CS 40: Computational Complexity

Sair Shaikh

September 25, 2025

Problem 1. For a complexity class $\mathcal{C}$, define two new complexity classes $\exists\mathcal{C}$ and $\forall\mathcal{C}$ as follows.

$$\exists\mathcal{C} = \{\{x \in \Sigma^* : \exists y \in \Sigma^* \text{ with } |y| = \text{poly}(|x|) \text{ such that } \langle x, y\rangle \in L_0\} : L_0 \in \mathcal{C}\}$$
$$\forall\mathcal{C} = \{\{x \in \Sigma^* : \forall y \in \Sigma^* \text{ with } |y| = \text{poly}(|x|) \text{ we have } \langle x, y\rangle \in L_0\} : L_0 \in \mathcal{C}\}$$

The notation $|y| = \text{poly}(|x|)$ means that there exist fixed constants $c, k > 0$ such that $|y| \leq c|x|^k$ for all $|x| \geq 1$. Prove, rigorously, that $\exists\text{P} = \text{NP}$ and $\forall\text{P} = \text{coNP}$.
Use only the basic definitions, where NP is defined using NDTMs and

$$\text{coNP} = \{L \subseteq \Sigma^* : \overline{L} \in \text{NP}\}$$

In your proofs, make sure you precisely defined appropriate languages L_0 used in the definitions above.

Solution. We handle each part seperately.

(a) First, we show that NP $\subseteq \exists P$. Let L be an arbitrary language in NP. We need to show that $L \in \exists P$.

Since $L \in$ NP, for each input string $x \in L$, there exists a computation path, with number of transitions polynomial in $|x|$, that an NDTM can take to reach an accept state. Moreover, the description of each configuration reached in this computational path is also upper-bounded by a polynomial in x (the tape contents are at most polynomial as otherwise the NDTM will not be able to read/write this in polynomial time). Thus, the description of the computation path to reach an accept state is also polynomial in x.

Define $y(x)$ to be this description if $x \in L$ and $y(x) = \perp$ if $x \notin L$. ($\perp$ is a stand-in for any character that is not in the alphabet for L).

Next, we can design a deterministic TM, M, that takes as input $\langle x', y' \rangle$ (with appropriate delimiters) and does the following:

- If $y' = \perp$, reject.
- Otherwise, mimic the computation conducted by the NDTM on input x', using y' as a description of the computation path to take deterministically.

The language that M decides is $L_0 := \{\langle x, y(x) \rangle : x \in L\}$. Since this computation is polynomial-time in the input, we have $L_0 \in P$. We have shown that:

$$x \in L \iff \exists y(x), |y| = poly(|x|), \langle x, y \rangle \in L_0 \in P$$

Thus, $L \in \exists P$. Thus, NP $\subseteq \exists P$.

Next, we need to show that $\exists P \subseteq$ NP. Let $L \in \exists P$ be arbitrary. We need to show that $L \in$ NP.

Let $L_0 \in$ P be the language associated to L in the definition of $\exists P$. Let M' be the deterministic polynomial-time TM that decides L_0. We design a NDTM N as follows: Given current tape contents a,

(a) First run M' on the input a. Accept if M' accepts.

(b) If not, transition to the configurations corresponding to the starting configurations of M' on $a \circ c$ for each $c \in \Sigma$.

Since Σ is finite, the number of states N transitions to is also finite. Moreover, for any input x, N will continue adding characters to x, running M' on each string non-deterministically. Then, if there exist y_x with $|y_x| \in poly(|x|)$ such that $\langle x, y \rangle \in L_0$, N will find (by adding) y_x in polynomial time, run M' in polynomial time, and accept. If there is no such y_x, N will not halt and continue adding characters. Thus, $L(N) = L$.

Thus, $L \in$ NP (not sure if this is true as N recognizes but doesn't decide L), thus $\exists$P $\subseteq$ NP. Thus, NP $= \exists$P.

