CIS 511 Homework 7

Stephen Phillips, Dagaen Golomb

April 29, 2015

Problem 1

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

Note that $STRONGLY-CONNECTED \in NL$. To see this, recall that $PATH \in NL$. To test for $STRONGLY-CONNECTED$, we iterate through each s-t node pair and run the $PATH$ algorithm on it. We only need to keep a counter for which s and t we are on, each logarithmic in the input size, and run the $PATH$ algorithm which can be done in log space. Thus, we can check for $STRONGLY-CONNECTED$ in log space and $STRONGLY-CONNECTED \in NL$.

Now we will reduce $PATH$ to $STRONGLY-CONNECTED$. Since $PATH \in NL-COMPLETE$, this will show $STRONGLY-CONNECTED$ is as well.

The reduction is as follows: given (G, s, t) as input to the $PATH$ problem, we add a directed edge from every node to s, and one from t to every node. Now, this new graph G' is strongly connected if a path from s to t exists. To see this, that for any node we can take the edge to s, then the path to t (assuming it exists), and then the edge from t to the destination node. Therefore, if $(G, s, t) \in PATH$ then $G' \in STRONGLY-CONNECTED$.

In the other direction, let G' be strongly connected. Then the original graph must have had a path from s to t. Had it not, then G' won't have one either (this is apparent since we have not added any paths except those to s and from t, so we could not have inadvertently added an s-t path). But this contradicts that G' is strongly connected. Therefore the original graph must have had an s-t path.

Since $(G, s, t) \in PATH$ iff $G' \in STRONGLY-CONNECTED$, $STRONGLY-CONNECTED \in NL-COMPLETE$.

Problem 2

Show that $2SAT$ is NL-complete.

We show that $\overline{2SAT}$ is in co-NL, which means that since $NL =$ 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 chain 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 unsatisfiability 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 $pad(A, n^2) \in \text{TIME}(n^3)$

Let M be the machine that decides A in time $O(n^6)$, and note $f(n) = n^2$ for the given problem. Construct a new TM M' which does the following:

On input x, check if the input is of the form $pad(y, |y|^2)$ for some string y. Note that from our construction, the string will be a perfect square in length with the first sqrt(n) characters in A and the rest the special symbol. One can successively square an incremented counter until the answer is greater than or equal to n. Then check that x matches the required format. Copying and comparing integers are linear operations, and multiplying is quadratic time, and this needs to be done at most a linear number of times (even if each square only incremented by 1 each time, which is clearly conservative). Thus, this can be done in O(n^3) time or less. If it is not in the correct form, reject.

If x is in the correct form, simulate M on y and echo its answer. This takes time $O(|y|^6) = O((|y|^2)^3) = O(n^3)$.

Since each part can be done in cubic time, this whole process can be done in $O(n^3)$ time.

Problem 5

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

To do this we show the converse, that $P = NP \implies NEXPTIME = EXPTIME$. Given a language $L \in NEXPTIME$, we know that there is a non-deterministic Turing machine M that runs in $2^{O(f(n))}$ where $f(n)$ is a polynomial, bounded by say cn^k. Now we construct the language

$$pad(L, 2^{cn^k}) = \left\{ x\#^{2^{c|x|^k}} \mid x \in L \right\}$$

as the definition of *pad* above. This language is in NP, since we construct a machine N that first checks that the string is in the right form of $x\#^{2^{c|x|^k}}$, then simulates M on x, which runs in $O(2^{c|x|^k})$ time, which is linear in the size of the input (due to the padding). Since this runs in non-deterministic polynomial time in the size of the input, by our assumption $P = NP$, we know that there exists a deterministic polynomial time algorithm that accepts $pad(L, 2^{cn^k})$, let us denote it N'.

We construct a machine A that accepts L in deterministic exponential time as follows. First we generate the string $y = x\#^{2^{c|x|^k}}$ in exponential time. Then we run $N'(y)$, which runs in polynomial time in the length of the input. We return whatever N' returns. This is deterministic since N' is deterministic. It is still in exponential time. Let us say N' has time complexity bounded by $c'n^{k'}$, then the runtime of this machine A is $c'\left(2^{c|x|^k} + |x|\right)^{k'} = O(2^{ck'|x|^k})$ which is still exponential. Therefore, we have constructed a deterministic machine that accepts in exponential time L for any L in $NEXPTIME$.

This technically only shows that $P = NP \implies NEXPTIME \subseteq EXPTIME$, but since $EXPTIME \subseteq NEXPTIME$ by definition, this is sufficient.

