CS 351: Solutions to Problem Set 2

(1) Consider the 3-COLORABILITY problem that is known to be NP-complete. The proof checking problem associated with 3-COLORABILITY is the following: Given a graph $G = (V, E)$ and a mapping $\chi : V \longrightarrow \{r, g, b\}$, verify that χ is a legal coloring of the vertices of G, that is, for all $(u, v) \in E$, $\chi(u) \neq \chi(v)$. When G is presented on a read-only, two-way input tape (say, as a list of edges) and χ is presented on a read-only, one-way proof tape (say, as a sequence of $|V|$ colors, one per vertex), it is possible to show that there is no algorithm using logarithmic workspace that correctly performs the verification task. Therefore, NP $\neq$ NL.

Where is the flaw in this reasoning?

The above argument only says that a *specific* proof system for 3-COLORABILITY, namely the relation LEGAL-3-COLORINGS $= \{\langle G, \chi\rangle \mid \chi \text{ is a legal 3-coloring of } G\}$, does not admit efficient verification (with a logspace verifier that has one-way access to the proof). It does not preclude the possibility that there is some other binary relation R' that is also a (complete and sound) proof system for 3-COLORABILITY and that admits one-way logspace verification. Thus, to show NP $\neq$ NL, we must prove that no proof system whatsoever for 3-COLORABILITY admits one-way logspace verification.

For extra credit, show the impossibility of the aforementioned proof verification task.

I intend to make this a homework problem in the near future (with copious hints), so I'll refrain from posting its solution now.

(2) *Equivalence between search and decision.* Suppose we have a polynomial-time algorithm A that solves the Boolean satisfiability problem, that is, the decision problem SAT. Construct a polynomial-time algorithm (using A as a sub-routine) that takes a Boolean formula φ as input and produces a satisfying assignment for φ, if one exists, or reports that none exists. How many calls to A does your algorithm make?

Let $x_1, \ldots, x_n$ denote the variables of φ. Run A on φ. If $A(\varphi) =$ "no," report that no satisfying assignment exists; if $A(\varphi) =$ "yes," then run A on two more formulas $\varphi[0]$ and $\varphi[1]$, where $\varphi[b]$ is the formula obtained from φ by setting $x_1 = b$. If φ is satisfiable, then one of these two formulas must be satisfiable, and A will tell us which one. Inductively, assume that we have a partial assignment a for the first k variables, $0 \leq k < n$, such that $\varphi[a]$ is satisfiable. Here $\varphi[a]$ denotes the simplification of the formula φ after setting $x_1 = a_1, \ldots, x_k = a_k$, and deleting all satisfied clauses. Make two queries, $\varphi[a0]$ and $\varphi[a1]$, each of which is a formula with at most $n - (k + 1)$ variables, to the sub-routine A. Since $\varphi[a]$ is satisfiable, one of these must be; if $\varphi[ab]$ is satisfiable, then replace a by ab, increment k, and continue. When $k = n$, we have an assignment a that satisfies φ. This procedure makes at most $2n + 1$ calls to A.

(3) Let R be a binary relation, and let S_R denote the search problem for R and let E_R denote the corresponding existence problem. Assuming that E_R is NP-complete, show that there is a polynomial-time reduction from S_R to E_R.

The idea is to mimic the satisfiability example. Let $c > 0$ be a constant such that for inputs of length n, the proof length and verification time for E_R are bounded by n^c. Define the relation R' by $R' = \{((x, a), y) \mid (x, y) \in R, |a| \leq |x|^c, \text{ and } a \sqsubseteq y\}$, where $a \sqsubseteq y$ denotes that a is a prefix of y. The existence problem $E_{R'}$ is defined, as usual, by $\{(x, a) \mid (\exists y)((x, a), y) \in R'\}$. Since $E_R \in \text{NP}$, it is easy to see that $E_{R'} \in \text{NP}$

We will describe a polynomial-time algorithm for S_R, assuming that we can solve $E_{R'}$. Since $E_{R'} \in \text{NP}$ and since E_R is NP-hard, we can replace queries to $E_{R'}$ by queries to E_R.

Suppose we are given an instance x of S_R. Let ε denote the empty string. If $(x, \varepsilon) \notin E_R$, then report that there is no y satisfying $(x, y) \in R$. Otherwise, we know that there is a y such that $|y| \leq |x|^c$ and $(x, y) \in R$, and we will find the lexicographically first such y by the following procedure.

> *set* $a \leftarrow \varepsilon$*;*
> *repeat the following for* $|x|^c + 1$ *steps:*
>   *if* $(x, a) \in R$*, then output* a*.*
>   *otherwise, if* $(x, a0) \in E_{R'}$ *then set* $a \leftarrow a0$*, else set* $a \leftarrow a1$*.*