(b) To show that coNP $= \forall P$, we will show that for any L, $L \in$ coNP $\iff$ $L \in \forall$P.

We know $L \in$ coNP if and only if $\overline{L} \in$ NP.

By the previous part, $\overline{L} \in$ NP if and only if $\overline{L} \in \exists$P.

Next, we will show that $\overline{L} \in \exists \text{P} \iff L \in \forall \text{P}$. Note that $\overline{L} \in \exists P$ if and only if there exists $L_0 \in \text{P}$, such that:

$$x \in \overline{L} \iff \exists y_x, |y_x| \in poly(|x|) \text{ such that } \langle x, y_x \rangle \in L_0$$

Taking the complement, we have:

$$x \in L \iff \forall y_x, |y_x| \in poly(|x|), \langle x, y_x \rangle \in \overline{L_0}$$

Thus, to claim that $L \in \forall \text{P}$, we only need to show $\overline{L_0} \in \text{P} \iff L_0 \in \text{P}$ (i.e. $\text{coP} = \text{P}$). Let M be a polynomial-time TM that decides L_0. Then, we can design a machine M' that runs the same computation as M, but flips the result before returning. By construction, M' decides $\overline{L_0}$. Moreover, the run-time of M' is only a constant time slower than M, and is thus polynomial in the input. Thus, $L_0 \in \text{P} \implies \overline{L_0} \in \text{P}$. For the other direction, note that complements are involutions, so the same arguement applies. Thus, $\overline{L_0} \in \text{P} \iff L_0 \in \text{P}$.

Thus, $\overline{L} \in \exists P \iff L \in \forall P$.

To summarize, we have shown:

$$L \in \text{coNP} \iff \overline{L} \in \text{NP} \iff \overline{L} \in \exists \text{P} \iff L \in \forall \text{P}$$

Thus, $\text{coP} = \forall \text{P}$.

Problem 2. Define the following two complexity classes:

$$\mathrm{EXP} = \bigcup_{i=1}^{\infty} \mathrm{DTIME}\left(2^{n^i}\right)$$
$$\mathrm{NEXP} = \bigcup_{i=1}^{\infty} \mathrm{NTIME}\left(2^{n^i}\right)$$

Prove that P = NP implies EXP = NEXP. This proof requires one creative idea, namely "padding": given a language L, think about designing a new language wherein each input instance is constructed by starting with an instance of L and "padding it out" by appending a long string of extra symbols.

Solution. Assume P = NP. We know that EXP $\subseteq$ NEXP. Thus, we need to show NEXP $\subseteq$ EXP. Towards that goal, let $L \in$ NEXP be an arbitrary language. It suffices to show $L \in$ EXP.

Since $L \in$ NEXP, there exists a positive integer i such that $\mathrm{TimeCost}_M(n) \in O(2^{n^i})$ where M is the TM that decides L. We construct a new language L' in the following manner:

- Take any string $x \in L$
- Create string x' by appending $\perp$ (assumed to be not in the alphabet for L) to the end of the string until $|x'| = 2^{n^i}$ where $n = |x|$.
- These strings x' are the strings in L'.

Similarly, we can use a modification of M that treats $\perp$ the same as an empty character to decide L'. Call this machine M'.

The computational paths of M' look identical to that of M, thus, on input x' derived from $x \in L$ with $|x| = n$, it runs in $O(2^{n^i})$ time. However, since $|x'| = 2^{n^i}$, $\mathrm{TimeCost}_{M'}(k) \in O(k)$ is linear. Thus, we conclude that $L' \in$ NP (as M' is an NDTM).

Using our hypothesis, $L' \in$ P. Thus, there exists a deterministic Turing machine T' that decides L' in time polynomial in the input. Since this machine is derived from M', which treats $\perp$ the same as empty characters, so does this machine (i.e. its internal computation do not depend on our padding). Thus, this machine also decides L.

Since the run-time of this machine is polynomial in 2^{n^i}, it is exponential in n. Thus, $L \in$ EXP. This suffices to show NEXP = EXP.