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:

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

As in class, the proof will be over induction on the number of variables in the polynomial. Like before, the base case of one variable is easy, as a monomial with [total] degree t can have at most t roots. Now assume this holds for $n - 1$ variables. For a polynomial of n variables, rewrite it like so:

$$P(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d} x_1^i P_i(x_2, \ldots, x_n)$$

Using the largest such i such that P_i is not identically zero, we have that the degree of P_i is less than or equal to $t - i$.

Now we randomly assign values as we did in class. Now,

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

(by induction hypothesis) and given the above is non-zero,

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

The probability of $P(r_1, \ldots, r_n) = 0$ is the addition of of its probability when $P_i(r_2, \ldots, r_n) = 0$ and $P_i(r_2, \ldots, r_n) \neq 0$, so the total probability is

$$\leq \frac{t-i}{|S|} + \frac{i}{|S|} = \frac{t}{|S|}$$

Problem 7

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

Let $NP \subseteq BPP$. We know $RP \subseteq NP$ (via using non-determinism for the coin tosses) so we only need to show $NP \subseteq RP$ under our given assumption. We can do this by showing an NP-Complete problem is in RP. We'll use the classic 3SAT.

Since $3SAT \in NP$ and given our assumption, we have a BPP algorithm for 3SAT. Let the machine for this be M. Let the Boolean function be ϕ. Now we construct a machine M' to decide 3SAT: Run M on ϕ. If it rejects, reject. If it accepts, try setting the first variable to true and run M on the modified formula. If this returns false, set the variable to false and repeat. If it returns true, move on to the next variable and probe its value. Repeat this until an assignment for all variables is reached. Then check if it is a correct assignment and accept/reject as appropriate.

It is clear that if ϕ is unsatisfiable that M' will never accept, since no complete assignment will cause it to accept and we only accept after a full assignment is found and verified.

If ϕ is satisfiable and M gives us correct answers at each step, we will reach a satisfying assignment and accept. Using the amplification lemma (10.5 in book), we can make any one decision have an error probability of at most 2^{-k}. We do this for n variables, so we have error probability $n2^{-k}$. We can make this less than 1/2 via our selection of k (the book shows how to chose such k within polynomial time).

Thus, $3SAT \in RP$ so $NP \subseteq RP$.

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 language 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. Let ZPP be the set of all languages that decided by ZPP machines. Formally, a language $L \in ZPP$ if there is a probabilistic turing machine that satisfies

$$\forall x \in L, \mathbf{Pr}[Z(x) = 1] \geq 2/3,\ \mathbf{Pr}[Z(x) = 0] = 0,\ \mathbf{Pr}[Z(x) = ?] \leq 1/3$$

$$\forall x \notin L, \mathbf{Pr}[Z(x) = 0] \geq 2/3,\ \mathbf{Pr}[Z(x) = 1] = 0,\ \mathbf{Pr}[Z(x) = ?] \leq 1/3$$

Show that $ZPP = RP \cap \text{co-}RP$.

First we show that $ZPP \supseteq RP \cap \text{co-}RP$. If a language L is in $RP \cap \text{co-}RP$, then there exists two machines R'_1 and R'_2 such that

$$\forall x \in L, \mathbf{Pr}[R'_1(x) = 1] \geq 1/2, \quad \mathbf{Pr}[R'_2(x) = 1] = 1$$

and

$$\forall x \notin L, \mathbf{Pr}[R'_1(x) = 0] = 1, \quad \mathbf{Pr}[R'_2(x) = 0] \geq 1/2$$

To make the analysis simpler, we use the fact that we can enhance the probabilities by running them several times, so we can create new machines R_1 and R_2 such that:

$$\forall x \in L, \mathbf{Pr}[R'_1(x) = 1] \geq 2/3, \quad \mathbf{Pr}[R'_2(x) = 1] = 1$$

and

$$\forall x \notin L, \mathbf{Pr}[R'_1(x) = 0] = 1, \quad \mathbf{Pr}[R'_2(x) = 0] \geq 2/3$$

So now we can construct a ZPP machine that accepts this language as follows.

```
function $Z(x)$
    Let $y_1 = R_1(x)$
    Let $y_2 = R_2(x)$
    If $y_1 = y_2 = 0$, reject
    If $y_1 = y_2 = 1$, accept
    Output ?
end function
```

