CS500: Solutions for Final Exam

1. (10 points) Suppose L_1 and L_2 are regular languages recognized by DFAs M_1 and M_2, where both M_1 and M_2 have n states. Prove that if $L_1 \neq L_2$, there is a word w of length at most n^2 which distinguishes them (i.e., $w \in L_1$ but $w \notin L_2$ or vice versa).

   Answer. Let $M_1 = (\Sigma, Q_1, \delta_1, q_1^0, F_1)$ and $M_2 = (\Sigma, Q_2, \delta_2, q_2^0, F_2)$ (we assume that L_1 and L_2 are defined on the same alphabet Σ). We construct a machine $M = (\Sigma, Q, \delta, q^0, F)$ as follows:

   $$\begin{aligned} Q &= Q_1 \times Q_2 \\ \delta((q_1, q_2), a) &= (\delta_1(q_1, a), \delta_2(q_2, a)) \\ q^0 &= (q_1^0, q_2^0) \\ F &= (F_1 \times \overline{F}_2) \cup (\overline{F}_1 \times F_2) \end{aligned}$$

   Then clearly M recognizes

   $$L = L_1 \oplus L_2 = (L_1 \cap \overline{L}_2) \cup (\overline{L}_1 \cap L_2)$$

   (this is analogous to the constructions for $L_1 \cap L_2$ or $L_1 \cup L_2$ that show the regular languages are closed under intersection and union). Now $L_1 \neq L_2$ if and only if $L \neq \emptyset$, i.e. if M accepts some word. But if M accepts any word, it accepts a word of length at most $|Q| = n^2$ (in fact, $n^2 - 1$); any word longer than this has a loop by the pigenhole principle, and removing that loop gives a shorter word which is also accepted. □

   Exercise: find a family of pairs of languages L_1, L_2 with n-state DFAs such that the smallest word that distinguishes them has length $\Theta(n^2)$. Hint: such language pairs exist even for a single-symbol alphabet.

2. (20 points total) Consider the following problem.

   Input: a nondeterministic finite automaton $M = (\Sigma, Q, \delta, q_0, F)$
   Question: let L be the language accepted by M. Is $L = \emptyset$?

   (a) (10 points) This problem is complete for one of the complexity classes we discussed this semester. State which one, and prove it.

   (b) (10 points) If we replace the NFA by a DFA, the resulting problem is also complete for one of the complexity classes we discussed. State which one, and prove it. If you think it is complete for the *same* complexity class as the problem for NFAs, describe a restricted version which is complete for a *lower* complexity class.

Answer. Let us call this question NFA Emptiness. We show it is NL-complete by showing its complement, NFA Non-Emptiness, is NL-complete and using the fact that NL = coNL.

To show that NFA Non-Emptiness is in NL, notice that given a description of the NFA on its input tape, a nondeterministic logspace Turing machine can guess a word $w \in L$, and guess a series of transitions, and verify that it ends in an accepting state. At each step, it only needs to keep track of the current state of the machine, which takes $O(\log n)$ bits.

We now kill two birds with one stone by showing that DFA Non-Emptiness is NL-complete; this proves that the more general problem NFA Non-Emptiness is NL-complete, and also shows that this problem is just as hard for DFAs as for NFAs. We reduce from Reachability to DFA Non-Emptiness.

Given an instance (G, s, t) of Reachability where $G = (V, E)$ is a directed graph and $s, t \in V$, we construct a DFA whose states are the vertices of G plus a reject state, $Q = V \cup \{\text{reject}\}$, and which has one transition for each directed edge of G. To make this is a DFA, we give it a large enough alphabet Σ so that each outgoing edge from a given v can have its own symbol: one simple way to do this is to let $\Sigma = V$ and define

$$\delta(v, w) = \begin{cases} w & \text{if } (v, w) \in E \\ \text{reject} & \text{if } (v, w) \notin E \end{cases} .$$

We set the start state and accepting state to demand a path from s to t, i.e., $q_0 = s$ and $F = \{t\}$. Then a path exists from s to t if any only if L is nonempty; indeed, L is the set of paths from s to t. □

To restrict the problem further, recall that Reachability is L-complete in the special case where no vertex has out-degree greater than 1. (If this isn't clear, think about why Reachability was NL-complete in the first place.) However, this is a somewhat unnatural condition to impose on a DFA. A nicer condition is to stipulate that the DFA's alphabet consists of a single symbol, $\Sigma = \{a\}$. In that case no state can have more than one transition out of it. Since Reachability again corresponds to the nonemptiness of L, the problem Single-Symbol DFA Emptiness is co-L-complete.

Another way to remove nondeterminism is to ask if a DFA M accepts a specific word w, i.e., given M and w whether $w \in L(M)$. This problem, DFA Acceptance, is L-complete (even if the word in question is one long string of as!) □

Exercise: to prove that the DFA problem is also NL-complete, why is it not enough to recall that an NFA can be converted into a DFA? Hint: complexity is defined as a function of the size of the input.

