels of PH
   - Definition the PH via Oracle Machine

2. Non-uniform Polynomial Time (P/poly)
   - Boolean Circuits
   - Machines That Take Advice

Alternation of Quantifiers

Definition (The class Σ_k)

For a natural number k, a decision problem $S \subseteq \{0,1\}^*$ is in Σ_k if there exists a polynomial p and a polynomial-time algorithm V such that $x \in S$ if and only if

$$\exists y_1 \in \{0,1\}^{p(|x|)} \forall y_2 \in \{0,1\}^{p(|x|)} \exists y_3 \in \{0,1\}^{p(|x|)} ... Q_k y_k \in \{0,1\}^{p(|x|)}$$

$$s.t. V(x, y_1, ..., y_k) = 1$$

Note That $\Sigma_1 = NP$ and $\Sigma_0 = P$.
The Polynomial-time Hierarchy, denoted PH, is the union of all the aforementioned classes(i.e., $PH = \cup_k \Sigma_k$)

Example

INDSET = {<G,k>: $\exists$ S $\subseteq$ V(G) s.t. |S| $\geq$ k and $\forall$ u,v $\in$ S, uv $\notin$ E(G)}
INDSET $\in$ NP, the witness is S.
EXACT INDSET = {<G,k>: the largest independent set in G has size exactly k}
<G,k> $\in$ EXACT INDSET $\Leftrightarrow$ $\exists$ a size-k subset S of V(G) and $\forall$ size greater than k subset S', S is an independent set in G and S' is not.
So, EXACT INDSET $\in \Sigma_2$

Alternation of Quantifiers

Definition (The class Π_k)

$\Pi_k = co\Sigma_k$,where $co\Sigma_k = \{\{0,1\}^* \backslash S : S \in \Sigma_k\}$

Note $\Pi_1 = coNP$

Theorem

For every $k \geq 0$, a set S is in Σ_{k+1} if and only if there exists a polynomial p and a set $S' \in \Pi_k$ such that $S = \{x : \exists y \in \{0,1\}^{p(|x|)} s.t. (x,y) \in S'\}$.

if $PH = P$, then $P = NP$

$\Sigma_k \subseteq \Pi_{k+1} \subseteq \Sigma_{k+2}$

Collapse of the Polynomial-time Hierarchy

We believe that $P \neq NP$ and $NP \neq coNP$. It is commonly conjectured that$\Sigma_k \neq \Pi_k$, for all $k \in N$

Theorem

For every $k \geq 1$,if $\Sigma_k = \Pi_k$ then $\Sigma_{k+1} = \Sigma_k$, which in turn implies $PH = \Sigma_k$

Collapse of the Polynomial-time Hierarchy

Proof.

For any set S in Σ_{k+1}, $\exists p$ and a set $S' \in \Pi_k$ such that
$S = \{x : \exists y \in \{0,1\}^{p(|x|)} s.t. (x, y) \in S'\}$
Using the hypoyhesis, $S' \in \Sigma_k$, So,$\exists p'$ and a set $S'' \in \Pi_{k-1}$ such that
$S' = \{x' : \exists y' \in \{0,1\}^{p'(|x'|)} s.t. (x', y') \in S''\}$
it follows that
$S = \{x : \exists y \in \{0,1\}^{p(|x|)} \exists z \in \{0,1\}^{p'(|(x,y)|)} s.t. ((x, y), z) \in S''\}$
By collapsing the two adjacent existential quantifier, we conclude that
$S \in \Sigma_k$
If $\Sigma_{k+1} = \Sigma_k (\Pi_{k+1} = \Pi_k)$,
For any set S in Σ_{k+2}, $\exists p$ and a set $S' \in \Pi_{k+1}$ such that
$S = \{x : \exists y \in \{0,1\}^{p(|x|)} s.t. (x, y) \in S'\}$
$S' \in \Pi_{k+1} = \Pi_k$, so, $S \in \Sigma_{k+1}$, thus, $\Sigma_{k+2} = \Sigma_{k+1}$, and similarly
$\Sigma_{j+2} = \Sigma_{j+1}$ for all $j \geq k$, Thus $\Sigma_{k+1} = \Sigma_k$ implies $PH = \Sigma_k$ □

if $\Sigma_1 = \Sigma_0$, then $PH = \Sigma_0 = P$

Complete problems for levels of PH

Definition (Σ_i-complete)

A language L is Σ_i-complete, if $L \in \Sigma_i$ and $\forall L^{'} \in \Sigma_i, L' \leq_p L$.

