Advanced Complexity
Final Homework

Due on Monday, October 23rd, 2017, 2 p.m.

All questions below are subjectively given 1 to 3 stars depending on whether they are more or less difficult, hence require more careful and detailed answers.

Questions asking for a reduction require defining a reduction rigorously and arguing its correctness.

Since this is to be done at home, the marking of answers will be especially strict w.r.t. mathematical rigour and completeness of the reasoning. Rigour does not mean length: most questions can be answered completely in just two or three sentences, and all in less than ten.

1 \( \nabla \text{NP} \), a new complexity class.

We write \( \nabla \text{NP} \overset{\text{def}}{=} \{ L_1 \setminus L_2 \mid L_1, L_2 \in \text{NP} \} \) for the class of all languages obtained by taking some language in \( \text{NP} \) and removing all words from another language in \( \text{NP} \).

(*) 1. Show that \( \text{YesNoSAT} \in \nabla \text{NP} \) where \( \text{YesNoSAT} \overset{\text{def}}{=} \{ (F,G) \mid F \in \text{SAT}, G \notin \text{SAT} \} \).

\[ \text{Solution:} \]
Take \( L_1 = \{ (F,G) \mid F \in \text{SAT} \} \) and \( L_2 = \{ (F,G) \mid G \in \text{SAT} \} \). Obviously \( L_1, L_2 \leq L \text{SAT} \) hence both languages are in \( \text{NP} \).

(*) 2. Show that \( \text{Prime} \in \nabla \text{NP} \), where \( \text{Prime} \) is the language of all prime numbers (written in base 2).

\[ \text{Solution:} \]
The set of non-primes is in \( \text{NP} \) (guess a factorization and check it in polynomial time), so \( \text{Prime} = L_1 \setminus L_2 \) for \( L_1 = \text{"all numbers"} \) and \( L_2 = \text{"the non-primes"} \).

A *Number Expression* (or NE) \( e \) is an expression made up of natural numbers, and symbols ’+’ and ’\( \cup \)’ according to the following inductive definition:

\[ e ::= n \mid (e_1 \cup e_2) \mid (e_2 + e_2) \]

where \( n \in \mathbb{N} \) is any number. We often omit some of the parentheses when writing NEs. We define \(|e|\), the size of \( e \), as the number of occurrences of the symbols ’0’, ’1’, ’+’ and ’\( \cup \)’ in \( e \), assuming that numbers are written in binary. (NB: the real size on a TM tape includes the parentheses, hence is \( O(|e|) \).)

An NE is interpreted as a subset \( V(e) \) of \( \mathbb{N} \), defined by

\[ V(n) \overset{\text{def}}{=} \{ n \} \, , \, V(e_1 \cup e_2) \overset{\text{def}}{=} V(e_1) \cup V(e_2) \, , \]
\[ V(e_1 + e_2) \overset{\text{def}}{=} \{ n_1 + n_2 \mid n_1 \in V(e_1), n_2 \in V(e_2) \} \, . \]
3. Let $0 < n \in \mathbb{N}$ be a positive number and consider $e_n = (1 \cup 2) + (2 \cup 4) + \cdots + (2^{n-1} \cup 2^n)$. What is $V(e_n)$ and $|e_n|$?

**Solution:**

$|e_n| = 2n - 1 + 2 \sum_{i=0}^{n} |2^i| - |2^0| - |2^n|$. Using $|2^i| = i + 1$ we get $|e_n| = 2n - 1 + (n + 1)(n + 2) - 1 - n - 1 = n^2 + 4n - 1$.

Let us prove $V(e_n) = [2^n - 1, 2^{n+1} - 1]$ by induction on $n$. Base case: $V(e_1) = V(1 \cup 2) = [1, 2] = [1, 3] = [2^1 - 1, 2^2 - 1]$. Inductive step: $V(e_{n+1}) = V(e_n) + 2^n \cup V(e_n) + 2^{n+1} = (\text{by ind. hyp.}) [2^n - 1, 2^{n+1} - 1] + 2^n \cup [2^n - 1, 2^{n+1} - 1] + 2^{n+1} = [2^n - 1 + 2^n, 2^{n+1} - 1 + 2^n] \cup [2^n - 1 + 2^n, 2^{n+1} - 1 + 2^n] = [2^{n+1} - 1, 2^{n+2} - 1]$.

