IIT PALAKKAD

Indian Institute of Technology Palakkad
Department of Computer Science and Engineering

CS5616 Computational Complexity

January – May 2024

Problem Set – 5 | Total Points – 50

Name: Neeraj Krishna N
Roll no: 112101033

Given on 15 Apr
Due on 05 May

**Instructions**

- Use of resources other than class notes and references is forbidden.
- Collaboration is not allowed. Credit will be given for attempts and partial answers.

1. (10 points) (NP **Completeness** of Nae3Sat) Nae3Sat is the set of all satisfiable 3CNF formulas $\phi$ that have a satisfying truth assignment where there is no clause in $\phi$ with all its literals being true. Show that Nae3Sat is NP-complete. [*Hint:* 3SAT!]

**Solution:**

It suffices to prove 3SAT $\leq_m^p$ Nae3Sat.

**Claim 1.** 3SAT $\leq_m^p$ Nae4Sat *where* Nae4Sat *is the set of all satisfiable 4CNF formulas $\phi$ that have a satisfying truth assignment where there is no clause in $\phi$ with all its literals being true*

*Proof.* The reduction is as follows, with input being $\phi$ an instance of 3CNF and output being $\phi'$ an instance of Nae4Sat

1. For each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ in $\phi$, add 2 clauses $\tilde{C}_i = l_{i,1} \vee l_{i,2} \vee l_{i,3} \vee \tilde{l}_i$ and $\tilde{C}_i{}' = l_{i,1} \vee l_{i,2} \vee l_{i,3} \vee \neg\tilde{l}_i$ to $\phi'$ where $\tilde{l}_i$ is a dummy variable

It is clear that the reduction is poly-time reduction

*Correctness.* $\phi \in 3SAT$
$\Rightarrow \exists$ a satisfying assignment for $\phi$, call it $S$
$\Rightarrow S$ with the variables $\tilde{l}_i$ set to **False** for all $i$ is a satisfying assignment for $\phi'$
$\Rightarrow \phi' \in$ Nae4Sat

$\phi \notin 3SAT$
$\Rightarrow$For any truth assignment $S$ of $\phi$, $\exists i$ such that $C_i$ evaluates to **false**
$\Rightarrow$For any truth assignment $S$ of $\phi$ along with truth assignments for the dummy variables, we need to exhibit an $i$ such that either $\tilde{C}_i$ or $\tilde{C}_i{}'$ evaluates to **false**.

Any truth assignment for dummy variables must set $\tilde{l_i}$ to either **true** or **false** Consider any truth assignment $S$ of $\phi$, We know that $\exists i$ such that $C_i$ evaluates to **false**. For such an $i$, consider the following cases

1. If $\tilde{l_i}$ is set to **true**, $\tilde{C_i}$ evaluates to **True**, but then $\tilde{C_i}'$ evaluates to **False**
2. Similarly, If $\tilde{l_i}$ is set to **false**, $\tilde{C_i}'$ evaluates to **True**, but then $\tilde{C_i}$ evaluates to **False**

$\Rightarrow$We have shown that for all truth assignments of $\phi'$, we have shown an $i$ such that $\tilde{C_i}$ or $\tilde{C_i}'$ evaluates to **false**
$\Rightarrow\phi'$ is unsatisfiable $\Rightarrow\phi' \notin$ NAE4SAT □

Thus $\phi \in 3SAT \Leftrightarrow \phi' \in$ NAE4SAT □

**Claim 2.** NAE4SAT $\leq_m^p$ NAE3SAT

Call the below condition as invariant

**Invariant 1.** *Clause does not have all its literals set to true*

*Proof.* The reduction is as follows

1. Let the NAE4SAT instance be denoted by $\phi = (C_1, \ldots, C_m)$, where $C_i$ are clauses and $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3} \vee l_{i,4}$
2. Output is $\psi = (C_1', C_1'', C_2', C_2'', \ldots, C_m', C_m'')$ where $C_i' = l_{i,1} \vee l_{i,2} \vee x_i$ and $C_i'' = l_{i,3} \vee l_{i,4} \vee \neg x_i$ where $x_i$ is a dummy variable. This is an instance of NAE3SAT

$\phi \in$ NAE4SAT
$\Rightarrow$there exists an assignment such that, for all $i$, $C_i$ is true, and moreover all $l_{i,1}, l_{i,2}, l_{i,3}, l_{i,4}$ cannot be true since $C_i$ satisfies invariant1
$\Rightarrow$The following are the subcases

