Computational Complexity **(Due: Flexible)**

Assignment #5 (Arora Barak Ch - 5)

Instructor: Shreesh Maharaj *Name:* Jimut Bahan Pal

Q. 1. Show that **APSPACE** = **EXP**.

Soln. We can show **EXP** $\subseteq$ **APSPACE**. Let **L** $\in$ **EXP**, be some language [1] decidable in 2^{n^c} time, i.e., there exists a TM M_L which runs in 2^{n^c} time deciding L. For any input x, we can look at $2^{|x|^c} \times 2^{|x|^c}$ computation table of M_L on x. The i^{th} row describes the i^{th} configuration during the computation $\sigma_1\sigma_2...\sigma_i q \sigma_{i+1}\sigma_{i+2}...\sigma_{2^{|x^c|}}$, where $\sigma_i \in \Sigma \cup \Gamma, q \in Q$.

This means that at the ith step $\sigma_1...\sigma_{2^{|x|^c}}$ is the content of the working tape, we are at state q, and the head is at the symbol appearing after q (we should also include the input somewhere). We denote by c_{ij}, the content of the cell in the i^{th} row and the j^{th} column in the computation table of $M_L(x)$. Our **APSPACE** machine will be able to decide whether or not $c_{ij} = \sigma$ (then we can go over the last row and check if q_{acc} is written somewhere). Note that the index i can be written in polynomial number of bits. As in the Cook-Levin Theorem, the important fact is that the computation is local. The value of c_{ij} is determined by the values $c_{i-1,j-1}, c_{i-1,j}, c_{i-1,j+1}, c_{i-1,j+2}$, say by some known function g (depends on the transition function of M_L). $\forall \sigma \in \Sigma \cup \Gamma \cup Q : C_{ij} = \sigma \iff \exists \sigma_1, \sigma_2, \sigma_3, \sigma_4 : C_{i-1,j-1} = \sigma_1 \wedge C_{i-1,j} = \sigma_2 \wedge C_{i-1,j+1} = \sigma_3 \wedge C_{i-1,j+2} = \sigma_4 \wedge \sigma = g(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$.

To decide whether $c_{ij} = \sigma$, under existential quantifiers, non-deterministically guess $\sigma_1, \sigma_2, \sigma_3, \sigma_4$. Verify that $\sigma = g(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ if not reject. The next step is to verify recursively that $c_{i-1,j-1} = \sigma_1 \wedge c_{i-1,j} = \sigma_2 \wedge c_{i-1,j+1} = \sigma_3 \wedge c_{i-1,j+2} = \sigma_4$. Let us denote by S_i the space used by our machine, when it begins with the i^{th} configuration (i.e., verifying $c_{ij} = \sigma$ for some j,σ). Even if we use the same space for all four calls, we get $S_i = S_{i-1} + poly(|x|) = ... = 2^{|x|^c} poly(|x|)$. This happens because we keep a call stack, we donot overwrite the current frame. Under universal quantifiers, we write non-deterministically, $(J, \sigma_J) \in \{(j-1, \sigma_1), (j, \sigma_2), (j+1, \sigma_3), (j+2, \sigma_4)\}$. Suppose, our machine has four transition functions, we output "*yes*" if $c_{i-1,J} = \sigma_J$.

The universal quantifiers allowed us to compute the conjunction of four recursive calls simultaneously, thus we don't need to keep a call stack. In that case, in all times we need to keep a call stack. In that case, we need to remember i, j, σ which requires polynomial space, in recursive call for $c_{i-1,J}$ we use the same space that was used to store i.

Q. 2. Show that SUCCINCT SET-COVER $\in \Sigma_2^p$

Soln. In the SUCCINCT-SETCOVER problem [2], we are given a collection $S = \{\phi_1, \phi_2, ...\phi_m\}$ of $3-DNF$ formulae of n-variables, and an integer k. The task is to determine whether there is a subset $s' \subseteq \{1, 2, ...m\}$ of size at most k for which $\bigvee_{i \in s'} \phi_i$ is a tautology.

It is sufficient to provide the appropriate certificate definition for the language. Let $S = \{\phi_1, \phi_2, ...\phi_m\}$, k be an instance of SUCCINCT SET-COVER, over variables $U = \{u_1, u_2, ...u_m\}$. Let s_f denote the truth value of S resulting from the assignment $f : U \to 0, 1$. Note that a formula ϕ is a tautology if for all assignments f, $\phi_f = 1$. This immediately gives us the following certificate definition.

$< s, k > \in SUCCINCT - SETCOVER \iff \exists s' \subseteq [1, m] \forall f : U \to \{0, 1\}, |s'| \leq \ k \ and \ (V_{i \in s'} \phi_i) f = 1$, thus showing that SUCCINCT SET-COVER $\in \Sigma_2^p$.

Disclaimer: All the answers are collected from the internet, and are not the author's creation.

References

[1] Ariel (https://cs.stackexchange.com/users/27055/ariel). Proof of apspace = exp. Computer Science Stack Exchange. URL:https://cs.stackexchange.com/q/49647 (version: 2015-11-18).

[2] Somindu Chaya Ramanna. Tutorial 10: Polynomial hierarchy, 2019. https://cse.iitkgp.ac.in/~somindu/toc-2019/tutorial10-solutions.pdf last accessed June 25, 2020.