Let $\text{IsolVal} = \{(e, n) \mid n \in V(e) \land n - 1, n + 1 \notin V(e)\}$. In other words, we consider the problem of checking whether a given number appears as an isolated value in some set of numbers denoted by a NE.

4. (*) Show that $\text{IsolVal} \in \nabla \np$.

**Solution:**

For $k \in \mathbb{Z}$, let $L_k = \{(e, n) \mid n + k \in V(e)\}$. Each $L_k$ is in $\np$ (for each `∪` in $e$, choose one summand; this provides a sum of numbers that we just have to evaluate and compare with $n + k$). Similarly any $L_k \cup L_m$ is in $\np$. Now $\text{IsolVal} = L_0 \setminus (L_1 \cup L_{-1})$.

AlmostSAT is the language of Boolean formulae $S$ in clausal normal form $S = C_1 \land \cdots \land C_n$ such that (1) $S$ is unsatisfiable, and (2) for every $i = 1, \ldots, n$, the reduced formula $S \setminus C_i \defeq C_1 \land \cdots \land C_{i-1} \land C_{i+1} \land \cdots \land C_n$ is satisfiable. I.e., $S$ is unsatisfiable but removing any clause makes it satisfiable.

5. (*) Show that AlmostSAT $\in \nabla \np$.

**Solution:**

Idea: Checking that all the $S \setminus C_i$ are satisfiable can be done in $\np$.

6. (*) Show that $\np \cup \co\np \subseteq \nabla \np$.

**Solution:**

For $L \in \np$, take $L_1 = L$ and $L_2 = \emptyset$. For $L \in \co\np$, take $L_1 = \Sigma^* \setminus L$ and $L_2 = \Sigma^* \setminus L$.

7. (***) Prove that $\nabla \np$ is either closed under unions, or closed under intersections.

**Solution:**

If $L = L_1 \setminus L_2$ and $L' = L'_1 \setminus L'_2$ then $L \cap L' = (L_1 \cap L'_1) \setminus (L_2 \cup L'_2)$. Furthermore, and since $\np$ is closed under intersection and union, $L_1 \cap L'_1$ and $L_2 \cup L'_2$ are in $\np$ when $L_1, L'_1, L_2, L'_2$ are. We conclude that $\nabla \np$ is closed under intersections. (NB: It is not known whether $\nabla \np$ is closed under unions.)

2 A few simple $\nabla \np$-complete problems.

8. (*) Show that YesNoSAT is $\nabla \np$-complete.

**Solution:**

Since $\text{YesNoSAT} \in \nabla \np$, we only have to show that any $L = L_1 \setminus L_2$ in $\nabla \np$ reduces to $\text{YesNoSAT}$. Since $\text{SAT}$ is $\np$-complete, there are reductions $x \mapsto S_{1,x}$ such that $x \in L_1$ iff $S_{1,x}$ is satisfiable. This means $x \in L$ iff $(S_{1,x}, S_{2,x}) \in \text{YesNoSAT}$, providing the required reduction.
Let $\text{Clique}$ be the set of all $(G,k)$ such that $G$ is a graph with a clique (a complete subgraph) of size $k$. Let $\text{BestClique}$ be the set of all $(G,k)$ such that $k$ is the size of a largest clique in $G$.

(**) 9. Assuming that $\text{BestClique}$ is NP-hard, show that $\text{BestClique}$ is $\nabla \text{NP}$-complete.

Solution:

$\text{BestClique}$ is in $\nabla \text{NP}$ because it asks whether $(G,k)$ is in $\text{Clique}$ and $(G,k+1)$ is not. In other words, we may write $\text{BestClique} = L_1 \setminus L_2$ for $L_1 = \text{Clique}$ and $L_2 = \{(G,k) \mid (G,k+1) \in \text{Clique}\}$ and observe that $L_2 \in \text{NP}$.

