Wrapping Up NP-Completeness

Frank

March 30, 2022

1 How to deal with NP-Completeness

NP complete problems probably do not have efficient solutions, so what do we do?

1. Give up on worst case time complexity

   Ex. average time complexity (but it is very complex, and simplication needed to do the analysis, ex. assumptions, may render the argument impractical)

   One reason that certain algorithms do not have efficient solutions is because of their exponential search space. Thus, brute force search through the search space results in an exponential time complexity.

   (a) However, there are sometimes ways to intelligently search through the solution space:
   - backtracking
   - branch and bound

   Ex. using logical arguments to get rid of a large portion of the search space at each iteration.

   (b) Develop general purpose algorithms for central problems like:
   - SAT solver
   - ILP solver

2. Develop polytime algorithms for special cases of NP complete problems.
   - 2SAT (linear)
   - 2DM (polytime using maxFlow)

3. Pseudo-polytime algorithms

   Ex. polynomial in value of input (ex. numeric problems - we can usually represent numbers in binary)
   - Subset sum (pseudo-polytime: need to iterate from 1 to the number we are searching for, which is exponential compared to the representation of the number itself)

4. Give up optimality, and settle for good enough
   - local search (search for optimal solution within a local context, as opposed to a global optimal)

     ex. travelling salesman (picking different paths and compare if it is better locally)

5. Instead of exact solution, obtain polytime approximation algorithms with guaranteed closeness to optimal
   - 2-approximation

2 Class CoNP (complement of NP languages)

2.1 Recall...

P = languages decided by a polytime DTM. (efficiently decidable)

NP = languages decided by a polytime NTM. ($P \subseteq NP$)

Note that $L \in P \implies \bar{L} \in P$ (by definition)

P is closed under complementation

But $L \in NP \implies \bar{L} \in NP$? (unknown)

Note that for P, we can simply define a TM M that decides L, and define $\bar{M}$ as another TM that reverses the output of M to decide $\bar{L}$.

However, this does not work for NTM, because for a NTM to accept an input, there only has to be 1 accepting computation path. If we reverse all the results of the computation paths, we probably will still end up accepting the string.

It is widely believed that the implication is false.

Lets take a look at SAT problem.
$\phi \in$ SAT $\iff$ there is a t.a. that satisfies ϕ
We can easily verify that a t.a. satisfies ϕ
consider $\phi \in \overline{SAT} \iff$ all t.a. falsify ϕ
It seems much harder to find a certificate that can verify this.
So it seems very likely that $\overline{SAT} \notin NP$
But we can say that $\overline{SAT} \in coNP$

2.2 Definition

Def: $coNP = \{L : \bar{L} \in NP\}$

Recall that NP problems have short, efficiently verifiable certificates for yes-instances of the problem

Now, coNP problems have short, efficiently verifiable certificates for no-instances of the problem

Conjecture: $coNP \neq NP$

2.3 Problems

- $\phi \in TAUT \iff$ all t.a. satisfy ϕ

  Note that we can give a single falsifying t.a. to prove that $\phi \notin TAUT$, so we say that $\overline{TAUT} \in NP$ and $TAUT \in coNP$

- $G \in DHC \iff$ G has a cycle that visits every node exactly once

  It is difficult to find a certificate that verifies in polytime that there does not exist such cycle.

  So we believe $\overline{DHC} \in coNP$

2.4 Theorems

Thm 11.1: $P \subseteq NP \cap coNP$

Proof:

Suppose $L \in P$
$\implies \bar{L} \in P$ [P is closed under compl.]
$\implies \bar{L} \in NP$ [$P \subseteq NP$]
$\implies L \in coNP$ [by def of coNP]

We can actually draw some analogy between this thm and computability.

