CSE 200
Computability and Complexity
Homework 3
NP, Completeness, and Reductions
Part 1
Due Wed. May 28

May 30, 2008

Give proofs for each problem. Proofs can be high-level, but be precise. You may use without giving a proof any result proved in class or in the textbook. In particular, to prove NP-completeness, it suffices to give a reduction from any of the NP-complete problems from the text or from class. However, you must show your reduction is valid, by showing the equivalence of the constructed instance and the original.

SAT restriction Prove that the 3SAT problem remains NP-complete when restricted to formulas where each variable appears in at most 3 clauses. Remember that the input to 3SAT is a CNF formula with *at most* three variables per clause, so clauses of size 2 and 1 are also permissible.

Let $\Phi = C_1 \wedge C_2 \wedge C_m$ be a $3 - CNF$ formula in $x_1, ..x_n$. We define a new 3-CNF formula Ψ where each variable appears in at most 3 clauses as follows. If x_i appears (positive or negated) in clause C_j, we have a variable $x_{i,j}$. We define a clause C'_j to be C_j where we replace each variable x_i with the corresponding variable $x_{i,j}$. (For example, if $C_j = x_1 \vee x_2 \vee \neg x_3$, then $C'_j = x_{1,j} \vee x_{2,j} \vee \neg x_{3,j}$.) We then add the following clauses for each variable x_i : Let j_1, j_d be the indices of the clauses x_i appears in. The constraint $\neg x \vee y$ expresses $x \rightarrow y$; for clarity, we'll describe the clauses as implications. We add the d clauses $x_{i,j_1} \rightarrow x_{i,j_2}, x_{i,j_2} \rightarrow x_{i,j_3}, ...x_{i,j_d} \rightarrow x_{i,j_1}$.

Ψ is constructible from Φ in polynomial time in the straight-forward way. Each $x_{i,j}$ appears in clause C'_j and at most 2 implication clauses, so each variable appears at most 3 times.

Assume Φ is satisfiable, by assignment $x_i = a_i$, $i = 1..n$. Then we can assign each $x_{i,j} = a_i$. Since $a_i \to a_i$, all the last implication clauses are true. Let C'_j be a non-implication clause. Then C_j was satisfied by some literal it contains, either x_i or $\neg x_i$. Since we give $x_{i,j}$ the same value as x_i had, C'_j is satisfied by the corresponding literal under the new assignment. Thus, if Φ is satisfiable, Ψ is satisfiable.

Conversely, if $x_{i,j} = a'_{i,j}$ is an assignment that sasisfies Ψ, to satisfy the implication clauses we have $a'_{i,j_1} \to a'_{i,j_2} \to .... \to a'_{i,j_d} \to a'_{i,j_1}$. So for each i, all of the $a'_{i,j}$'s are identical. We assign x_i this value $a_i = a_{i,j_1} = ...a_{i,j_d}$. As before, if C'_j is satisfied by variable $x_{i,j} = a'_{i,j}$, C_j is satisfied by variable $x_i = a_i = a'_{i,j}$. So if Ψ is satisfiable, Φ is satisfiable.

So we have reduced an arbitrary 3-SAT formula into an equivalent restricted formula in polynomial time. Therefore, the problem remains NP-complete under the restriction.

neq SAT Problem 7.24 from the Sipser text, page 296. a. In a $\neq$-assignment, each clause has at least one true and at least one false literal. In the negation of such an assingment, the true literal becomes false and the false true. So there are still at least one true and one false literal in each clause, so the negation is also a satisfying assignment.

b. Let $\Phi = C_1 \wedge C_2 \wedge ...C_m$ be a 3-SAT formula and let Ψ be the set of $\neq$ clauses constructed in the text, with D_i, E_i be the first and second constructed clauses corresponding to C_i.

Assume $x_1 = a_1, ..x_n = a_n$ is a satisfying assigment for Φ. Consider the following assignment: $b = 0$, $x_i = a_i$ for original variables, and for each C_j, z_j is 1 if the first two literal y_1, y_2 of C_j are both 0, and 0 otherwise. Consider the clause D_j. If the first two literals are both 0, $z_j = 1$, so the clause has at least one true and one false literal. If not, at least one of the two are 1, and $z_j = 0$, so the clause has at least one true and one false literal. For the clause E_j, if the first two literals in C_j are false, then the third must be true (since a is a satisfying assignment) and z_j is set to true, so $\neg z_j = 0$. Thus E_j has one true (y_3) and one false z_j literal. If not, E_j has one true literal ($\neg z_j$) and one false literal b. In all cases, the new assignment is a $\neq$ assignment to each D_j and E_j, hence to Ψ.

