Take Home Exam 1

Mahya Jamshidian
Computatinal Complexity

March 31, 2019

Problem 1. a
$\forall L \in \mathbf{P} : L \cup \overline{L} \in \mathbf{P} \Rightarrow \overline{L} \in \mathbf{P}$[1]
$\mathbf{P} = \mathbf{NP}$ and $1 \Rightarrow \forall L \in \mathbf{NP} : L \cup \overline{L} \in \mathbf{NP} \Rightarrow \overline{L} \in \mathbf{NP} \Rightarrow L \in \mathbf{CoNP} \Rightarrow \mathbf{NP} \subseteq \mathbf{CoNP}$[2]
$\forall L \in \mathbf{CoNP} : \overline{L} \in \mathbf{NP} \Rightarrow L \in \mathbf{NP} \Rightarrow \mathbf{CoNP} \subseteq \mathbf{NP}$[3]
2 and 3 $\Rightarrow \mathbf{NP} = \mathbf{CoNP} = \mathbf{P}$

Problem 1. b
L being in CoNP (hint) : an string u of length m serves as certifiacte in this problem. for all certificates indicating assignment of variables y_1,y_2,y_3,...,y_m if ϕ is $TRUE$ then the NDTM M will output YES and NO otherwise and this can be checked in polynomial-time; which is the definition for the language being in **CoNP**.
for another NDTM $M^{'}$, now we set the certificate $u^{'}$ to be an string in Σ^* of length n indicating assignment of varibles x_1,x_2,x_3,...,x_n, we give (ϕ,$u^{'}_0$ $u^{'}_1$ ... $u^{'}_{n-1}$) as input to M. if M outputs YES, then $M^{'}$ outputs YES and if M outputs NO for all $u^{'}$ then $M^{'}$ will output NO.
Now if $\mathbf{NP} = \mathbf{CoNP} = \mathbf{P}$ (1.a) then there exists some DTM deciding L(M); Thus checking wether $u^{'}$ is a valid assignment would take polynomial time. therefore L($M^{'}$) $\in \mathbf{NP}$.
$\mathbf{SinceNP} = \mathbf{CoNP} = \mathbf{P}, \mathbf{L}$(M$^{'}$)(GBF) is in $\mathbf{P}$.

Problem 2. a
$\mathbf{P}^* \subseteq \mathbf{NP}^*$
Consider any instance in $\mathbf{P}^*$ like S; By defenition, there exists some DTM D(x) that computes x in polynomial time and outputs y.
Now we construct another DTM $D^{'}$ that takes some x and y as input and outputs YES if D(x)=y and $y \leq p(|x|)$ (p(n) is some polynomial).
$D^{'}(x, y)$ construction consists of D(x)= $y^{'}$ and some comparison between $y^{'}$ and y; if they are equal, $D^{'}$ will output YES and NO otherwise.
since D runs in polynomial time $p^{'}(|x|)$ on input x, length of $y^{'}$ at most would be $p^{'}(|x|)$ and the comparison would take $p^{'}(|x|)$ as well; so we set p(n) = $2p^{'}$(n).
We construct $D^{'}$ as some DTM with (x,y) as input for every instance in $\mathbf{P}^* \Rightarrow \mathbf{P}^* \subseteq \mathbf{NP}^*$

Problem 2. b

$\mathbf{P} = \mathbf{NP} \Rightarrow \mathbf{P}^* = \mathbf{NP}^*$

We need to show that if $\mathbf{P} = \mathbf{NP}$, then for every polynomial-time TM M and polynomial p(n), there is a polynomial-time TM B with the following property: for every $x \in \{0,1\}^n$, if there is $y \in \{0,1\}^{p(n)}$ such that M(x, y) = YES (a soloution that x is in the language computed by M) then $|B(x)| = p(n)$ and $M(x, B(x)) = YES$.