Example (Σ_i-complete problem)

$\Sigma_i SAT = \exists u_1 \forall u_2 ... Q_i u_i \phi(u_1, u_2, ..., u_i) = 1$

Notice that $\Sigma_i SAT$ is a special case of TQBF.

Complete problems for levels of PH

Theorem

If there exists a language L that is PH-complete. then there exists an i such that $PH = \Sigma_i$.

Proof.

$L \in PH = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup ...$
So, $\exists$ i, $L \in \Sigma_i$.
Since L is PH-complete, We can reduce every language of PH to L.
But $L \in \Sigma_i$, $\forall$ L' $\in$ PH, if $L' \leq_p L$ then $L' \in \Sigma_i$
$(...\Sigma_{i-2} \subseteq \Pi_{i-1} \subseteq \Sigma_i \subseteq \Pi_{i+1} \subseteq \Sigma_{i+2}...)$
We have $PH \subseteq \Sigma_i$, So, $PH = \Sigma_i$. □

PH = PSPACE?

$\forall$ i, $\Sigma_i \subseteq$ PSPACE
PH $\subseteq$ PSPACE
PH = PSPACE?
If PH = PSPACE, then TQBF will be PH-complete.

Oracle Turing Machine (OTM)

Define an OTM M^O as:

- M is just a Turing Machine with normal computation.
- M also has access to an Oracle, denoted by O.
  - M can ask any question to O during its computation.
  - O always answers correctly and in constant time.
- $M^O(x)$:Output of M^O on input x.

Definition (P^A and NP^A)

P^A is the class of languages decidable with a polynomial time oracle Turing machine that uses oracle A. Define the class NP^A similarly.

For example, NP $\subseteq P^{SAT}$

Oracle Turing Machine (OTM)

Define $C_1^{C_2}$: M is a TM that is in complexity class C_1.

- P^{NP}: a DTM with access to oracle in class NP.
  - $P^{NP} = P^{SAT}$.
- P^{P}: a DTM with access to oracle in class P.
  - $P^{P} =$ P.
- NP^{NP}: a NTM with access to oracle in class NP.
  - Clearly NP $\subseteq NP^{NP}$.
  - Question: $NP^{NP} \subseteq$ NP?

Oracle Turing Machine (OTM)

Theorem

$\overline{SAT} \in P^{SAT}$

Proof.

To decide if formula ϕ is unsatisfiable, M asks oracle SAT if $\phi \in SAT$ and gives its opposite answer. □

Therefore: $coNP \subseteq P^{NP} \subseteq NP^{NP}$, if $NP^{NP} \subseteq$ NP, then coNP $\subseteq$ NP.

Definition the PH via Oracle Machine

Now we can define the polynomial heirarchy another way!

$$\Sigma_1 = NP$$

$$\Pi_1 = coNP$$

$$\Sigma_2 = NP^{NP}$$

$$\Pi_2 = coNP^{NP}$$

$$\Sigma_i = NP^{\Sigma_{i-1}}$$

$$\Pi_i = coNP^{\Sigma_{i-1}}$$

Claim: The above relations are correct.

Definition the PH via Oracle Machine

Theorem

$\forall i \geq 2$, $\Sigma_i = NP^{\Sigma_{i-1}}$, *where the latter class denotes the set of languages decided by the polynomial-time NTM 's with access to the oracle* Σ_{i-1}.

- We will only prove for $\Sigma_2 = NP^{NP} = NP^{SAT}$.
- We have to prove both ways:

$$L \in \Sigma_2 \Rightarrow L \in NP^{SAT}$$

$$L \in NP^{SAT} \Rightarrow L \in \Sigma_2$$

First direction

$$L \in \Sigma_2 \Rightarrow L \in NP^{SAT}$$

- Suppose that $L \in \Sigma_2$,then $\exists$ PDTM M s.t.