In the other direction, assume Ψ has a $\neq$ assignment. By part a., there is such an assigment with $b = 0$. (Otherwise, negate all the variables in an assignment with $b = 1$.) Consider the assignment that gives all original variables their values under this assignment. Consider each clause C_j. Since $b = 0$, if y_3 is given value 0, z_j must be given value 0 to satisfy E_j. Then at least one of y_1 or y_2 must be given value 1 to satisfy D_j. Thus, C_j is satisfied either by y_3 or by one of y_1 or y_2. Thus, each C_j is satisfied, and the assignmnent satisfies Φ.

c. So from Φ we can construct a Ψ whose $\neq$ satisfiability is equivalent to Φ's satisfiability. This construction is poly-time. We can also, given an assignment to a formula Ψ, check that each clause has at least one 0 and one 1 in polynomial time. Thus $\neq$-SAT is in NP, and $3 - SAT$ reduces to it. Thus, it is NP-complete.

MAX-CUT is NP-complete MAX-CUT $\in NP$ since we can compute the value of a given cut in polytime by adding up the values of the edges that cross the cut. To show NP-hardness, we reduce from 3NAE-SAT. Let $F \in 3 - CNF$ have m clauses and n variables. For each variable x, construct a complete bipartite graph $G_x = (L_x, R_x.E_x)$ with $L_x = \{a_{x,C} \mid C \in F\}$, $R_x = \{a_{\bar{x},C} \mid C \in F\}$. (The hint says to make $|L_x| = 3m$, but that's overkill, m is enough as long as we assume that each clause has noncontradictory literals, which we might as well since otherwise the clause is automatically satisfied.) Let G be the union of these graphs. Now for clause $C = \{l_1, l_2, l_3\} \in F$, add to G a triangle of edges connecting nodes $a_{l_1,C}, a_{l_2,C}, a_{l_3,C}$. Set $k = m^2n+2m$. We claim that $(G, k) \in MAX - CUT \Leftrightarrow F \in 3NAE - SAT$.

($\Rightarrow$) Let S be a cut of G with value $\geq k$. At most 2 edges from each clause triangle cross the cut, so the number of cut edges from the G_x must be m^2n. But that's all of the edges in the G_x graphs. This also implies that *exactly* 2 edges from each clause triangle cross the cut. Since each G_x is complete bipartite, this implies that for each x, either all of L_x is in the cut and none of R_x in which case we assign x as true, or all of R_x is in the cut and none of L_x, in which case we assign x as false. Let $C \in F$. Since exactly 2 of the edges of the triangle t corresponding to C cross the cut, at least 1 node of t must be in S and at least 1 node must be in $\bar{S}$. The literals corresponding to these 2 nodes are assigned distinctly. So $F \in 3NAE - SAT$.

($\Leftarrow$) Let $a \in \mathrm{sol}_{\mathrm{NAE}}(\mathrm{F})$. Let S be the set of nodes of G corresponding to literals true under a. Then each G_x has either all of L_x in S and none of R_x or all of R_x in S but none of L_x. This contributes $m^2 n$ to the cut value. Each clause $C \in F$ has a literal l_1 with a truth value distinct from that of the other 2 literals l_2, l_3 at a. So the 2 edges $(a_{l_1,C}, a_{l_2,C}), (a_{l_1,C}, a_{l_3,C})$ are cut edges. So the clause triangles contribute another $2m$ to the cut value. So $(G,k) \in MAX - CUT$.

Turing reductions Prove that NP is closed under Turing reductions if and only if $NP = co - NP$.

We first show that $P^{NP \cap co-NP} = NP \cap co - NP$ without any assumptions. (This part by Chris Calabro).

Nb. we will use $(x \in A)$ to denote the truth value of whether $x \in A$.

Lemma $\mathrm{P}^{\mathrm{NP}\cap\mathrm{coNP}} = \mathrm{NP} \cap \mathrm{coNP}$.

($\subseteq$) Let M^A be a polytime oracle Turing machine where $A \in \mathrm{NP} \cap \mathrm{coNP}$, and let L be its language. Note that $\forall x \in \Sigma^*$, there is a polytime verifiable witness to whether $x \in A$ or not; i.e. there is a verifier V giving 1 of 3 outputs - $0, 1, \perp$ - and a polynomial p s.t. $\forall x \in \Sigma^* \; \exists y \in \Sigma^{p(|x|)} \; V(x,y) = (x \in A) \wedge \forall z \; (V(x,z) \neq \perp \rightarrow V(x,z) = (x \in A))$ and $V(x,z)$ runs in time $\leq p(|x|)$.

