CS 151 Complexity Theory	**Spring 2004**

Problem Set 2

Out: April 8 — *Due:* **April 15**

Reminder: you are encouraged to work in groups of two or three; however you must turn in your own write-up and note with whom you worked. You may consult the course notes and optional text (Papadimitriou). Please attempt all problems.

1. A *strong* nondeterministic Turing Machine has, in addition to its q_{accept} and q_{reject} states, a special state $q_?$. Such a Turing Machine accepts its input if all computation paths lead to q_{accept} and $q_?$ states, and it rejects its input if all computation paths lead to q_{reject} and $q_?$ states. Moreover, on every input, there is at least one computation path leading to q_{accept} or q_{reject}. Show that the class of languages accepted by a strong nondeterministic Turing Machine in polynomial time is exactly $\mathbf{NP} \cap \mathbf{coNP}$.

2. In this problem you will prove Mahaney's Theorem: a sparse language S cannot be **NP**-complete unless $\mathbf{P} = \mathbf{NP}$. Throughout this problem, S is a sparse language in **NP** with a polynomial bound $p(n)$ on the number of strings of length at most n.

   (a) Explain briefly where the proof of the special case of Mahaney's Theorem for *unary* languages (from class) breaks down for sparse languages.

   (b) Show that if $\overline{\text{SAT}}$ reduces to S in polynomial time via reduction R, then a procedure very similar to the one for unary languages from class decides $\overline{\text{SAT}}$ in polynomial time, and hence implies $\mathbf{P} = \mathbf{NP}$.

   (c) Define $c(n)$ to be the exact number of strings of length at most n in S (clearly $c(n) \leq p(n)$ for all n). Argue that the following language is in **NP**:

   $$\hat{S} = \{(x, 1^k) : k < c(|x|) \text{ or } (k = c(|x|) \text{ and } x \notin S)\}.$$

   Hint: do not try to compute $c(|x|)$; rather, focus on describing an **NP** algorithm that decides $\hat{S}$ properly under the assumption that $k = c(|x|)$, and then see what your algorithm does when $k \neq c(|x|)$.

   (d) Finally we assume S is **NP**-complete. Thus, everything in **NP** reduces to S, and we give names to two of these reductions: let T be a polynomial-time reduction from SAT to S, and let U be a polynomial-time reduction from $\hat{S}$ to S. Using T and U, describe a *family* of "candidate reductions from $\overline{\text{SAT}}$ to S," R_k, with the following properties:

   $$\begin{aligned} R_k(\phi) \in S & \qquad \text{if } k < c(|T(\phi)|) \\ R_k(\phi) \in S \Leftrightarrow \phi \in \overline{\text{SAT}} & \qquad \text{if } k = c(|T(\phi)|) \\ R_k(\phi) \notin S & \qquad \text{if } k > c(|T(\phi)|) \end{aligned}$$

(e) Using parts (b) and (d), prove Mahaney's Theorem. You may need to modify part (b) slightly so that on a given input ϕ, the procedure only applies R to formulae ϕ' for which $|\phi'| = |\phi|$. This should require at most a syntactic change: we can think of any partial assignment of values to variables in ϕ as having the same length as ϕ if we don't perform any simplification.

3. A directed graph $G = (V, E)$ is *strongly connected* if for every pair of vertices (x, y) there is a directed path from x to y and a directed path from y to x. Consider STRONGLY CONNECTED, the language of graphs G that are strongly connected. Either show that this problem is in $\mathbf{L}$, or prove a complexity consequence of such a containment.