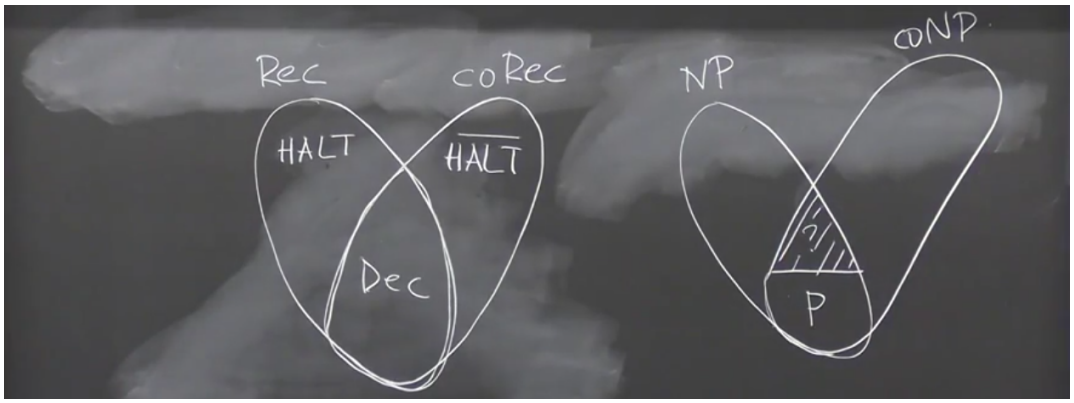


Figure 1: t1

However, we do not know if $P = NP \cap coNP$ like we know that $Dec = Rec \cap coRec$

Thm 11.2: $NP = coNP \iff NP$ is closed under complementation

Proof:

$NP = coNP \iff \forall L,\ L \in NP \iff L \in coNP$ ($\bar{L} \in NP$)
$\iff \forall L,\ L \in NP \iff \bar{L} \in NP$
$\iff NP$ closed under compl.

Thm 11.3: $NP \neq coNP \implies P \neq NP$ [$P = NP \implies NP = coNP$]

Proof:

Assume $P = NP$
$\implies NP$ is closed under compl. [since P is closed under compl.]
$\implies NP = coNP$ [Thm 11.2]

Converse of this theorem is unknown.

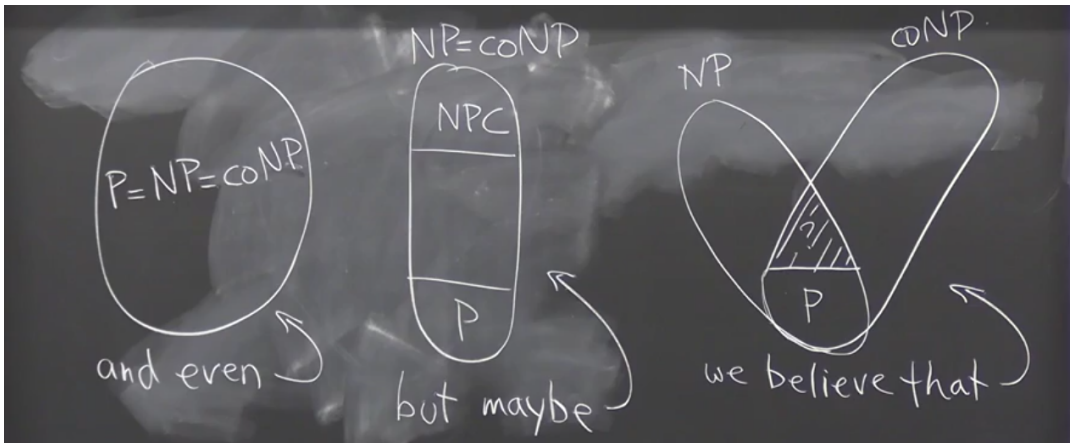


Figure 2: t2

Def: L is coNP-complete:

1. $L \in coNP$
2. $\forall L' \in coNP$, $L' \leq_m^P L$

Note: $L_1 \leq_m^P L_2 \implies \bar{L_1} \leq_m^P \bar{L_2}$

Thm 11.4: L is coNP-complete $\iff \bar{L}$ is NP-complete

Proof:

L is coNP-complete $\implies L \in coNP$ and All $L' \in coNP$ reduces to L
$\implies \bar{L} \in NP$ and by above property, all $\bar{L'} \in NP$ reduces to $\bar{L}$
$\implies \bar{L}$ is NP-complete

Thm 11.5: $NP = coNP \iff \exists L \in NPC$ s.t. $\bar{L} \in NP$

Proof:

($\implies$)

Assume $NP = coNP$
$\implies NP$ is closed under compl. [Thm 11.2]
$\implies \forall$ NPC L, $\bar{L} \in NP$

($\impliedby$)

Let $L \in NPC$ s.t. $\bar{L} \in NP$
Let $L' \in NP$
$\implies L' \leq_m^P L$ [b/c L is NPC]
$\implies \bar{L'} \leq_m^P \bar{L}$
Since $\bar{L} \in NP$, there is a polytime NTM M that decides $\bar{L}$ [b/c $\bar{L} \in NP$]
$\implies$ there is a polytime NTM M' that decides $\bar{L'}$

M' works as follows:

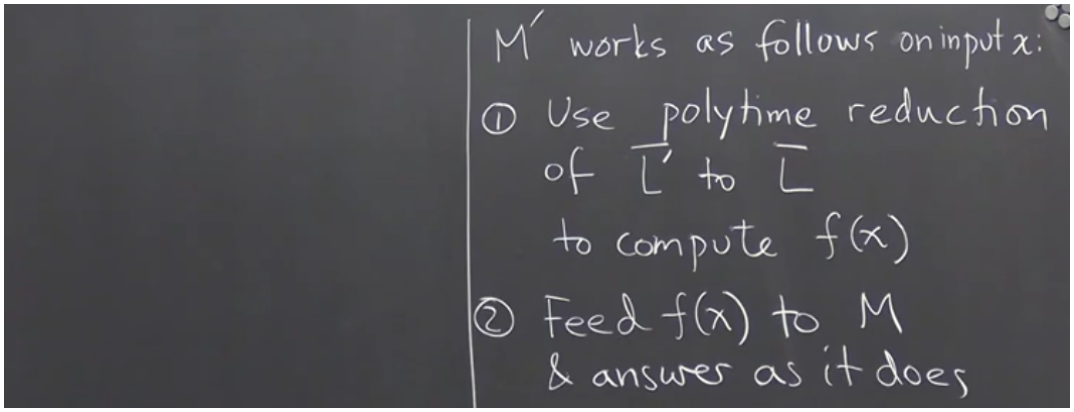


Figure 3: t3

$\implies \bar{L}' \in NP$
$\implies NP$ closed under compl.
$\implies NP = coNP$

3 Alternate Characterizations

Recall that $L \in NP$ means

- $x \in L \iff \exists y, |y| \leq |x|^k$, s.t. predicate $R(x, y)$ is polytime

For example, $\phi \in SAT \iff \exists \tau$ s.t. τ satisfies ϕ where $R(x, y) =$ "τ satisfies ϕ"

Now, $\phi \in \overline{SAT} \iff \neg(\exists \tau$ s.t. τ satisfies $\phi)$

$\iff \forall \tau$, τ falsifies ϕ, where $S(x, y) =$ "τ falsifies ϕ"

So coNP is: "for all certificates, something that can be checked in polytime is true"

In this case, something that can be checked in polytime is whether a t.a. τ falsifies ϕ, but for coNP membership, we need to check all certificates and verify that it is true.