To show that $L \in \mathrm{NP}$, given input w, we simulate $M^A(w)$ and replace each oracle query x by a nondeterministic guess $y \in \Sigma^{p(|x|)}$ of a witness for whether or not $x \in A$ followed by running $V(x,y)$ and rejecting if $V(x,y) = \perp$ and returning $V(x,y)$ as the oracle answer otherwise. If $x \in L$, then each oracle query x has a witness y that causes $V(x,y)$ to correctly return $(x \in A)$ and so our simulation will accept. Conversely, if $x \notin L$, then on any run of the simulation, we either reject because some witness y to a query x causes $V(x,y) = \perp$ or else every query x is answered correctly, and we reject as $M^A(x)$ would. The simulation takes polytime since M and V take polytime.

To show that $L \in \mathrm{coNP}$ reduces to the above: $\overline{L} \in \mathrm{P}^{\mathrm{NP}\cap\mathrm{coNP}}$, so from what has already been shown, we know that $\overline{L} \in \mathrm{NP}$.

($\supseteq$) Obvious.

The claim follows from this lemma. If $NP = co - NP$, then $NP = NP \cap Co - NP$, so $P^N P = P^{NP \cap co-NP} = NP \cap co - NP = NP$, and

NP is closed under Turing reductions. Conversely, each language L reduces to $\overline{L}$, by making the input as query and negating the answer. So if NP is closed under Turing reductions, in particular, it is closed under compliment, and $co-NP \subset NP$. But then also $NP \subset co-NP$ (If $L \in NP, \overline{L} \in co-NP \subset NP$, so by definition of $co-NP$, $L \in co-NP$), so $NP = co-NP$.

1 Reduction from sudoku to SAT

Chris's solution:

Here are 2 reductions from $n^2 \times n^2$ sudoku to SAT. Each relies on representing a solution as a function f from the set of cells to the set $[n^2]$. We will use the generic term *region* to refer to a row, column, or n-aligned $n \times n$ submatrix of the sudoku board. Note that there are $3n^2$ such regions.

1. Each variable represents whether a particular ordered pair is in f; i.e. variable $x_{i,j,k}$ is represents whether cell (i,j) has the value k. We need n^6 such variables. The constraint that each cell gets at least one number can be represented by n^4 n^2-clauses. The constraint that no 2 cells in the same region get the same number can be represented by $\binom{3n^2}{n^2 2n^2 = \Theta(n^8)}$ 2-clauses. (That each cell gets at most one number is forced by the above.)
2. We represent the number in each cell in binary. This requires $2(\lg n)n^4 = \Theta(n^4 \lg n)$ variables. That each cell gets a number is now represented implicitly. That no 2 cells in the same region get the same number can be represented with $\binom{3n^2}{n^2 2n^2 = \Theta(n^8)}$ $4 \lg n$-clauses.

Despite the fact that the 2nd method generates fewer variables and a smaller maximum clause width, SAT solvers would probably perform worse on such instances than on the instances generated by the 1st method since the 1st method generates so many 2-clauses, which promote unit propagation.

Russell's solution: (a bit more detailed and has a different approach)

There are some obvious ways of performing the reduction. In these ways, the variables are $x_{i,j,k}$ representing "$M[i][j] = k$". Let $B_{(a,b)}$ be the (a, b)'th block, the set of positions (i, j) with $(a-1)n+1 \leq i \leq an$ and $(b-1)n+1 \leq j \leq bn$.

We can define the following sets of clauses:

$V_{\leq 1}$: (at most one value)

For each (i, j), there is at most one k with $M[i][j] = k$: for each $1 \leq i \leq n^2, 1 \leq j \leq n^2$, $1 \leq k < k' \leq n^2$, $\neg x_{i,j,k} \vee \neg x_{i,j,k'}$.

($O(n^6)$ clauses of size 2.)

$V_{\geq 1}$: (at least one value)

For each (i, j), there is at least one such k with $M[i][j] = k$: for each $1 \leq i \leq n^2, 1 \leq j \leq n^2$, $\vee_{1 \leq k \leq n^2} x_{i,j,k}$. ($O(n^4)$ clauses of size n^2)

$R_{\leq 1}$: (at most one k in each row)

For each row, k appears at most once: for each $1 \leq i \leq n^2, 1 \leq k \leq n^2$, $1 \leq j < j' \leq n^2$, $\neg x_{i,j,k} \vee \neg x_{i,j',k}$. ($O(n^8)$ clauses of size 2.)

$R_{\geq 1}$: (at least one k in each row)

For each row there is at least one k: with $M[i][j] = k$: for each $1 \leq i \leq n^2, 1 \leq k \leq n^2$, $\vee_{1 \leq j \leq n^2} x_{i,j,k}$.

