Miguel Alcón Doganoc
Computational Complexity
May 21, 2019

Homework 2

1 Exercise: Adaptive vs non-adaptive queries

1.1 $\mathbf{P}_{||}^{\mathbf{NP}} \supseteq \mathbf{P}_{\mathbf{log\ n}}^{\mathbf{NP}}$

Let M be a TM that uses at most $O(\log n)$ adaptive NP queries. For each query asked to M, there are two possible answers, and, since the queries are adaptive, for each of the answers there are two other queries to be asked. Hence, there are $2^{k \log n} = O(n^k)$ queries that can be possibly asked during the whole computation (for some constant k). M can be reproduced with a non-adaptive oracle machine, first computing the $O(n^k)$ queries, and then reconstructing the correct path to follow with the given answers.

1.2 $\mathbf{P}_{||}^{\mathbf{NP}} \subseteq \mathbf{P}_{\mathbf{log\ n}}^{\mathbf{NP}}$

Let L be a language decidable by polynomially many non-adaptive SAT queries. Consider the following algorithm:

1. Determine the number of correct answers to the non-adaptive queries using binary search in $O(\log n)$ NP queries, asking in each of them if at least k of the expressions in L have satisfying truth assignments. Notice that the binary search is applied changing the value of k in each query (depending on the answer of the previous question) until it reaches the precise one.
2. Ask if there exists k satisfying truth assignments for k of the expressions s.t. the oracle machine will end up accepting if all other expressions are unsatisfiable.

With this, we proved that L can also be decided with logarithmically many adaptive NP queries.

2 Exercise: SAT, BPP, and RP

As it is said in the statement, we start from $SAT \in BPP$. Let M be a PTM that runs in polynomial time and solves SAT instances (of length n) with error at most $2^{-\Omega(n)}$. With it, we can define a PTM M' that runs in polynomial time, which on input X accepts with probability $\frac{1}{2}$ if X is satisfiable, and accepts with probability 0 if X is unsatisfiable.

Definition of M':

Given input $X = \{x_1, x_2, ..., x_n\}$
Let $B = \{b_1, b_2, ..., b_n\}$ be a boolean array
If $M(X) = 0$ **then**
    Output 0 and halt
For $i = 1, 2, ..., n$ **do**
    If $M(\{b_0, ..., b_{i-1}, 0, x_{i+1}, ..., x_n\}) = 0$ **then**
        Set $b_i = 0$
    Else if $M(\{b_0, ..., b_{i-1}, 1, x_{i+1}, ..., x_n\}) = 0$ **then**
        Set $b_i = 1$
    Else output 0 and halt
Check if B satisfies X
If it does, output 1 and halt, **else** output 0 and halt

M' runs in polynomial time because it makes at most $1 + 2n = O(n)$ calls to M, and each of this calls are done on input of length at most n. Since M' needs a satisfiable assignment to end with acceptance (output 1), the probability of accepting an unfeasible assignment is 0. Also, the probability of rejecting (output 0) a satisfiable input is $(1 + 2n) \times 2^{-\Omega(n)}$, which with a sufficiently big n is smaller than $\frac{1}{2}$. So $SAT \in BPP \Rightarrow SAT \in RP$.