COMS 4236: Intro to Computational Complexity — Assigned September 23, 2026

Homework 1 Solutions

Instructor: *Toniann Pitassi* — Due: October 1, 2026

1 How short can NP witnesses be?

Solution.

(a) Since A has accepting witnesses of length at most $c \log n$, we can give a polynomial time algorithm for A. The algorithm will search through all possible witnesses of length at most $c \log n$ and check each one using the verifier. The algorithm is correct by the correctness of the verifier. It takes polynomial time because there are $2^{c \log n + 1} - 1 = \text{poly}(n)$ many witnesses, and each one requires polynomial time to check with the verifier. Since A is NP-complete, this shows that $\mathsf{P} = \mathsf{NP}$.

(b) In this case, the same algorithm requires time $2^{c \log^2 n + 1} - 1 = O(2^{c \log^2 n})$. For another language $A' \in \mathsf{NP}$, there exists a polynomial time f such that $x' \in A'$ if and only if $f(x') \in A$. Since f is polynomial time, $|f(x')|$ is polynomial in $|x'|$. Therefore, running the algorithm for A on $f(x')$ requires time $O(2^{c \log^2 |f(x')|}) = O(2^{ck^2 \log^2 |x'|})$ for some integer k. This is quasipolynomial in $|x'|$. This would imply that every language in NP has a quasipolynomial time algorithm. This is unlikely, because we believe problems in NP to be hard, but has not been proved.

2 Resolution, and NP versus coNP

Solution.

(a) If $(x \vee C') \wedge (\neg x \vee D')$ is satisfiable, then $C' \vee D'$ is satisfiable by casework on x. Now for contradiction assume F is satisfiable and it has a resolution refutation. Then inductively, any derived clause is satisfiable, including the final derived empty clause, a contradiction.

(b) Notice that if a resolution refutation exists, then it has size at most $O(n3^n)$, since there can only be that many unique clauses.

To show completeness, we induct on the number of variables. When there are no variables, we are done, so suppose there exists at least one variable x. Consider the new set of clauses F' consisting of all clauses from F that doesn't contain variable x, together with $A \vee B$ where $x \vee A$ and $\neg x \vee B$ are both present in F. By part (a), if F is satisfiable, then F' is satisfiable.

We now show the converse. We may WLOG assume F contains a clause of the form $x \vee A$ and $\neg x \vee B$. Let $A_1, \ldots, A_a$ be all clauses such that $x \vee A_i \in F$, and $B_1, \ldots, B_b$ be all clauses such that $\neg x \vee B_j \in F$. Suppose F' is satisfiable. Suppose A_i is unsatisfied on a satisfying assignment for some i, then all B_js must be satisfied, so by setting $x = 1$ satisfies all $x \vee A_i$s and $\neg x \vee B_j$s. Similarly if some B_j is unsatisfied, then all A_is are satisfied, so setting $x = 0$ satisfies every clause in F.

This process eliminates one variable and by induction, if F is unsatisfiable, we can eliminate all variables through resolution refutation.

(c) Encode each clause as a sorted list of signed variable indices, and encode F as a list of clauses. Each proof line records its clause and either an input-clause index or two earlier proof-line indices and a pivot variable. The verifier checks that each input line matches its referenced clause of F, each derived line is exactly the indicated resolvent, and the final clause is empty. All these checks take time polynomial in the combined encoding length $|F| + |\Pi|$.

(d) By part (c), this means that UNSAT $\in$ NP. Since UNSAT is coNP-complete, coNP $\subseteq$ NP. Since UNSAT $\in$ NP implies SAT $\in$ coNP, by the same argument, NP $\subseteq$ coNP. Therefore NP = coNP.

3 Poly Hierarchy and Alternating Quantifiers

Solution.

(a) You can modify the verifier to always accept on u values longer than $\lceil c \log n \rceil$. Then, the language is $\{x \mid \forall u \in \{0,1\}^{|x|} \exists v \, R(u, v, x)\}$, which is in Π_2^P by definition.

To show $L \in \mathsf{NP}$, we construct a witness and verifier R' for L. Suppose $x \in L$. We define U to be a list of all strings in $\{0,1\}^{\lceil c \log n \rceil}$ in lexicographic order. Then, for each u_i in U, take v_i such that $R(u_i, v_i, x) = 1$. Our witness is $v_0 \circ v_1 \circ \cdots$. Our verifier R' will iterate though each i and check that $R(u_i, v_i, x) = 1$. If any checks fail, R' outputs 0. Otherwise, it outputs 1.

We show that our verifier is correct: $x \in L$ iff for all u_i, there exists v_i such that $R(u_i, v_i, x)$. This happens iff there exists a witness $v_0 \circ v_1 \circ \cdots$ causing R' to accept.

We show that the witness has polynomial length. There are $2^{\lceil c \log n \rceil + 1} - 1 = \text{poly}(n)$ strings in the witness, and each one has polynomial length, so the total length is polynomial.

Finally, we show that the verifier runs in polynomial time. The verifier has to make $2^{\lceil c \log n \rceil + 1} - 1 = \text{poly}(n)$ many calls to R, and each call takes polynomial time. Therefore the total runtime is polynomial.

(b) If u has polynomial length, $2^{|u|+1} - 1$ is not polynomial. If we used the idea above, the witness length and verifier runtime would both be greater than polynomial.