($O(n^4)$ clauses of size n^2)

$C_{\leq 1}$ For each column, k appears at most once: for each $1 \leq j \leq n^2, 1 \leq k \leq n^2$, $1 \leq i < i' \leq n^2$, $\neg x_{i,j,k} \vee \neg x_{i',j,k}$. ($O(n^8)$ clauses of size 2.)

$C_{\geq 1}$: (at least one k in each column)

For each $1 \leq j \leq n^2, 1 \leq k \leq n^2$, $\vee_{1 \leq i \leq n^2} x_{i,j,k}$.

($O(n^4)$ clauses of size n^2)

$B_{\leq 1}$ For each block, k appears at most once: for each $1 \leq a \leq n, 1 \leq b \leq n, 1 \leq k \leq n^2, (i, j) \in B_{a,b}, (i', j') \in B_{a,b}, (i, j) \neq (i', j')$,

$\neg x_{i,j,k} \vee \neg x_{i',j,k}$. ($O(n^8)$ clauses of size 2.)

$B_{\geq 1}$: (at least one k in each block)

For each $1 \leq a \leq n, 1 \leq b \leq n, 1 \leq k \leq n^2$, $\vee_{(a-1)n+1 \leq i \leq an, (b-1)n+1 \leq j \leq bn} x_{i,j,k}$.

($O(n^4)$ clauses of size n^2)

Many combinations of these constraints are valid reductions. (Together with $x_{i,j,k}$ whenever we are given that $M[i, j] = k]$).

For example, $V_{\geq 1}, R_{\leq 1}, B_{\leq 1}, C_{\leq 1}$. People were conjecturing different kinds of tradeoffs between more clauses versus smaller clause size. But actually, for complete methods (algorithms that explore all possibilities), it is usally better to have as many constraints as possible. My guess is it would be better to just include all constraints above than to select subsets for these methods. Other heuristic search methods such as WalkSAT might get confused by a large number of "less important" constraints. However, all of these methods have $O(n^6)$ variables, which may make them prohibitively large.

But here's another method that uses $O(n^5)$ variables instead. Consider variables $Y_{i,b,k}$ and $Z_{a,j,k}$ for $1 \leq i \leq n^2, 1 \leq a, b \leq n, 1 \leq k \leq n^2$ that express: there is a k in row i in a column between $n(b-1)+1$ and nb (i.e., in the intersection of row i and block $B_{idivn,b}$) and likewise for columns. These variables also determine the solution by $M[i][j] = k$ if and only if $Y_{i,jdivn,k}$ and $Z_{idivn,j,k}$. Plus, the above constraints can be "factored" into these new variables:

For example: $R_{\leq 1}$: (at most one k in each row)

For each $1 \leq i \leq n^2, 1 \leq k \leq n^2$, $1 \leq b < b' \leq n$, $\neg Y_{i,b,k} \vee \neg Y_{i,b',k}$. ($O(n^6)$ clauses of size 2.)

$R_{\geq 1}$: (at least one k in each row)

For each row there is at least one k: with $M[i][j] = k$: for each $1 \leq i \leq n^2, 1 \leq k \leq n^2$, $\vee_{1 \leq b \leq n} Y_{i,b,k}$.

($O(n^4)$ clauses of size n)

This both reduces the number of variables from n^6 to $2n^5$ and the maximum clause width from n^2 to n. (We'll also have some clauses of size 4 saying that there can't be two values in the same square, $\neg Y_{i,b,k} \vee \neg Y_{i,b,k'} \vee \neg Z_{a,j,k} \vee \neg Z_{a,j,k'}$ for $1 \leq i \leq n^2, 1 \leq j \leq n^2, a = idivn, b = jdivn, 1 \leq k < k' \leq n^2$.) My feeling is that this would be a dramatic improvement over the reductions above, but no experiments have been made.

Sodoku The *sudoku* problem of size n is as follows. The input is an $n^2 \times n^2$ matrix M whose entries are either "blank" or an integer between 1 and n^2. A solution fills in the blank spaces with integers between 1 and n^2. The following constraints must be met: Each integer from 1 to n^2 appears exactly once in each row, in each column, and in each $n \times n$ sub-matrix of the form $M[jn+1...(j+1)n][in+1..(i+1)n]$ for each $0 \leq i, j \leq n-1$. The problem is to find any solution meeting the constraints, or return “no solution possible” if there is no such solution.

Give at least two ways of reducing sudoku to $CNFSAT$. For each, specify how many variables and clauses are created as a function of n, and also the size of the different categories of clauses (e.g., clauses of n variables vs. clauses with n^2 variables.) Give a guess for which reduction methods you would expect to work well with SAT solvers and why (see next assignment).