1. $l_{i,1} = 0$ $\Rightarrow$atleast one of $l_{i,2}, l_{i,3}, l_{i,4}$ must be true.
   (a) If only $l_{i,2} = 1$, then $C_i'$ satisfies invariant 1, hence set $x_i = 1$ to make $C_i''$ satisfied and $C_i''$ satisfies invariant 1
   (b) If exactly one of $l_{i,3}$ or $l_{i,4}$ is true, then automatically $C_i'$ and $C_i''$ both evaluate to true and satisfies invariant 1 for any value of $x_i$
   (c) If all 3 are true, then $C_i'$ satisfies invariant 1 and we can set $x_i = 0$ to make $C_i''$ true and $C_i''$ will satisfy the invariant 1
   (d) If only $l_{i,2} = 1$ and $l_{i,3} = 1$, then automatically, both $C_i'$ and $C_i''$ evaluate to true and they satisfy invariant 1

(e) If only $l_{i,2} = 1$ and $l_{i,4} = 1$, then automatically, both $C_i'$ and $C_i''$ evaluate to true and they satisfy invariant 1

(f) If only $l_{i,3} = 1$ and $l_{i,4} = 1$, then clause $C_i'$ evaluates to true and it satisfies invariant 1, hence we can set $x_i = 1$ to make $C_i''$ satisfy the invariant 1

$\Rightarrow \psi \in$ NAE3SAT

$\phi \notin$ NAE4SAT
$\Rightarrow$For all assignments, there exist an $i$, such that either $C_i$ is false or all literals in $C_i$ are true (in other words $C_i$ does not satisfy invariant 1)
$\Rightarrow$For all assignments, consider the $i$ such that either one of the following holds

1. $C_i$ is false. Then $C_i'$ and $C_i''$ cannot be both true simultaneously since $C_i'$ contains $x_i$ and $C_i''$ contains $\neg x_i$ $\Rightarrow$either $C_i'$ or $C_i''$ is false and thus $\psi$ is unsatisfiable and hence $\psi \notin$ NAE3SAT

2. $C_i$ does not satisfy invariant 1. which implies all its literals are true, Then if $x_i = 1$, then $C_i'$ does not satisfy the invariant and if $x_i = 0$ then $C_i''$ does not satisfy the invariant $\Rightarrow$for any value of $x_i$, one of $C_i'$ or $C_i''$ doesn't satisfy the invariant 1 $\Rightarrow \psi \notin$ NAE3SAT

$\Rightarrow \psi \notin$ NAE3SAT.
Therefore $\phi \in$ NAE4SAT $\Leftrightarrow$ $\psi \in$ NAE3SAT and thus NAE4SAT $\leq_m^p$ NAE3SAT
□

Hence, by transitivity of many-one polynomial time reductions,we can conclude that 3SAT $\leq_m^p$ NAE3SAT and since 3SAT is NP-complete, NAE3SAT is also NP-complete.

2. (15 points) (**Algorithm for SAT from its existence**) Suppose SAT $\in$ DTIME($n^c$) wherein, the constant $c$ is known, while a description of the polynomial time algorithm is not given (that is, we just know it exists). For $i \in \mathbb{N}$, let bin($i$) be its binary representation and $M_{\mathsf{bin}(i)}$ be a machine whose description is bin($i$). Consider the algorithm 1 which finds a satisfying assignment using this information.

   Observe that this algorithm runs in polynomial time. Clearly reason out why this algorithm will correctly find a satisfying assignment if and only if $\phi$ is satisfiable. Is this algorithm practical ? [*Hint: Use the decision-to-search reduction for* SAT !]

**Solution:** To prove : $\phi$ is satisfiable if and only if the above algorithm finds a satisfying assignment

*Proof.* If $\phi$ is unsatisfiable, then no turing machine whose language is SAT will output a satisfying assignment

**Algorithm 1:** Algorithm for satisfiability

**Result:** Output a satisfying assignment for $\phi$
**Input:** A formula $\phi$ of size $n$
**for** *i from* 1 *to* $n$ **do**
  **if** $\mathsf{bin}(i)$ *is a valid description of a Turing machine* **then**
    Run $M_{\mathsf{bin}(i)}$ on $\phi$ for time $O(n^{c+1}\log n)$ ;
    **if** *the machine output a satisfying assignment for* $\phi$ **then**
      Accept and halt ;
Reject;

$\Rightarrow\phi$ is unsatisfiable $\Rightarrow$Algorithm won't find a satisfying assignment

□

3. (5 points) (**Deterministic Oracles**)

   Is it true $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{NP}}$ always or is not known ? What about $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{NP}$ ? Give proper justification for your answers.