For DHC,

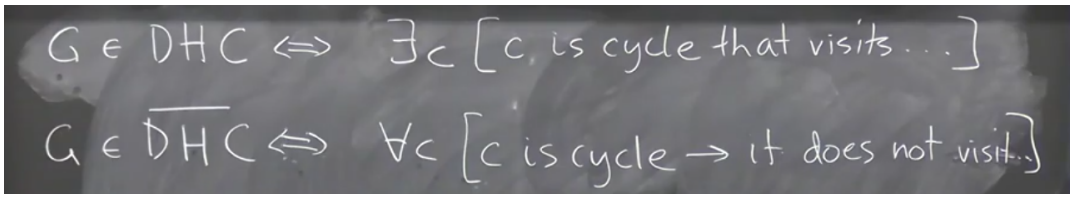


Figure 4: t4

So we see that

For NP, we have $\exists y$, $R(x, y)$
For coNP, we have $\forall y$, $S(x, y)$

4 New Complexity Classes

However, if we have the form $x \in L \iff \exists y \forall z,\ R(x, y, z)$

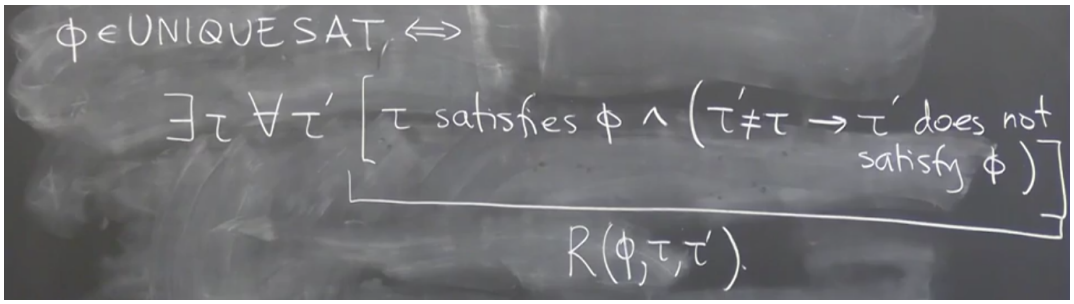


Figure 5: t5

We can express any NP problem with uniqueness requirement in this form.

It doesn't seem like we can give a certificate to validate the yes-instance nor the no-instance that is of a reasonable size and can be validated in polytime

So it seems like problems of this form are neither NP nor coNP

Class of such L is called Σ_2^P where 2 is the number of quantifier alternation and first quantifier is $\exists$

Consider coΣ_2^P

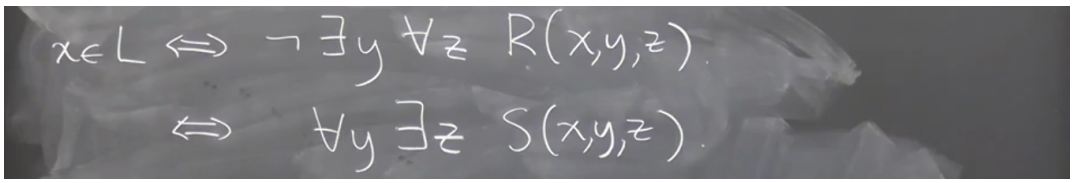


Figure 6: t6

This class is called Π_2^P

A language in this class is NOTUNIQUESAT

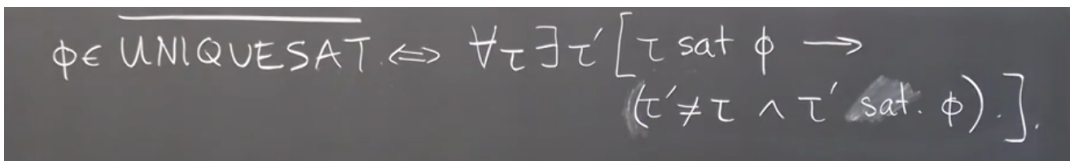


Figure 7: t7

Another language in this class is MINFORM (minimum formula)