To show $\nabla \text{NP}$-hardness, we exhibit a reduction $L \leq_L \text{BestClique}$ for $L = L_1 \setminus L_2$ where $L_1, L_2$ are any two languages in NP. The assumption that $\text{BestClique}$ is NP-hard means that there exist logspace reductions $x \mapsto (G,k)$ such that $x \in L_1 \iff k$ is the size of a largest clique in $G$. It is also known that $\text{Clique}$ is NP-hard, so we also have a logspace reductions $x \mapsto (G',k')$ such that $x \in L_2 \iff k'$ is the size of a clique in $G'$. We modify the second reduction by adding a clique of size $k' - 1$ to $H$, ensuring that the minimum size of a clique in $H$ is $k'$ if $x \in L_2$, $k' - 1$ otherwise. We also make sure that $k \neq k'$ by adding some nodes and edges to one graph if necessary. If we now take the product graph $G \times G'$, its largest clique has size $k \times (k' - 1)$ iff $(G,k) \in \text{BestClique}$ and $(G',k') \notin \text{Clique}$, iff $x \in L_1$ and $x \notin L_2$.

(**) 10. Show that $\text{IsolVal}$ is $\nabla \text{NP}$-complete (Hint : $\text{SubsetSum}$ is NP-hard! Recall that $\text{SubsetSum}$ asks, given a set $\{a_1, \ldots, a_k\}$ of natural numbers, and a target $t \in \mathbb{N}$, whether $t = \sum_{i \in J} a_i$ for some $J \subseteq \{1, \ldots, k\}$).

Solution:

The hint is useful since a $\text{SubsetSum}$ instance “$n$ from $\{a_1, \ldots, a_k\}$?” can be written “$n \in V(e)$?” with $e = (a_1 \cup 0) + \cdots + (a_k \cup 0)$.

Let $L = L_1 \setminus L_2$ with $L_1, L_2 \in \text{NP}$. Since $L_i \leq_L \text{SubsetSum}$, there are two logspace reductions $x \mapsto (n_{i,x}, e_{i,x})$ such that, for $i = 1, 2$, $x \in L_i$ iff $n_{i,x} \in V(e_{i,x})$. We claim that

$$x \in L_1 \setminus L_2 \iff 3n_{1,x} n_{2,x} \text{ is isolated in } V(3n_{2,x} \cdot e_{1,x} \cup (3n_{1,x} \cdot e_{2,x} + 1))$$

(†)

Here $a \cdot e$ denotes an NE obtained by replacing every $n$ in $e$ by $an$, so that $V(a \cdot e)$ = \{an | n \in V(e)\}. We have thus provided a logspace reduction from $L$ to $\text{IsolVal}$, so $\text{IsolVal}$ is $\nabla \text{NP}$-hard, and Question 4 already showed it is in $\nabla \text{NP}$.

3  A more complex reduction.

Let $Z = \{z_1, \ldots, z_n\}$ be $n$ Boolean variables and define the following clauses for all $i = 1, \ldots, n$, and for all $j = 1, \ldots, n$ such that $i \neq j$:

$$C^Z \equiv z_1 \lor z_2 \lor \cdots \lor z_n, \quad C^Z_i \equiv z_1 \lor \cdots \lor z_{i-1} \lor \neg z_i \lor z_{i+1} \lor \cdots \lor z_n, \quad D^Z_{i,j} \equiv \neg z_i \lor \neg z_j.$$  

(*) 11. Show that $S^Z \overset{\text{def}}{=} C^Z \land \bigwedge_i C^Z_i \land \bigwedge_{i \neq j} D^Z_{i,j}$ is in $\text{AlmostSAT}$. Don’t forget the case $n = 0$.

Solution:

There is an error in the question. By allowing all clauses $D_{i,j}$ for $i \neq j$ we make $S^Z$ unsatisfiable even after we remove some clause $\neg z_i \lor \neg z_j$ since $\neg z_j \lor \neg z_i$ is still there. The right formulation defines $S^Z$ with only $\bigwedge_{1 \leq i < j \leq n} D^Z_{i,j}$. Note that the answers below assumes this corrected formulation. Students who say that $S^Z$ as given is not in $\text{AlmostSAT}$ will be considered as providing a correct answer.

$S^Z$ is unsatisfiable: if a valuation sets all variables to false it fails on $C^Z$, if it sets just one to true it fails on the corresponding $C^Z_i$, and if it sets at least two variables to true it fails on the corresponding $D^Z_{i,j}$.
The above reasoning shows how removing any clause in $S^Z$ allows finding a satisfying valuation. E.g., when removing $C_i^Z$ we use the valuation where only $z_i$ is set to true, and when removing $D_{i,j}^Z$ we use the valuation where only $z_i$ and $z_j$ are true.