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 M(x, u_1, u_2) = 1$$

- Fixing u_1, "$\forall u_2\ M(x \circ u_1, u_2) = 1$" is in coNP.
- Consider a NTM N with coNP as an oracle.
- Guess u_1, and use oracle to decide $\forall u_2\ M(x \circ u_1, u_2) = 1$
- L $\in NP^{coNP} = NP^{\overline{SAT}} = NP^{SAT}$

Other direction

$$L \in NP^{SAT} \Rightarrow L \in \Sigma_2$$

- Suppose L is decidable by a polynomial-time NDTM N with oracle access to SAT.
- x is in L iff there exists a sequence of nondeterministic choice and correct oracle answers that makes N accept x. That is there is a sequence of choice $c_1, c_2, ..., c_m \in \{0, 1\}$ and answers to oracle queries $a_1, ..., a_k \in \{0, 1\}$ s.t. if N uses the choice $c_1, c_2, ..., c_m$ in its execution and receives a_i as the answer to its ith query, then:
  - (1). N reaches the q_{accept}.
  - (2). All the answer are correct.
- Let ϕ_i denote the ith query that M makes to its oracle.
- The condition(2) can be phrased as follow:
  - If $a_i = 1$, then $\exists\ u_i$ s.t. $\phi_i(u_i) = 1$
  - If $a_i = 0$, then $\forall\ v_i$ s.t. $\phi_i(v_i) = 0$

Other direction

Thus, we have that
x $\in$ L $\Leftrightarrow \exists$ $c_1, ..., c_m, a_1, ..., a_k, u_1, ..., u_k \ \forall \ v_1, ..., v_k$ s.t.
N accept x using choices $c_1, ..., c_m$ and answers $a_1, ..., a_k$ AND
$\forall i \in [k]$ if $a_i = 1$ then $\phi_i(u_i) = 1$
$\forall i \in [k]$ if $a_i = 0$ then $\phi_i(v_i) = 0$
This implies $L \in \Sigma_2$.
So, $\Sigma_2 = NP^{NP}$, $\Sigma_3 = NP^{NP^{NP}}$ and so on.

Boolean Circuits

Definition (Boolean Circuits)

A Boolean circuit is a directed acyclic graph with internal nodes marked by elements of $\wedge, \vee, \neg$. Nodes with no in-going edges are called input nodes, and nodes with no out-going edges are called output nodes.

Bounded fan-in. We will be most interested in circuits in which each gate has at most two incoming edges.
For concreteness and simplicity, we assume throughout this section that all circuits have bounded fan-in.

Boolean Circuits

Theorem (circuit evaluation)

There exists a polynomial-time algorithm that, given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ *and an n-bit long string x, returns* $C(x)$.

Definition (family of circuits computes a function)

We say that a family of circuits $\{C_n\}_{n \in N}$computes a function $f : \{0,1\}^* \to \{0,1\}^*$ if for every n the circuit C_n computes the restriction of f to strings of length n. In other words, for every $x \in \{0,1\}^*$, it must hold that $C_{|x|}(x) = f(x)$.

The size of a circuit is defined as the number of edges.

Machines That Take Advice

Definition (Machines That Take Advice)

We say that a function f is computed in polynomial time with advice of length $l : N \to N$ if these exists a polynomial-time algorithm A and an infinite advice sequence $(a_n)_{n \in N}$ such that

1. For every $x \in \{0,1\}^*$, it holds that $A(x, a_{|x|}) = f(x)$.
2. For every $n \in N$, it holds that $|a_n| = l(n)$.

Machines That Take Advice

Definition (P/poly)

We say that a computational problem can be solved in polynomial time with advice of length l if a function solving this problem can be computed within these resources.We denote by P/l the class of decision problems that can be solved in polynomial time with advice of length l, and by $P/poly$ the union of P/p taken over all polynomials p.

Clearly, $P/0 = P$

The Power of Advice

Theorem (The Power of Advice)

There exist functions that can be computed using one-bit advice but cannot be computed without advice.

Proof.

Starting with any uncomputable Boolean function $f : N \to \{0,1\}$,consider the function f' defined as $f'(x) = f(|x|)$.Note that f is Turing-reducible to f' (e.g.,on input n make any n-bit query to f', and return the answer).Thus, f' cannot be computed without advice. On the other hand, f' can be easily computed by using the advice sequence $(a_n)_{n\in N}$ such that $a_n = f(n)$, that is, the algorithm merely outputs the advice bit(and indeed $a_{|x|} = f(|x|) = f'(x)$, for every $x \in \{0,1\}^*$) □

The End