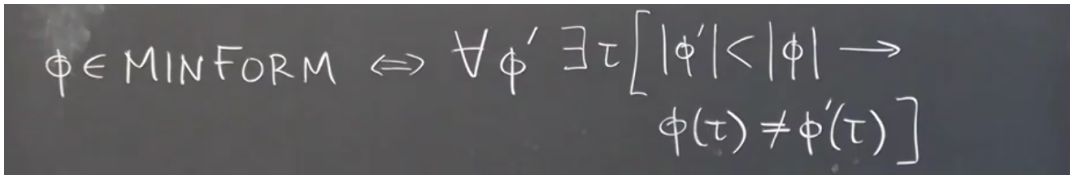


Figure 8: t8

Of course, we have Σ_k^P and $\text{co}\Sigma_k^P = \Pi_k^P$

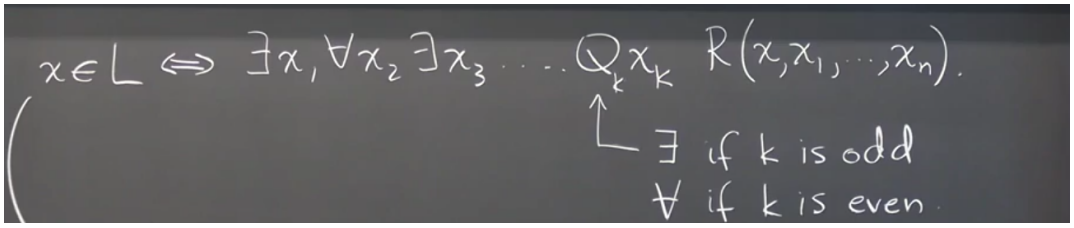


Figure 9: t9

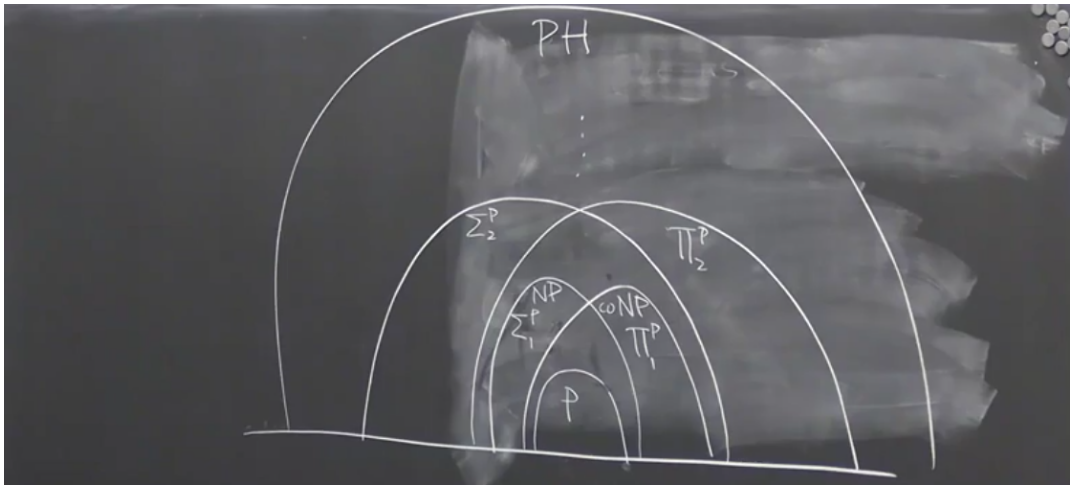
The union of all these classes is called **polynomial time hierarchy**

$\text{PH} = \bigcup_{k>0} \Sigma_k^P = \bigcup_{k>0} \Pi_k^P$

Note the special cases:

- $\Sigma_1^P = \text{NP}$
- $\Pi_1^P = \text{coNP}$

We believe this is a real hierarchy (problems get increasingly harder as we go up the hierarchy)



Thm: $\Pi_i^P = \Sigma_i^P \implies \forall j > i, \Pi_j^P = \Sigma_j^P = \Pi_i^P = \Sigma_i^P$