NB: if $n = 0$, $S^Z = C^Z = \emptyset \supset \bot$ (i.e., the only clause is the empty disjunction) which belongs to AlmostSAT since removing the only clause yields $\top$ (the empty conjunction).

(***) 12. Show coSAT $\leq_L$ AlmostSAT. For this reduction we suggest introducing one extra Boolean variable for each clause in the original instance and using them to modify the original clauses (and to add new clauses), reusing ideas from question 11.

Solution:
coSAT is the union of Not a CNF, the language of all strings that are syntactically not a SAT instance, and UNSAT, the language of formulas in clausal form that are unsatisfiable. Since Not a CNF is trivial, we have obviously coSAT $\leq_L$ UNSAT and it is enough to show UNSAT $\leq_L$ AlmostSAT.

In fact, it is enough to consider 3UNSAT where we restrict to formulas in 3CNF. With $S = F_1 \land \cdots \land F_n$ such that $F_i = l_{i,1} \lor l_{i,2} \lor l_{i,3}$ and $l_{i,j}$ is a literal $x_{i,j}$ or $\neg x_{i,j}$ we associate new clauses $F'_i \equiv F_i \lor C'_i$ where $C'_i \equiv z_1 \lor \cdots \lor z_{i-1} \lor z_{i+1} \lor \cdots \lor z_n$ is inspired by question 11 and use new variables from $Z = \{z_1, \ldots, z_n\}$. We further consider $G_i,k$ (for $k = 1, \ldots, 3$) given by $G_{i,k} = \neg l_{i,k} \lor C'_i$, and all clauses $D_{i,j}^Z \equiv \neg z_i \lor \neg z_j$ as in question 11. We claim that the obtained $S'' \equiv \bigwedge_i F'_i \land \bigwedge_i \bigwedge_{k=1}^3 G_{i,k} \land \bigwedge_{i \neq j} D_{i,j}^Z$ is in AlmostSAT iff $S$ is unsatisfiable.

For this we check (1) if $S$ is satisfiable, $S''$ is also satisfiable: extend the valuation for $S$ by setting all $z_i$’s to false; (2) if $S''$ is satisfiable, $S$ is too: the valuation $\nu$ for $S''$ cannot set two $z_i$’s to true because of the $D_{i,j}$’s and if it sets just one $z_i$ to true, it must satisfy the corresponding $F_i$ —from satisfying $F_i$’s—and the three $\neg l_{i,k}$’s— from satisfying the $G_i,k$’s—which is impossible. Thus $\nu$ sets all $z_i$’s to false and, since it satisfies all $F_i$’s, it also satisfies $S$: (3) if $S''$ is unsatisfiable, removing any clause makes it satisfiable: this is now clear.

(**) 13. Show now SAT $\leq_L$ AlmostSAT with similar ideas.

Solution:
With $S$ we associate $S'' = S' \land C^Z$ with $S'$ as above and $C^Z$ from question 11. Now $S''$ is unsatisfiable, whatever $S$. Furthermore, if $S$ is unsatisfiable, we can remove $C$ from $S''$, obtaining $S'$ that is still unsatisfiable. And if $S$ is satisfiable, removing any clause in $S''$ yields a satisfiable formula as seen in previous question. Finally, $S''$ is in AlmostSAT iff $S$ is in SAT.

Let $\text{doubleAlmostSAT} \equiv \{ (S, S') \mid S, S' \in \text{AlmostSAT} \}$.

(**) 14. Provide a reduction showing that $\text{doubleAlmostSAT} \leq_L \text{AlmostSAT}$.

Solution:
With $S = \bigwedge_i C_i$ and $S' = \bigwedge_j C'_j$ such that $S$ and $S'$ use disjoint sets of Boolean variables, we associate $T \equiv \bigwedge_i \bigwedge_j (C_i \lor C'_j)$.

1. We first claim that $T$ is satisfiable iff $S$ or $S'$ is (proof omitted) hence $T \not\in \text{SAT}$ iff $S, S' \not\in \text{SAT}$.

2. Then we claim that if $S, S' \in \text{AlmostSAT}$ then $T \in \text{AlmostSAT}$. We already saw that $T$ is not satisfiable. Now consider any $T \setminus (C_i \lor C'_j)$. Since $S$ is in AlmostSAT, there is a valuation $\nu$ that makes $S \setminus C_i$ true. And some $\nu'$ makes $S' \setminus C'_j$ true. Now $\nu \cup \nu'$ validates any $C_k \lor C'_l$ where $k \neq i \lor l \neq j$, hence validates $T \setminus (C_i \lor C'_j)$.

