Analysis of Algorithms

Homework 6 – P vs NP

Abraham Murciano & Elad Harizy

January 13, 2021

Question 1

Part A

We are to prove that if the languages $\mathcal{L}_1$ and $\mathcal{L}_2$ are in P, meaning that there automata that can tell us whether a word is in the language or not in polynomial time ($O(n * k)$ for some constant k), then $\mathcal{L}_1 \cup \mathcal{L}_2 \in P$.

Since $\mathcal{L}_1$ and $\mathcal{L}_2$ can be decided in polynomial time, their union can also, as explained in Part B.

Part B

We are told that the languages $\mathcal{L}_1, \mathcal{L}_2$ can be decided in polynomial time using algorithms A_1, A_2 respectively, with running times $O(n^{k_1}), O(n^{k_2})$. To decide their union, one would have to decide each one individually, and decide their union based on their logical disjunction. Thus the complexity of deciding their union is $O(n^{k_1} + n^{k_2})$, or $O(n^{\max(k_1,k_2)})$, which is polynomial.

Question 2

Part A

We must prove that if $\mathcal{L} \in P$ then $\forall k \in \mathbb{N}, \mathcal{L}^k \in P$. Meaning that for any constant k, we can decide the concatenation of the language to itself k times, in polynomial time.

We will use a lemma which states that if $\mathcal{L}_1, \mathcal{L}_2 \in P$, then $\mathcal{L}_1\mathcal{L}_2 \in P$. (Proof omitted.)

We will prove this by induction.

For $k = 0, \mathcal{L}^k = \mathcal{L}^0 = \{\varepsilon\} \in P$.

Assume that for $k = n, \mathcal{L}^k = \mathcal{L}^n \in P$.

Then for $k = n + 1, \mathcal{L}^k = \mathcal{L}^{n+1} = \mathcal{L}^n \mathcal{L}$. However we know that both $\wedge$ and $\mathcal{L}^n$ are in P, so using our lemma, their concatenation, $\mathcal{L}^{n+1}$ must be in P.