4 Languages between P and EXP

Solution.

(a) We will show that P $\subseteq$ TIME($t(n)$) $\subsetneq$ TIME($T(n)$) $\subseteq$ QP.

Take any $k > 0$, for $n \geq 2^{2^k}$, we have $n^k \leq n^{\log \log n}$. Therefore TIME(n^k) $\subseteq$ TIME($n^{\log \log n}$). Taking union over all k gives P $\subseteq$ TIME($n^{\log \log n}$).

Now consider $\frac{t(n) \log t(n)}{T(n)} \leq \frac{t(n)^2}{T(n)} = \frac{n^{2 \log \log n}}{n^{\log n}} = \frac{1}{n^{\log n - 2 \log \log n}}$ for n large enough. Since $\log n - 2 \log \log n \to \infty$ as $n \to \infty$, we get that $\lim_{n \to \infty} \frac{t(n)^2}{T(n)} = 0$, proving $t(n) \log t(n) \leq o(T(n))$.

Finally, $\text{TIME}(n^{\log n}) \subseteq \text{QP}$ by definition.

(b) Take $t(n) = 2^n, T(n) = 2^{n^2}$, and one can show $\text{QP} \subseteq \text{TIME}(t(n)) \subsetneq \text{TIME}(T(n)) \subseteq \text{EXP}$ using similar method in part (a).

5 Compositions of logspace reductions

Solution.

We construct a TM M computing $g(f(x))$. Let M_g be a logspace TM computing g and let M_f be a logspace TM computing f.
M works as follows. It runs M_g, but every time M_g queries the i-th cell of its input it simulates a modified version of M_f as follows. It maintains a single output cell, initalized at $\not\emptyset$, and a counter tracking the output head position of M_f. When M_f writes an output, if the counter is not i, it ignores it. When M_f moves the output head, it updates the counter accordingly. When M_f outputs to the i-th cell, it stores the output in the single output cell. When M_f halts, it returns the single output cell.
The simulation of M_f only requires $O(\log n)$ space (logspace on all of the work tapes, plus a counter of length $O(\log n)$, plus a single output cell). The simulation of M_g requires space logarithmic in the input length $\text{poly} \log n$, which is also $O(\log n)$.

6 Unary NP

Solution.

Let $L \in \mathsf{NEXP}$. Then, there exists a polynomial $p(n)$ and a verifier $V(x, u)$ that runs in time $2^{p(|x|)}$ such that $x \in L$ if and only if there exists u with length $2^{p(|x|)}$ such that $V(x, u) = 1$.
Let $f : \{0,1\}^* \times \mathbb{N} \to 1^*$ be defined as follows. On an input $(x, p(|x|))$, we first construct the binary string consisting of x with every bit repeated twice, then the separator symbol 01, then $1^{p(|x|)}$. Then, $f(x)$ is defined as the unary encoding of 1 followed by this binary string.
There exists an inverse f^{-1}. To compute f^{-1} on $x' \in 1^*$, first encode x' in binary. Then, remove the leading 1. Then, read bits until reaching the separator symbol and find x by deduplicating bits. Finally, find $p(|x|)$ by counting bits after the separator symbol.
Define

$$L' = \{x' \in 1^* \mid \text{the binary encoding of } x' \text{ is of the form } 1 \circ \{00, 11\}^* \circ 01 \circ 1^* \text{ and } f^{-1}(x')_1 \in L\}.$$

We claim this language is in NP. To see this, we give a verifier $V'(x', u)$. V' first checks that x' is of the required form, then uses f^{-1} to obtain $(x, p(|x|))$. Then, it outputs $V(x, u)$.
We claim V' is complete and sound. If $x' \in L'$, then $f^{-1}(x')_1 = x$ where x in L. By the completeness of V, there exists u causing $V(x, u)$ to accept. If there exists u causing V' to accept, then $f^{-1}(x')_1 = x$ where $V(x, u) = 1$. By soundness of V, $x \in L$, so $x' \in L'$.
Now we show V' is polytime. Observe $|x'| \geq 2^{1+2|x|+p(|x|)}$. V' requires $\text{poly}(|x'|)$ time to check the form of x' and compute f^{-1} then $2^{p(|x|)} \leq |x'|$ time to compute V. In total, this requires time $\text{poly}(|x'|)$.
Finally, we show u is polylength: $|u| \leq 2^{p(|x|)} \leq |x'|$.

Thus, $L' \in \mathsf{NP}$. By assumption, there exists a polytime algorithm A deciding L'. Then, we give a deterministic algorithm deciding L. The algorithm is to define $x' := f(x, p(|x|))$ and output $A(x')$.
We now show A is complete and sound: $x \in L$ if and only if $x' \in L'$ if and only if $A(x') = 1$.
Finally, we show A runs in time $2^{\text{poly}(|x|)}$. Computing f requires time $\leq 2 \cdot 2^{1+2|x|+p(|x|)}$. Computing A requires time $\text{poly}(|x'|) = 2^{\text{poly}(|x|)}$. Therefore, $L \in \mathsf{EXP}$.
The opposite inclusion $\mathsf{EXP} \subseteq \mathsf{NEXP}$ is true because TMs are NTMs. We conclude $\mathsf{EXP} = \mathsf{NEXP}$.