**Solution:** $\mathsf{NP} \subseteq \mathsf{P}^{\mathsf{NP}}$ is always true.

Let $L \in \mathsf{NP}$, the following is a turing machine $M^L$ using $L$ as an oracle and accepting $L$ in polynomial time

$M =$ "
On input $x$,

1. Query the oracle if $x \in L$.
2. accept if and only if oracle says yes

Correctness is trivial since the oracle always gives correct responses and the oracle is the same language which we want to accept and the machine runs in polynomial time

Thus $L(M^L) = L$, thus $L \in \mathsf{P}^{\mathsf{NP}}$.

It is not known if $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{NP}$ is true or not.

4. (5 points) (**Immerman-Szelepcsényi Theorem in general form**)

   For any $s(n) \geq \log n$ that is space constructible, show that $\mathsf{NSPACE}(s(n)) = \mathsf{coNSPACE}(s(n))$. [*Hint: For* $L \in \mathsf{NSPACE}(s(n))$*, start with* $L_{pad} = \{x\#1^{2^{s(n)}-n-1} \mid x \in L$ *where* $n = |x|\}$ *and use* $\mathsf{NL} = \mathsf{coNL}$ !]

**Solution:**

**Claim 3.** $L \in \mathsf{NSPACE}(s(n)) \Rightarrow L_{pad} \in \mathsf{NL}$

*Proof.* Since $L \in \mathsf{NSPACE}(s(n))$, there is a turing machine $N$ accepting $L$ using $s(n)$ space
Below is a description of turing machine $N_p$ which accepts $L_{pad}$
$N_p =$
“
On input $y = x\#0^{2^{s(n)}-n-1}$

1. Run $N$ on $x$ (note that we don't have to copy the input $x$, we can use the input tape as input tape for $N$ and simulate $N$ in work tape, which takes $s(n)$ space)
2. Accept if and only if $N$ accepts

“
Input is $y = x\#0^{2^{s(n)}-n-1}$ and $|y| = 2^{s(n)}$ and the machine $N_p$ uses $s(n)$ space which is $\log(2^{s(n)}$. Hence $L(N_p) \in NL$
We need to prove $L(N_p) = L_{pad}$
$y = x\#0^{2^{s(n)}-n-1} \in L_{pad}$
$\Leftrightarrow x \in L$
$\Leftrightarrow N$ accepts $x$
$\Leftrightarrow N_p$ accepts $y$
$\Leftrightarrow y \in L(N_p)$
Thus $L(N_p) = L_{pad}$ and hence $L_{pad} \in NL$ □

**Claim 4.** $L_{pad} \in co\mathsf{NL} \Rightarrow L \in co\mathsf{NSPACE}(s(n))$

*Proof.* instead of directly constructing $x_{pad}$ use a counter to denote how many zeroes, we have given the turing machine

$L_{pad} \in co\mathsf{NL}$
Let $N_c$ be the turing machine which accepts $\overline{L_{pad}}$. The below is a turing machine $N$ which accepts $L$
$N =$
“
On input $x$, Reserve $s(n)$ space in work tape for maintaining a counter (possible using $s(n)$ space since it is space constructible) and initialize it with the binary representation of $2^{s(n)-n}$

1. Start simulating $N_c$, whenever it asks for input, Check the following cases
   (a) If input head is still valid (i.e head points to $x_i$ for some $i$), use this as the symbol read by $N_c$ and make the head movement corresponding to the transition of $N_c$

(b) If the input head is invalid and if the counter value is $2^{s(n)} - n$, then return # as symbol read and if head moves right, decrement counter by one and don't change if head moves left

(c) Else, return 0 as the symbol to $N_c$ and decrement counter by one if the corresponding head movement is towards right and increment counter by one if corresponding head movement is towards left

(d) If counter value is 0, then return blank symbol as input symbol for $N_c$

2. Accept if and only if $N_c$ accepts

"

Note that this machine uses only $O(s(n))$ space

*Correctness.* The non trivial part is the steps in 1. We know that input to $N_c$ should be of the form $x\#0^{2^{s(n)}-n-1}$, i.e we know that after and if we can keep track of how many zeroes $N_c$ have read after #, we can easily identify whether the machine $N_c$ would have read 0 or $x_i$ for some $i$, This is the reason we are maintaining a counter and once input head goes past $x$, we will start using the counter to determine whether to return 0 or # or blank symbol. Thus $N_c$ will always get the correct input