Another exercise: prove that Reachability is still NL-complete, and Outdegree-1-Reachability is still L-complete, even if we restrict the graphs to be acyclic. Hint: add a counter to a logspace Turing machine.

3. (30 points total) Consider the following model of cellphone conversations. There is an undirected graph $G = (V, E)$, where the vertices are people, and each edge indicates that two people are within range of each other so that they can potentially have a conversation. However, while two people are talking, their neighbors must stay silent (on that frequency) to avoid interference. Thus a set of conversations consists of a set of edges $S \subset E$, where vertices in different edges in S cannot be neighbors of each other. Formally:

$$(u, v), (w, t) \in S \Rightarrow (u, w) \notin E \ .$$

The *cellphone capacity* of G, denoted $C(G)$, is the largest number of conversations that can take place simultaneously on one frequency, i.e., the size of the largest such set S. For instance, if G is a cube (with 8 vertices and 12 edges) its cellphone capacity is 2. Now consider the following problem:

CELLPHONE CAPACITY
Input: a graph G and an integer k
Question: is $C(G) \geq k$?

Prove that CELLPHONE CAPACITY is NP-complete. You may reduce from any problem proved to be NP-complete in Sipser, but I think the easiest is to use either 3-SAT or Vertex Cover. For 10 points extra credit, provide reductions from both of these.

Answer. First, to prove membership in NP, the set S is clearly a polynomial-time checkable certificate: just check that $|S| \geq k$, that $S \subset E$, and that the condition above holds (that no two edges in S have neighboring endpoints).

Reduction from 3-SAT. Given a 3-SAT formula n variables and m clauses, we convert it to a graph G using the gadgets shown in Figure 1. Each variable gadget has a central vertex, which can talk to one but not the other of its neighbors. The clause gadget consists of a triangle, each corner of which is connected to a variable gadget (one end or the other depending on whether the variable is negated) plus a central vertex connected to all three. To complete the reduction we define $k = n + m$.

To prove this reduction works, the key point is that exactly one of the edges in a clause gadget can be in S if one or more of the neighboring edges in its variable gadgets are not in S; if all three of the neighboring variable edges are in S, then none of the edges in the clause gadget can be. These two cases correspond to the clause being satisfied (because one or more of its variables agree with it) or dissatisfied (because all three of them disagree with it).

Now, suppose the 3-SAT formula is satisfiable. Choose a satisfying assignment. S consists of one edge for each variable and each clause: in each variable gadget we include the edge corresponding to its value, and in each clause gadget we include the edge from its central vertex to the vertex connected to one of the variables that satisfies that clause. None of these edges are neighboring, and $|S| = n + m = k$.

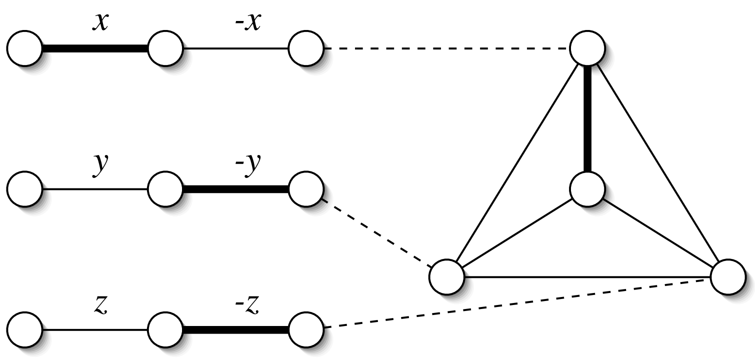

Figure 1: The choice and variable gadgets for reducing 3-SAT to Cellphone Capacity. Shown is a gadget for the clause $x \vee y \vee z$.

Conversely, suppose that $C(G) \geq k$. Note that none of the connecting edges between clause gadgets and variable gadgets (shown dashed in Figure 1) can be in S, since in that case none of the edges in either of those gadgets can be in S. Thus S consists of one edge in each clause gadget and each variable gadget. If we define a truth assignment according to which edge in each variable gadget is in S, then the observation above (and the fact that one edge in each clause gadget is in S) implies that this truth assignment satisfies the formula. $\square$

Reduction from Vertex Cover. Given an instance $\langle G = (V, E), k\rangle$ of Vertex Cover, we convert it to an instance $\langle G' = (V', E'), k'\rangle$ of Cellphone Capacity as follows. For each $v \in V$, define a new vertex v'; then let $V' = V \cup \{v' \mid v \in V\}$ and let $E' = E \cup \{(v, v') \mid v \in V\}$. In other words, we give each vertex v a partner v'; the edges of G' consist of the edges of G, plus a new edge between each vertex and its partner. Finally, we set $k' = |V| - k$.

To prove that this reduction works, recall that the complement of a vertex cover is an independent set. In that case, let S consist of the edges connecting the vertices *not* in the vertex cover to their partners; these edges do not have neighboring endpoints and thus will form a legal set of conversations. Thus if G has a vertex cover of size k or less, $C(G') \geq k'$.