We start by showing the theorem for the case of SAT. In particular, we show that given an algorithm A that decides SAT, we can come up with an algorithm B that on input a satisfiable CNF formula ϕ with n variables, finds a satisfying assignment for ϕ using n + 1 calls to A and some additional polynomial-time computation.

The algorithm B works as follows: We first use A to check that the input formula ϕ is satisfiable. If so, we first substitute $x_1 = 0$ and then use A to decide if it is satisfiable. Say it is satisfiable. Then we fix $x_1 = 0$ (it is trivial that if it is not, we fix x_1 to be 1) from now on and continue with the simplified formula. Continuing this way, we end up fixing all n variables while ensuring that each intermediate formula is satisfiable. Thus the final assignment to the variables satisfies ϕ.

To solve the search problem for an arbitrary NP-language L, we use the fact that the reduction from L to SAT (cook-levin theorem) is actually a Levin reduction. This means that we have a polynomial-time computable function f such that not only $x \in L \iff f(x) \in SAT$ but actually we can map a satisfying assignment of f(x) into a soloution for x. Therefore, we can use the algorithm above to come up with an assignment for f(x) and then map it back into a soloution for x.

$\mathbf{P}^* = \mathbf{NP}^* \Rightarrow \mathbf{P} = \mathbf{NP}$

$\mathbf{P}^* = \mathbf{NP}^* \Rightarrow$ there exists some polynomial time DTM M for all problems in NP^* which gets an $x \in \{0,1\}^*$ and returns a $y \in \{0,1\}^*$ such that $(x, y) \in S$ if such y exists and returns NO otherwise.

By defenition, a problem in NP has some DTM M :

$$x \in L \iff \exists u \in \{0,1\}^{p^{'}(|x|)} s.t. M(x,u) = YES$$

for all languages in **NP**, we can convert M to some DTM $M^{'}$ which takes (x,y) $\in \Sigma^* \times \Sigma^*$ and return YES iff M(x,y) = YES. That is a certificate for decision problems in **NP** can serve as soloution for that problem in its search form.

all decision problems in **NP** have one search-like candidate in $\mathbf{NP}^*$. and we know $\mathbf{P}^* = \mathbf{NP}^*$ $\Rightarrow$ there exists some polynomial time DTM M for all problems in NP^* which gets an $x \in \{0,1\}^*$ and returns a $y \in \{0,1\}^*$ such that $(x, y) \in S$ if such y exists and returns NO otherwise.

it is obvious that each DTM M that computes some language in $\mathbf{P}^*$, can be easily modified to decide its decision-like problem which is in **P**(if it outputs anything other than NO, its a YES and it is in polynomial time).

$$\forall L \in \mathbf{NP} : \mathbf{L} \in \mathbf{P} \longrightarrow NP \subseteq P$$

and we know $P \subseteq NP$:

$$NP = P$$

Problem 3. a
Γ_1 and Γ_2 are not equal; Γ_1 is the class of all of the languages that are decideable using NDTM N such that for every $x \in \{0,1\}^*$, the computation tree of N(x) admits at most one accepting path(condition is on NDTM).
Although, Γ_2 is the class of all languages that the elements of that language admits at most one accepting path; that means even if some x has been accepted by NDTM $N^{'}$ but it has more than one accepting path(or not accepted at all), it is not in L($N^{'}$)(the condition is on computation tree).

Problem 3. b
it is not known if Γ_1 has any complete problem.(Sipser in lectures in computer science.)
but as in Γ_2 we can construct a complete language, namely modified-SAT that all boolean formulas with 0 and exactly one assignment that makes it TRUE. Since all NDTMs in Γ_2 can be simulated by some Boolean formula using cook intuition, s.t if constructed φ contains 0 or 1 satisfying assignment , so will simulated NDTM.