$x \in L$
$\Rightarrow x\#0^{2^{s(n)}-n-1} \in L_{pad}$
$\Rightarrow N_c$ rejects $x\#0^{2^{s(n)}-n-1}$
$\Rightarrow N$ rejects $x$
$\Rightarrow x \notin L(N)$

$x \notin L$
$\Rightarrow x\#0^{2^{s(n)}-n-1} \notin L_{pad}$
$\Rightarrow N_c$ accepts $x\#0^{2^{s(n)}-n-1}$
$\Rightarrow N$ accepts $x$
$\Rightarrow x \in L(N)$
Thus $L(N) = \overline{L_{pad}}$ □

Hence $L(N) \in co\mathsf{NSPACE}(s(n))$ □

We know that $\mathsf{NL} = co\mathsf{NL}$ and combining this with claims 3 and 4, we get $\mathsf{NSPACE}(s(n)) = co\mathsf{NSPACE}(s(n))$ for any space constructible $s(n) \geq \log n$

5. (15 points) Show that $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$. That is, the set of all languages that can be accepted by a two-sided error randomized polynomial time algorithm is the same as those that can be accepted with an oracle access to a BPP language. Is $\mathsf{RP}^{\mathsf{RP}} = \mathsf{RP}$

? Argue.

**Solution:**

**Claim 5.** $\mathsf{BPP} \subseteq \mathsf{BPP}^{\mathsf{BPP}}$

*Proof.* suppose there exists a $\mathsf{BPP}$ machine $M$ which accepts a language $L \in \mathsf{BPP}$, then the same machine is also a $\mathsf{BPP}^{\mathsf{BPP}}$.

The following is a turing machine which uses $L$ as an $\mathsf{BPP}$ oracle
$M' =$ "
On input $x$

1. Query if $x \in L$
2. Accept if and only if oracle accepts

"

$x \in L$
$\Rightarrow Pr[\text{oracle says yes}] \geq 1 - \epsilon$
$\Rightarrow Pr[M' \text{ says yes}] \geq 1 - \epsilon$

$x \notin L$
$\Rightarrow Pr[\text{oracle says yes}] \leq \epsilon$
$\Rightarrow Pr[M' \text{ says yes}] \leq \epsilon$

Thus $L \in \mathsf{BPP}^{\mathsf{BPP}}$ □

**Claim 6.** $\mathsf{BPP}^{\mathsf{BPP}} \subseteq \mathsf{BPP}$

*Proof.* Let $L \in \mathsf{BPP}^{\mathsf{BPP}}$
$\Rightarrow$there is a turing machine $M$ using $L' \in \mathsf{BPP}$ as an oracle which accepts $L$ with error probability $\epsilon$ i.e
$x \in L \Rightarrow Pr[M \text{ says yes}] \geq 1 - \epsilon$
$x \notin L \Rightarrow Pr[M \text{ says yes}] \leq \epsilon$

Let $M'$ be the turing machine which accepts $L'$ with error probability $\epsilon'$

We need to come up with a turing machine $N$ which accepts $L$ with a certain error probability $\epsilon''$

The following is the description of $N$
$N =$ "
On input $x$,

1. Run $M$ on $x$ without the oracle
2. Whenever a query of the form $y \in L'$ is asked, Run $M'$ on $y$ and return yes if and only if $M'$ accepts $y$

3. Accept if and only if $M$ accepts $x$

"

*Correctness.* The number of times oracle can be queried is only polynomial in size of $x$. Hence $N$ runs in polynomial time
The error of $M'$ is bounded, Hence error of $N$ is also bounded. i.e
$x \in L$
$\Rightarrow$

$$Pr[\text{N accepts x}] \geq Pr[\text{all queries returned correct answer}] \quad (1)$$
$$\geq 1 - \epsilon'^{n^c} \text{for some fixed constant c} \quad (2)$$

$x \notin L$
$\Rightarrow$

$$Pr[\text{N accepts x}] \leq Pr[\text{some queries returned wrong answer}] \quad (3)$$
$$\leq \epsilon'^{n^c} \text{for same constant c} \quad (4)$$

Thus $N$ is a turing machine without oracle accepting $L$ with error probability of $\epsilon'' = \epsilon'^{n^c}$ for some fixed constant $c$ □

Thus $\mathsf{BPP}^{\mathsf{BPP}} \subseteq \mathsf{BPP}$ □