CS452 HW11

Yuting Chen

9.16

Prove that $TQBF \notin SPACE(n^{\frac{1}{3}})$.

Using the space hierarchy theorem:

If g is a space-constructible which means $1^n \to 1^{g(n)}$ can be computed in space $O(g(n) \cdot f(n)) = O(g(n))$.

then, $SPACE(f(n)) \subsetneq SPACE(g(n))$.

There exists a language L which is solvable in linear space, but it can't be solved by sub-linear space.

Assume $TQBF \in SPACE(n^{\frac{1}{3}})$.

TQBF is space complete, then L should be reduced to TQBF in log space.

So, $L \in SPACE((n^c)^{\frac{1}{3}} + \log(n))$

That's contradiction.

Thus, $TQBF \notin SPACE(n^{\frac{1}{3}})$.

9.19

For this problem, we can solve by an polynomial time that will be assigned for an nondeterministic polynomial time class through a nondeterministic Turing Machine.

Since, $USAT \leq_p SAT$

$\Rightarrow \phi \in USAT$ if and only if the formula having variables x, y.

$\phi(x) \wedge \phi(y) \wedge$ "$x \neq y$" is satisfiable.

Rewrite it, use $\wedge$, $\vee$, negation operators such that

$x \neq y \Leftrightarrow \vee_{i=1}^{m} x_i \neq y_i \Leftrightarrow \vee_{i=1}^{m} x_i((x_i \wedge \bar{y}_i) \vee (\bar{x}_i \wedge y_i))$.

Thus, $USAT \in P^{SAT}$.

9.20

As we know that an oracle A exists such that $P^A \neq NP^A$ and $L_A \notin P^A$.

For a given oracle A. L_A should be defined as $L_A = \{w : |w| = |x| \text{ for some } x \in A\}$.

Then L_A is in NP^A. So $L_A \notin coNP^A$ will have to proof. $\bar{L_A} \notin NP^A$ as same.

If a problem is solvable in polynomial time then it is assigned to NP class by a nondeterministic Turing Machine.

The oracle TM M as follow:

On input w:

1. Calculate $g(w)$ and put it on the oracle tape.
2. Test the oracle if $g(w) \in SAT$, then accept.
3. Otherwise, reject.

The argument indicates that if C is a class and S is C-complete, then $C \subseteq P^S$.

As $P^{SAT} = coP^{SAT}$

So, $coNP \subseteq P^{SAT}$.

Thus, an oracle C exists for which $NP^C \neq coNP^C$.

9.21

(a) An oracle problem SAT will be NP complete problem.

Noted $L \subseteq NP$ and $P^{A,k} \in NP$.

$\Rightarrow$ This implies that L and $P^{A,k} \subseteq NP$ as it can be reduced in polynomial time by SAT.

So, $NP \subseteq P^{SAT}$.

As we know $L \subseteq NP$, $P^{A,k} \in NP$.

This implies L and $P^{A,k} \subseteq NP$ it can reduced in Polynomial time by SAT.

$\neg L (\in NP \subseteq P^{SAT})$ that also can be reduced in polynomial SAT.

Assume $coNP = NP$. Then, the polynomial collapsed to either NP or coNP.

The union operation is done between NP and coNP.

$\Rightarrow NP \subseteq P^{SAT}$ and $coNP \subseteq P^{SAT}$.

Union is completed in the polynomial time.

Thus, $NP \cup coNP \subseteq P^{SAT,1}$.

(b) An oracle problem SAT will be NP complete problem.

Noted $L \subseteq NP$ and $P^{A,k} \in NP$.

$\Rightarrow$ This implies that L and $P^{A,k} \subseteq NP$ as it can be reduced in polynomial time by SAT.

So, $NP \subseteq P^{SAT}$.

As we know $L \subseteq NP$, $P^{A,k} \in NP$.

This implies L and $P^{A,k} \subseteq NP$ it can reduced in Polynomial time by SAT.

$\neg L (\in NP \leq P^{SAT})$ that also can be reduced in polynomial SAT.
So, for each and every NP complete problem there is coNP complete problem.
$\Rightarrow NP \leq P^{SAT}$ and $coNP \leq P^{SAT}$.
If $NP \neq coNP$.
So, union is not computed in the polynomial time.
Thus, $NP \cup coNP \nsubseteq P^{SAT,1}$.

10.19

Assume an encoding of SAT in which inputs can be a Boolean input variable X_i or value 0 or 1.
Assume every Boolean formula with same number of inputs $(x, 0, 1)$ has same size.
Let a formula ϕ with n variables $X_1, X_2 \cdots X_n$ be input.
It would have the same size as $1, X_2 \cdots X_n$.
Suppose that $SAT \in BPP$.
Assume there is a probabilistic polynomial time Turing Machine M that with probability at least $\frac{1}{2k}$ outputs. the correct answer on any input of size k.
The Turing Machine as follow:
On input ϕ:
1. Initialize ϕ' to be ϕ.
2. Let be 1 to n, $\phi' \leftarrow \phi'$ with X_k set to 0.
3. Run M with (ϕ', y), y is a new random string.
4. If M accept, keep X_k set to 0 in ϕ.
5. If M reject, change ϕ', set X_k to 1 not 0.
6. Check the current setting for n variables satisfies the ϕ.
7. If so, accept. Else, reject.

If ϕ not satisfiable, there will be no setting of the variables that will satisfy ϕ, so reject.
If ϕ satisfiable, the probability that a run of M in any particular iteration outputs the correct answer $\frac{1}{2k}$, k is the size of ϕ.
Since $\phi' \neq \phi$.(size), so probability of fail does not change.

Since ϕ is at least the number of variables $\frac{1}{2^k} < \frac{1}{2^n}$.
The probability that any of n runs of M make error is at most $\frac{n}{2^n} < \frac{1}{2}$.
Thus, it's a ramdomized that decides SAT with one-sides error. $\Rightarrow$ SAT $\in$ RP.