Problem 3. c
USAT is not in Γ_2 Since it has not boolean formulas that do not satisfy under any assignment; However there is some complexity class US (Unique Polynomial) which contains strings in L(N) for N being some NDTM, s.t. it has one computation path.
obviously USAT is in US and it is US-complete :
we can use cook-levin intuition and make a boolean formual simulating NDTM N for some language in US, and if that boolean formula has one unique accepting path then N will have only one too. (since its the choice between two transition functions δ_0 and δ_1 can make different paths along computation tree (choosing either is simulated using conjuction)).

Problem 3. d
$\mathbf{P \subseteq \Gamma_1}$
$\Gamma_1 \subseteq \mathbf{NP} \cap \mathbf{\Gamma_2}$ Since both **NP** and Γ_2 contains Γ_1
$NP \cap \Gamma_2 \subseteq \mathbf{NP}$
$NP \subseteq Co - \Gamma_1$ since $\Gamma_1 \subseteq \mathbf{NP}$

Problem 3. e
both of the classes are closed under union; since any of checking can be done in polynomial time. it means for arbitary NDTM N_1 and N_2 which L(N_1) and L(N_2) $\in \Gamma_1$, there exists NDTM N_3 s.t. L(N_3)= L(N_1) $\cap$ L(N_2) and L(N_3) $\in \Gamma_1$; Since checking each certificate against input for each of N_1 and N_2 takes polynomial time and if that one accepting path for one input in both machines matches, then for every input there exists at most one accepting path which is that matching path.
for Γ_2, since every element of L_1 and $L_2 \in \Gamma_2$ has at most one accepting path, elements of $L_1 \cap L_2$ has at most one accepting path that is possible if certificate under same input leads to accept and only accept.

Problem 4. a
first we construct a polynomial-time reduction from 3SAT to NAE 4SAT and then to NAE 3SAT and to language in problem statement.

$$\text{3-SAT} \leq_{\text{p}} \text{NAE 4-SAT}$$

We are to show that ϕ is satisfiable if and only if ϕ is satisfiable. If x_1, . . . , x_n is a satisfying assignment for ϕ, then the same assignment satisfies ϕ when we choose z = 0.

The reason is

$$\text{a} \vee \text{b} \vee \text{c} = \text{NAE(a, b, c, 0)}$$

To show the other direction, suppose x_1, . . . , x_n, z is a satisfying assignment of ϕ. It is obvoius that $\neg\, x_1$, . . . ,$\neg\, x_n$,$\neg$z is also a satisfying assignment of ϕ (because NAE(a, b, c, d) = NAE($\neg$a,$\neg$b,$\neg$c,$\neg$d)). In one of these two assignments, the value assigned to the variable z is 0. This assignment corresponds to a satisfying assignment for ϕ.

$$\text{NAE 4-SAT} \leq_{\text{p}} \text{NAE 3-SAT}$$

it is obvious that by a 4-SAT (a, b, c, d) we can construct a 3-SAT in polynomial time. we put two of the literals in each clause C_i into another clause with on additional literal w_i s.t. $NAE(a, b, w_i) = NAE(c, d, \neg w_i)$ and two generated clause for 3-SAT are (a, b, w_i) and $(c, d, \neg w_i)$.

$$\text{NAE 3-SAT} \leq_{\text{p}} \text{HSP}$$

now we apply the following reduction from NAE 3SAT. For every 3CNF ϕ with variables x_1, . . . , x_n and clauses C_1, . . . , C_m, set $s = m + n$, $k = n$ and we are to choose k(n) variables for S_1 and the rest goes to S_2 so both partitions have the size of k :

$$\text{A} = \{\ x_1,x_2,x_3,...,x_n,\overline{x_1},...,\overline{x_n}\}$$
$$B_i = \{y_1,y_2,y_3\} \text{ for every clause } C_i = (y_1 \vee y_2 \vee y_3);\ \text{i} = 1,2,3...,\text{m}$$
$$B_{i+m} = \{x_1,\overline{x_1}\}\ \text{i} = \text{m+1}\ ,\ ...\ ,\ \text{m+n}$$

