CS6014 : Advanced Theory of Computation — Karthik Abinav S

Problem Set #6

1. (10 points) Oracle Queries.
   (a) Argue that $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$. Why does not the same argument work for NP and give $\mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}$?
   (b) If $\mathsf{NP} = \mathsf{coNP}$, argue that $\mathsf{PH} = \Sigma_1^p$.

Solution: $\mathsf{P}^{\mathsf{P}} \subseteq P$
Let $L \in \mathsf{P}^{\mathsf{P}}$. There exists an oracle TM N such that, it runs in polynomial time and has access to a language in P as an oracle. This can be simulated by deterministic polynomial time machine M without the oracle as follows :

M runs on the input in the same way as N. Except whenever N makes query to the oracle to check if $x \in K$ where K is the oracle, M runs the machine corresponding to the oracle language on x and gets the answer. It then proceeds in the similar way as the machine N. Since the oracle is a language in P, the simulation of the query to oracle takes deterministic polynomial time. Hence , M runs in polynomial time.

$\mathsf{P} \subseteq \mathsf{P}^{\mathsf{P}}$
This containment is trivially true, since the machine accepting P need not use the oracle and hence belongs to P^{P}

From above two containments, $\mathsf{P}^{\mathsf{P}} = \mathsf{P}$

The above argument wont work for NP because the machine does a guess and tries to verify the guess in deterministic polynomial time.
Consider a machine M which has access to a oracle in NP. Using the oracle in NP it verifies the certificate in polynomial time. To simulate the same using a NP machine without an oracle access, it has to simulate the query access to NP in P time. And since, we dont know if this is possible, hence we cant conclude the forward containment in above argument.

2. (10 points) (a) Reading Assignment : Read the proof (Section 3.4, Theorem 3.21, Page 93) of the claim : $\mathsf{DTIME}(2^{O(s(n))} \subseteq ASPACE(s(n))$. Determine an upper bound on the number of children for any universal configuration in the alternating Turing machine produced in the construction.
   (b) Conclude that $AL = \mathsf{P}$. Show that all CFLs are in P by giving an alternating Turing machine running in log space for checking membership. (Assume that the CFL is given at the input in the CNF form.).

3. (10 points) Dene the language:

$$\textsc{ShortestPath} = (G, k, s, t)| \text{ the shortest path from } s \text{ to } t \text{ in } G \text{ has length } k$$

   (a) (5 points) Prove that ShortestPath is in NL.

   (b) (5 points) Prove that ShortestPath is in L if and only if L = NL.

4. (15 points) An undirected graph is bipartite if its nodes can be divided into two sets such that all edges go from a node in one set to a node in the other set. Show that a graph is bipartite if and only if it does not contain a cycle that contains an odd number of nodes. Let

$$\textsc{Bipartite} = \{G \mid G \text{ is a bipartite graph } \}$$

   Show that Bipartite is in NL. (Hint : Use Immerman-Szelepsinyi theorem !).

5. (25 points) We define the product of two $n \times n$ Boolean matrices A and B as another $n \times n$ Boolean matrix C such that $C_{ij} = \bigvee_{k=1}^{n}(A_{ik} \wedge B_{kj})$.

   (a) (5 points) Show that boolean matrix multiplication can be done in logarithmic space.

   (b) (5 points) Using repeated squaring, argue that A^p can be computed in space $O(\log n \log p)$.

   (c) (5 points) Show that if A is the adjacency matrix of a graph, then ($A^k_{ij} = 1$ if and only if there is a path of length at most k from the vertex i to vertex j and is 0 otherwise.

   (d) (5 points) Use the above to give an alternative proof that $\mathsf{NL} \subseteq \mathsf{DSPACE}(\log^2 n)$. We originally proved it using Savitch's theorem.