3. Finally, if $T$ is in AlmostSAT then $S, S'$ are too: assume $T \in \text{AlmostSAT}$ (hence $S, S' \not\in \text{SAT}$) and consider any $S' \setminus C_i$. We need to show it is satisfiable. But, picking any $j$, we know that $T \setminus (C_i \lor C'_j)$ is satisfiable, say by $\nu$. Since $S'$ is not satisfiable, there is some $C'_k$
that is not satisfied by \( v \). However all \( C_i' \lor C_j' \), for \( i' \neq i \), occur in \( T \setminus (C_i \lor C_j') \) hence are validated by \( v \). We deduce that \( v \) validates \( C_i' \) for any \( i' \neq i \). Thus it validates \( S \setminus C_i \).
We now have the reduction, perhaps by adding some renaming to ensure that \( S \) and \( S' \) do not share any variable.

(*) 15. Now show that AlmostSAT is \( \nabla \text{NP} \)-complete.

**Solution:**
Since we already know that AlmostSAT \( \in \nabla \text{NP} \) we only have to prove that AlmostSAT is \( \nabla \text{NP} \)-hard. For this it is enough to show YesNoSAT \( \leq_L \) AlmostSAT. Combining the reductions in Questions 12 and 13, we obtain a reduction YesNoSAT \( \leq_L \) doubleAlmostSAT. We further reduce to AlmostSAT with Question 14.

4 A strict class?

(**) 16. Assuming NP \( \neq \text{coNP} \), show that BestClique \( \notin \text{NP} \).

**Solution:**
Let us show that BestClique is \( \text{coNP} \)-hard by reducing from \( \overline{\text{Clique}} \), well-known to be \( \text{coNP} \)-complete: here \((G, k) \in \text{Clique} \) iff the largest clique in \( G \) has size \(< k \), iff \((G \cup K_{k-1}, k-1) \in \text{BestClique} \), i.e., iff the largest clique in \( G \) extended by a clique of size \( k - 1 \) has size \( k - 1 \). This provides the required reduction and shows that any problem in \( \text{coNP} \) can be reduced to BestClique.
Thus if we assume BestClique \( \in \text{NP} \), we conclude that \( \text{coNP} \subseteq \text{NP} \), hence \( \text{NP} = \text{coNP} \). Therefore \( \text{NP} \neq \text{coNP} \), implies BestClique \( \notin \text{NP} \).

(**) 17. Assuming NP \( \neq \text{coNP} \), show that BestClique \( \notin \text{coNP} \).

**Solution:**
We prove the contraposition “BestClique \( \in \text{coNP} \) implies \( \text{NP} = \text{coNP} \).”
Clearly \((G, k) \in \text{Clique} \) iff \((G, k') \in \text{BestClique} \) for some \( k' \) between \( k \) and \(|G|\). Equivalently
\[
(G, k) \in \text{Clique} \iff \bigwedge_{k \leq k' \leq |G|} (G, k') \in \text{BestClique}.
\]
Assume now BestClique \( \in \text{coNP} \), i.e., there exists a polynomial-time NTM \( M \) accepting BestClique. One can transform \( M \) and obtain a NTM \( M' \) that receives \((G, k)\) as input, guesses accepting runs of \( M \) for all \((G, k')\) where \( k' = k, \ldots, |G| \), and thus accepts \((G, k)\) iff all \((G, k')\) are in \text{BestClique}. Therefore \( M' \) is a polynomial-time NTM that accepts \text{Clique}, proving \text{Clique} \( \in \text{NP} \). As with the previous question, this implies \( \text{NP} = \text{coNP} \) since \text{Clique} is \( \text{coNP} \)-complete.

(*) 18. Conclude that if NP \( \neq \text{coNP} \), then NP \( \cup \text{coNP} \subseteq \nabla \text{NP} \).

**Solution:**
Assume NP \( \neq \text{coNP} \). Then BestClique \( \notin \text{NP} \) and BestClique \( \notin \text{coNP} \) as just seen. Since also BestClique \( \in \nabla \text{NP} \) and NP \( \cup \text{coNP} \subseteq \nabla \text{NP} \), we deduce NP \( \cup \text{coNP} \subseteq \nabla \text{NP} \).