Computability and Computational Complexity, A.Y. 2019–2020

Written exam

Mauro Brunato

Wednesday, February 12, 2020

Exercise 1

Consider the following language in $\{0,1\}^*$:

$$K = \{0^n 1^n : n \in \mathbb{N}\} = \{\varepsilon, 01, 0011, 000111, 00001111, 0000011111, \dots\}$$

i.e., all strings composed by a sequence of zeroes followed by the *same* number of ones.

1.1) Write a single-tape Turing Machine with alphabet $\Sigma = \{\textvisiblespace, 0, 1\}$ that recognizes K.

1.2) Prove or disprove the decidability of each of the following properties of TMs:

$$\begin{aligned}
\mathcal{P}_1 &= \{\mathcal{M} : \mathcal{M} \text{ decides } K\},\\
\mathcal{P}_2 &= \{\mathcal{M} : \mathcal{M} \text{ decides } K \text{ in less than 100 steps}\},\\
\mathcal{P}_3 &= \{\mathcal{M} : \mathcal{M} \text{ decides } K \cap \Sigma^{100} \text{ (i.e., strings in } K \text{ not longer that 100 symbols)}\}.
\end{aligned}$$

For 1.1 use any notation you like, and encode acceptance and rejection as you prefer (0/1 on tape, two different halting states, etc.).

Exercise 2

Prove that K, the language defined in Exercise 1, belongs to the complexity class **L**.

*While 1.1 required a single-tape machine, class **L** has a different assumption. Here, however, you are not asked to write down the TM: just a few lines of pseudocode will do.*

Exercise 3

Let $L_1, L_2 \in$ **NP**. Does $L_1 \cup L_2 \in$ **NP**? Does $L_1 \cap L_2 \in$ **NP**? Why?

*Be as formal as you can, e.g.: "Since $L_1 \in$ **NP**, then there is a TM $\mathcal{M}_1$ such that…"*

Exercise 4

Consider the following classical **NP**-complete languages:

$$\begin{aligned}
\text{CLIQUE} &= \{(G,k) : \text{Undirected graph } G \text{ has a completely connected subgraph of size } k\},\\
\text{INDSET} &= \{(G,k) : \text{Undirected graph } G \text{ has a completely disconnected subgraph of size } k\}.
\end{aligned}$$

4.1) Describe a polynomial-time reduction from one language to the other.

4.2) Show that CLIQUE $\cap$ INDSET $\neq \emptyset$.

For 4.1, choose the direction you like. In 4.2, don't be afraid of simple answers: to show that a set is not empty, you just need to find an element in it.