Conversely, suppose that $C(G') \geq k'$. If we choose one endpoint from each edge in S, this gives an independent set of size at least k', and its complement is a vertex cover of size k or less. (Alternately, note that if two vertices in V are having a conversation, we can always change one of them to the partner of the other without reducing the size of S; therefore, we can assume without loss of generality that all edges in S are of the form (v, v'), and the set of endpoints in V forms an independent set.) $\square$

4. (20 points) Provide a reduction, as direct as possible, from Graph 3-Coloring to Positive 1-in-3 SAT. In other words, prove that if Positive 1-in-3 SAT is in P, then so is Graph 3-Coloring.

   Answer. Given an instance $G = (V, E)$ of Graph 3-Coloring we convert it to a Positive 1-in-3 SAT formula as follows. For each $v \in V$, define three variables r_v, g_v, b_v, and include the clause (r_v, g_v, b_v). We interpret these variables as giving the color of v, e.g. r_v is true if and only if v is colored red. For each $(u, v) \in E$, define three variables $r_{u,v}, g_{u,v}, b_{u,v}$ and include the three clauses

   $$(r_u, r_v, r_{u,v}) \wedge (g_u, g_v, g_{u,v}) \wedge (b_u, b_v, b_{u,v}) \ .$$

   Note that these clauses require that u and v be different colors; for instance, r_u or r_v may not both be true (and if both are false, we set $r_{u,v}$ to true).[1] This gives a formula consisting of $3|V|+3|E|$ variables and $|V|+3|E|$ clauses, and this reduction can clearly be carried out in polynomial time.

   If G is 3-colorable, choose a 3-coloring. Then for each v, set one of the r_v, g_v, b_v true and the other two false, according to v's color. For each $(u, v) \in E$, set one of the $r_{u,v}, g_{u,v}, b_{u,v}$ true (for whichever color neither u nor v has) and the other two false. This gives a satisfying assignment of the positive 1-in-3 SAT formula.

   Conversely, if the formula is satisfiable, assign colors to each vertex v according to which of r_v, g_v, b_v is true. Since the edge clauses are satisfied, the endpoints of every edge have different colors, so this gives a proper 3-coloring of G. $\square$

5. (10 points) A logspace Turing machine has two tapes, a read-only input tape of poly(n) size and a read/write work tape of $O(\log n)$ size. In the original definition, it has one head for each of these tapes. However, what happens if we allow the machine to have k heads on its input tape for some constant k? Prove that this does not make this class of machines any more powerful; i.e., any language accepted by a k-head logspace Turing machine is in L.

   Answer. We can describe this at many levels of detail, but the point is this. A 1-head TM M' can simulate a k-head TM M by keeping $k+1$ counters on its work tape, which record the positions of the k heads, and its own position, on the input. It simulates moving the k heads left or right by incrementing or decrementing these counters, and it moves to their positions, and reads the tape symbols there, by incrementing or decrementing its own counter as it moves and comparing its position to theirs.

   The key fact is then that since the input is of size n, all of these counters have values that range from 1 to n, which can be stored as $\log_2 n$-bit numbers. Thus the total space M' needs is $(k+1)\log_2 n = O(\log n)$, plus the $O(\log n)$ work space of M, for a total of $O(\log n)$. $\square$

[1] We can include the clause $(r_{u,v}, g_{u,v}, b_{u,v})$ as well if we like, but it isn't necessary.

6. (10 points) Prove that for any NAE-3-SAT formula there exists a truth assignment that satisfies at least 3/4 of its clauses. (Assume that no variable appears more than once in a single clause.)

   Answer. For a non-constructive solution, consider the set of 2^n truth assignments for an NAE-3-SAT formula with n variables and m clauses. Since any NAE-3-SAT clause is satisfied by 6 of the 8 possible assignments of its variables, and since the average of the sum is the sum of the averages, the average number of clauses satisfied by a random assignment is $(3/4)m$. Since the maximum of this set of 2^n numbers is at least its average, at least one truth assignment satisfies at least $(3/4)m$ clauses. □

   For a more algorithmic solution, start with an arbitrary truth assignment, and flip variables one at a time until we reach a local maximum of the number of satisfied clauses. That is, find a truth assignment such that flipping any variable changes at least as many satisfied clauses to dissatisfied ones as it does the reverse.

   Let x be a variable that appears in k dissatisfied clauses. Since flipping x will satisfy these k clauses, it must also appear in k satisfied clauses, which will be dissatisfied if we flip it. Say that these clauses "block" x from flipping; then note that each clause can only block one variable, namely the "odd one out" whose value (or negated value) is opposite to that of the other two literals. Then if there are ℓ dissatisfied clauses, we need a blocking clause for each of their 3ℓ variable occurrences. Thus there are at least 3ℓ satisfied clauses, so $\ell \leq m/4$ and at least $(3/4)m$ clauses are satisfied. □