now we show ϕ is satisfiable iff above instance of HSP is valid.
ϕ is satisfiable $\Rightarrow$ above instance of HSP is valid
Now that S_1 has size on k (n), it will contain either x_i or its negation; because the S_1 is assignment of ϕ and it can not set both TRUE. Thus it intersects with all subsets B_i i = m+1,...,m+n.
Since S_2 contains the negation of literals in S_1, it will also intersects with B_i i = m+1,...,m+n.
now that we know that literals in clauses C_i of ϕ are not all equal, it means not all of them are TRUE(exist in S_1) and because ϕ is satisfiable not all of them are FALSE (in S_2); Thus both S_1 and S_2 intersect with B_i i = 1,...,n.
above instance of HSP is valid $\Rightarrow$ ϕ is satisfiable
the instance is valid $\Rightarrow$ size of S_1 is n (we set n literals to be TRUE).
according to pigeon hole principle, since size of S_1 is n and it intersects with B_i i = m+1,...,m+n, exactly one literal from each of those subsets are in S_1 and the other one is in S_2 (S_1 And S_2 do not intersect).
S_1 intersects with all B_i i = 1,...,m; Thus there exists at least one TRUE literal in each clause $C_i \Rightarrow \phi$ is satisfiable.

Problem 4. b

when the size of all the subsets B_i i = 1,...,s are equal to 2, it means all of the clauses have 2 literals and therefor we perform a reduction from this problem to NAE 2SAT which is in **P**. For all of the subsets A_i i = 1,...,m, like $\{a,b\}$, we add a ,b , $\overline{a}$ and $\overline{b}$ as vertices to hypothetical graph G, and add edges ($\overline{b}$,a) , ($\overline{a}$,b), (a,$\overline{b}$) and (b,$\overline{a}$) as well (2 latter edges makes a conversion from NAE 2SAT to 2SAT)

we can make a partition in V (G(V,E)) iff no strongly connected component of G contains a variable and its negation.

⇒

We assume there exists some partition (S_1,S_2) in V; if a is in S_1 so is $\overline{b}$ (according to edges); but if b is in S_1 too, all 2 elements from one subset are in one partition, namely S_1 , and thus it has no intersection to S_2 and it contradicts with our assumption.

Now that we constructed a Directed Graph, if there is a path from literal a to its negation $\overline{a}$ then both of them should be in one partition and as in NAE 2SAT, both of them should be TRUE which is not possible and if there is a path from literal $\overline{a}$ to its negation a then both of them should be in one partition and as in NAE 2SAT, both of them should be FALSE which is not possible either.

⇐

let no strongly connected component of G contain x_i and $\neg x_i$ for any $i \in [1, n]$. Order the strongly connected components C_1, . . . , C_m of G in a topological order.

for any variable x, x is in S_1 iff x appears after $\neg x$; else x is in S_2. To prove that this partition is valid, we prove

for no edge (a,b) of the graph, vertex $a \in S_1$ and vertex $b \in S_2$.

Proof. Assuming it is incorrect, let edge (a,b) be a counterexample: $a \in S_1$ and $b \in S_2$. Let a appears in the biconnected component B_i. The reason edge (a,b) is in the graph is subset $\{\overline{a}, b\}$ which also generated edge ($\overline{b}$,$\overline{a}$). Since $a \in S_1$, $\in S_2$, vertex $\overline{a}$ appears before the biconnected component B_i, say B_j where $j < i$. Since $b \in S_2$, vertex $\overline{b}$ appears after B_i, say in B_k, where $k > j$. But any edge from B_k to B_j with $j < k$ contradicts the topological order of the biconnected components. □

Now, the Claim guarantees that (S_1,S_2) is valid. Indeed, if there were a subset {x,y} with x , $y \in S_2$ we would have an edge ($\overline{x}$,y) with $\overline{x} \in S_1$ and $y \in S_2$, contradicting the Claim.

and therefore we reduced $NAE2SAT$ and equivantly language stated in problem statement to Strongly Connected Component Problem which is in P.