ITCS:CCT09 : Computational Complexity Theory Apr 16, 2009

Problem Set #1

Topic: Lectures 1-8 *Due on:* **Apr 26, 2009**

Problem 1

(Padding Arguments)

This problem is to understand a simple technique of *padding* to prove translation results in complexity theory. The idea is to consider for any language L,

$$L_{\mathsf{pad}} = \left\{ x.\#^{f(|x|)} | x \in L \right\}$$

where choice of the function $f(.)$ depends on the specific application in mind. We will do two variants of this.

1. Show that $\mathsf{EXP} \neq \mathsf{NEXP} \Rightarrow \mathsf{P} \neq \mathsf{NP}$. Use a similar argument to show that $\mathsf{P} = \mathsf{L} \Rightarrow \mathsf{EXP} = \mathsf{PSPACE}$.
2. Prove that $\mathsf{P} \neq \mathsf{DSPACE}(n)$.

Problem 2

(Tape Reduction)

We saw tape reduction in the class, where we showed how a multi-tape Turing machine running in time $t(n)$ and space $s(n)$ can be simulated by a 1-tape Turing machine in time $O(t(n)^2)$ and space $O(s(n))$. We will show two improvements in this problem.

1. Prove that if we are simulating on a 2-tape Turing machine, this can be done in $O(t(n) \log t(n))$ and space $s(n)$.
2. If we allow Turing machine to be non-deterministic, then we can do tape reduction without a time overhead. Show that a multi-tape non-deterministic Turing machine running in time $t(n)$ can be simulated by a 1-tape non-deterministic Turing machine in time $O(t(n))$.

Problem 3

(Sparse and Tally Sets)

1. A set A is called *sparse* if there is a polynomial p, such that $|\{x \in A : |x| = n\}| \leq p(n)$. A set A is called *tally set* if $A \subseteq \{1\}^*$.
   Prove that following are equivalent.

- Restricted to tally sets $\mathsf{NP} = \mathsf{P}$. That is all tally sets in NP are in P.
- Restricted to sparse sets $\mathsf{NP} = \mathsf{P}$. That is all sparse sets in NP are in P.
- $\mathsf{EXP} = \mathsf{NEXP}$.

Hence conclude that $\mathsf{EXP} \neq \mathsf{NEXP} \Rightarrow \mathsf{P} \neq \mathsf{NP}$ (part(a) of the previous problem).

2. A set A is P-*computably sparse* if it is sparse but in addition, $|\{x \in A : |x| = n\}|$ is polynomial time computable. Show that $A \notin \mathsf{NP} \setminus \mathsf{coNP}$.

Problem 4

(Alternation)

Given an ϵ-free context-free language L (as its grammer in Chomsky Normal form) the problem of determining membership of a string in L can be done in P. Prove this by giving an ALOG parsing algorithm.

Problem 5

(Skew Circuits)

A circuit (with all gates of fan-in 2) is said to be a *skew circuit* if all negation gates appear at the input every $\wedge$ gate has at least one input which is a variable or a negation gate. Circuit Value Problem(CVP) asks; given a Boolean circuit and an input to the circuit, check if the circuit evaluates to 1.

1. Show that CVP restricted to the case when the input is skew circuits (call this Skew-CVP) is in NL.

2. Show that Reachability problem reduces to Skew-CVP via many-one logspace reductions, to Skew-CVP. Hence conclude that Skew-CVP is complete for NL.

Problem 6

(Need not be turned in.)

This problem is aimed at testing your understanding of some of the definitions & ideas that we discussed in the class. These need not necessarily be turned in. But you are welcome to write them down too.

- In the proof of Immerman-Szlepsinyi Theorem, to compute N_k, the number of configurations reachable in at most k steps from given configuration α (page 3-2, of notes for lecture 3) why cant we use the following simple step instead of the inductive way of counting.

  for each k, to compute N_k, we generate each configuration, β one by one, and for each one non-deterministically verify whether β is reachable from α. If yes, increment the count.

- In lecture 4, we saw $\mathsf{NP}^{\mathsf{P}} \subseteq \mathsf{NP}$. This was argued by saying that the non-deterministic oracle machine can simulate the membership query to oracle language (which is in P) without actually asking the query itself. But now, why wouldn't the same strategy work for $\mathsf{NP}^{\mathsf{NP}} \subseteq \mathsf{NP}$ or $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{NP}$?

- Suppose $\mathsf{P} = \mathsf{NP}$. That is as sets of subsets of Σ^*, P and NP are the same. Why does this not imply that $\mathsf{P}^C = \mathsf{NP}^C$ for every C? Note that we do know that (as we stated in class) there are oracles C for which $\mathsf{P}^C \neq \mathsf{NP}^C$. If the above argument is correct, then we already have $\mathsf{P} \neq \mathsf{NP}$!!.