Let us analyze this, using that the randomness of R_1 and R_2 are independent. For $x \in L$:

$$\begin{aligned}
\mathbf{Pr}[Z(x) = 1] &= \mathbf{Pr}[R_1(x) = 1,\ R_2(x) = 1] \\
&= \mathbf{Pr}[R_1(x) = 1]\mathbf{Pr}[R_2(x) = 1] \\
&\geq (2/3)(1) = 2/3 \\
\mathbf{Pr}[Z(x) = 0] &= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0]\mathbf{Pr}[R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0](0) = 0 \\
\mathbf{Pr}[Z(x) =?] &= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 1 \text{ or } R_1(x) = 1,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 1] + \mathbf{Pr}[R_1(x) = 1,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0]\mathbf{Pr}[R_2(x) = 1] + \mathbf{Pr}[R_1(x) = 1]\mathbf{Pr}[R_2(x) = 0] \\
&\leq (1/3)(1) + \mathbf{Pr}[R_1(x) = 1](0) = 1/3
\end{aligned}$$

So it satisifies all the conditions of ZPP in the case where $x \in L$. Now similarly for when $x \notin L$:

$$\begin{aligned}
\mathbf{Pr}[Z(x) = 0] &= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0]\mathbf{Pr}[R_2(x) = 0] \\
&\geq (1)(2/3) = 2/3 \\
\mathbf{Pr}[Z(x) = 1] &= \mathbf{Pr}[R_1(x) = 1,\ R_2(x) = 1] \\
&= \mathbf{Pr}[R_1(x) = 1]\mathbf{Pr}[R_2(x) = 1] \\
&= (0)\mathbf{Pr}[R_2(x) = 1] = 0 \\
\mathbf{Pr}[Z(x) =?] &= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 1 \text{ or } R_1(x) = 1,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0,\ R_2(x) = 1] + \mathbf{Pr}[R_1(x) = 1,\ R_2(x) = 0] \\
&= \mathbf{Pr}[R_1(x) = 0]\mathbf{Pr}[R_2(x) = 1] + \mathbf{Pr}[R_1(x) = 1]\mathbf{Pr}[R_2(x) = 0] \\
&\leq (1)(1/3) + (0)\mathbf{Pr}[R_2(x) = 0] = 1/3
\end{aligned}$$

So this also satisfies the conditions for ZPP, hence for any language in $RP\cap$ co-RP, we can construct a ZPP machine.

Now we show that $ZPP \subseteq RP \cap$ co-RP. First we show that $ZPP \subseteq RP$. For a language $L \in ZPP$, we have a machine Z that satisfies.

$$\forall x \in L, \mathbf{Pr}[Z(x) = 1] \geq 2/3,\ \mathbf{Pr}[Z(x) = 0] = 0,\ \mathbf{Pr}[Z(x) =?] \leq 1/3$$

$$\forall x \notin L, \mathbf{Pr}[Z(x) = 0] \geq 2/3,\ \mathbf{Pr}[Z(x) = 1] = 0,\ \mathbf{Pr}[Z(x) =?] \leq 1/3$$

For this we need to construct a machine, R_1, that satisfies

$$\forall x \in L, \mathbf{Pr}[R_1(x) = 1] \geq 1/2$$

$$\forall x \notin L, \mathbf{Pr}[R_1(x) = 0] = 1$$

We construct it as follows:

```
function $R_1(x)$
    Let $y = Z(x)$
    If $y = 1$, accept
    If $y = 0$ or $y =?$, reject
end function
```

The analysis fairly simple. For $x \in L$

$$\mathbf{Pr}[R_1(x) = 1] = \mathbf{Pr}[Z(x) = 1] \geq 2/3 \geq 1/2$$

And for $x \notin L$

$$\begin{aligned}\mathbf{Pr}[Z(x) = 0] &= \mathbf{Pr}[Z(x) = 0 \text{ or } Z(x) = ?] \\ &= 1 - \mathbf{Pr}[Z(x) = 1] \\ &= 1 - 0 = 1\end{aligned}$$

Now we show that $ZPP \subseteq RP$. So we construct another machine R_2 that satisfies

$$\forall x \in L, \mathbf{Pr}[R_2(x) = 1] = 1$$

$$\forall x \notin L, \mathbf{Pr}[R_2(x) = 0] \geq 1/2$$

Similarly to last time, we construct the following machine:

```
**function** $R_2(x)$
    Let $y = Z(x)$
    If $y = 0$, reject
    If $y = 1$ or $y = ?$, accept
**end function**
```

The analysis is similar to before. For $x \in L$

$$\begin{aligned}\mathbf{Pr}[Z(x) = 1] &= \mathbf{Pr}[Z(x) = 1 \text{ or } Z(x) = ?] \\ &= 1 - \mathbf{Pr}[Z(x) = 0] \\ &= 1 - 0 = 1\end{aligned}$$

And for $x \notin L$

$$\mathbf{Pr}[R_1(x) = 0] = \mathbf{Pr}[Z(x) = 0] \geq 2/3 \geq 1/2$$