CIS 511 Homework 7

Stephen Phillips, Dagaen Golomb

April 15, 2015

Problem 1

Show that $STRONGLY-CONNECTED = \{\langle G\rangle \mid G \text{ is a graph that is strongly connected}\}$ is NL-complete

Since $NL = \text{co-}NL$, and we know that $\overline{STRONGLY-CONNECTED}$ is in NL, then $STRONGLY-CONNECTED$ is in co-NL, hence NL.

Problem 2

Show that $2SAT$ is NL-complete.

We show that $\overline{2SAT}$ is in co-NL, which means that since $NL = \text{co-}NL$ that $2SAT$ is in NL. Given an instance of $2SAT$ in CNF form, we can interpret the clauses as implications: $(a+b) \iff (\bar{a} \implies b)$. We can 'derive' one implication from another by using : $(u \implies v)(v \implies w) \iff (u \implies w)$. Since there are only two variables per clause, if the formula is unsatisfiable, then we must be able to derive both $(x \implies \bar{x})$ and $(\bar{x} \implies x)$, since if there was a chaine of implications that led to a contradiction then we could reduce the chain using derivations to get to these implications. This in turn leads to:

$$(x \implies \bar{x})(\bar{x} \implies x) \implies (\bar{x}+\bar{x})(x+x) \implies \bar{x}x \implies 0$$

We can do this by non-deterministically following the implications that derive $(x \implies \bar{x})$ then again non-deterministically choose the ones that derive $(\bar{x} \implies x)$. If we find this line of implications, which we only need to store $O(\log n)$ bits for the intermediate implication we are using to derive the above.

We know that $\overline{PATH}$ is co-NL-complete, hence NL-complete. Therefore we want to show $\overline{PATH} \leq_L 2SAT$. To do this we consider given a graph G we create the graph $G^{\mathcal{R}}$, the graph with all the edges reversed. We know that there is a path from s to t in G if and only if there is a path from t to s in $G^{\mathcal{R}}$. We use this in the reduction.

We do this by, given an instance of $\overline{PATH}$, meaning a graph $\langle G, s, t\rangle$, we create a boolean formula ϕ from the edges. So $\forall (u,v) \in E$, we create clause $(u \implies v) = (\bar{u}+v)$ and clause $(v' \implies u') = (\bar{v'}+u')$ (corresponding to $G^{\mathcal{R}}$, different variables), then and all these clauses together. Except replace the variables for the nodes s and t that from the instance with literals x and $\bar{x}$ respectively, and also replace the variables s' and t' with x and $\bar{x}$, respectively. This can be done in log space since we only need to keep track of the two variables in the clause we are currently reading, a constant amount of space.

Why does this mapping work? If there is a path from s to t, then we can derive $x \implies \bar{x}$ by following the implications on the path from s to t. Similarly on the reversed graph we can follow the path from t to s to derive $\bar{x} \implies x$. Hence we can always derive both $x \implies \bar{x}$ and $\bar{x} \implies x$ when there is a path from s to t. If this is the case then we can derive our unsatisfiablility when we showed $2SAT$ was in NL. Hence if there is a path, this formula is never satisfiable. If there is not a path from s to t, the formula is always satisfiable. We examine several cases:

1. t has a path to s but not the other way around. That means in the G part of the formula we have a clause with $\bar{x} \implies x$, so we set x to true, and all variables that t can reach to true. For the variables corresponding to $G^{\mathcal{R}}$ we set to false since the implication $x \implies \bar{x}$ is in that set of clauses. Since we can vary the variables independently we can set both to true, and we can therefore set all the clauses to true.

2. s and t have no paths to each other at all. Then we are dealing with separate connected components, and similarly to last case we can just set x to a value and set the implications in the chain appropriately.

So this instance of $2SAT$ is satisfiable if and only if there is no path from s to t in G.

Problem 3

Give an example of an NL-complete context free language.

Problem 4

Define $pad : \Sigma^* \times \mathcal{N} \to \Sigma^* \#^*$ as $pad(s, l) = s\#^l$. Define the language $pad(A, f)$ for language A and function $f : \mathcal{N} \to \mathcal{N}$ as

$$pad(A, f) = \{pad(s, f(|s|)) \mid s \in A\}$$

Show that if $A \in \text{TIME}(n^6)$ then $pad(A, n^2) \in \text{TIME}(n^3)$

Is this not just that padding extends the length of the string? Now if you run the TM for A on the first part of the string, it will run in time $O(|s|^6)$, and since the length of our input is $n = |s|^2 + |s| = O(|s|^2)$ this becomes $O(|s|^6) = O((|s|^2)^3) = O(n^3)$ (?)

Problem 5

Prove using pad from previous problem that if $NEXPTIME \neq EXPTIME$ then $P \neq NP$.

Problem 6

Show that for an n variable polynomial P, with degree at most d, and total degree of t. We showed in class that if you pick $r_1, \ldots, r_n$ uniformly and independently at random in a set S then

$$\mathbf{Pr}[P(r_1, r_2, \ldots, r_n) = 0] \leq \frac{nd}{|S|}$$

We now want to strengthen this result to:

$$\mathbf{Pr}[P(r_1, r_2, \ldots, r_n) = 0] \leq \frac{t}{|S|}$$

Problem 7

Show if $NP \subseteq BPP$, then $NP = RP$

Problem 8

Define a ZPP-machine as a probabilistic Turing Machine that can output 3 things: accept, reject, and ?. A ZPP-machine M decided a langauge A if for every $x \in L$ it accepts with probability at least $2/3$, and rejects with probability 0, for $x \notin L$ it rejects with probability $2/3$ and accepts with probability 0, and it outputs ? on any input with probability at most $1/3$.