Computability and Computational Complexity, A.Y. 2018–2019

Written exam

Mauro Brunato

Wednesday, September 4, 2019

Exercise 1

Let Σ be the set of ASCII character. For each language $L \subseteq \Sigma^*$ listed below, tell whether it is computable or not, and why.

1. $L_1 =$ Encodings of Turing Machines that terminate on the empty input string
2. $L_2 =$ Encodings of Turing Machines that terminate within 100 steps on every input string
3. $L_3 =$ Binary representations of prime numbers
4. $L_4 =$ Encodings of Turing Machines that terminate within 100 steps on the empty input string

Assume that the TM encodings are given according to a predetermined set of specifications, i.e., for a given string it is always possible to say whether it is a TM encoding or not, and it is always possible to simulate the encoded TM for any finite number of steps on any input string.

Exercise 2

2.1) Define the classes **RP**, **PP** and **BPP**.

2.2) Put them in the correct inclusion order, if any, and explain why.

Exercise 3

Prove that **NL** $\subseteq$ **PSPACE**.

Answer outlines

Exercise 1

- L_1 is not computable because it is the language of TMs that terminate on the empty input string. As we know, deciding this language is equivalent to solving HALT, therefore it is not computable.
- L_4 is clearly computable, because we just need to emulate the TM for at most 100 steps on the empty input.
- L_2 is a "complication" of L_4, since the TM is required to halt within 100 steps for every input. However, the number of input strings the TM must be emulated on is bounded, because only the first 100 visited symbols are relevant to our analysis. Therefore, L_2 is computable.
- L_3 is clearly computable, as it is possible to write a computer program (and hence a TM) that decides whether a number is prime or not.

Observations

Note that L_1, L_2 and L_4 can be rephrased as properties of TMs; however, such properties are not semantic because they do not refer to the recognized language (respectively, "the TM halts", "the TM halts within 100 steps on all input strings", "the TM halts within 100 steps on the empty string"), therefore Rice's theorem is never applicable.
On the same note, L_3 does not refer to TMs having specific properties, but to numbers. Therefore, Rice's Theorem is not applicable. It would be applicable if L_3 were the language of TMs that decide prime numbers, not the language of prime numbers itself.

Exercise 2

See the course notes.

Observations

Remember that, unlike **BPP** and **PP**, the definition of **RP** is *not* symmetric: since **RP** $\subseteq$ **NP**, whenever $x \notin L$, *every* computation must reject.

Exercise 3

Since $\log n = O(n^c)$ (i.e., logarithms are dominated by polynomials), we know that **NL** $\subseteq$ **NPSPACE**. But we know that (as a consequence of Savitch's Theorem) **NPSPACE** $=$ **PSPACE**, hence the thesis.

As an alternative, observe that every computation of a NDTM $\mathcal{N}$ deciding a **NL** language occupies at most $c \log n$ cells of the read/write tape (where n is the input size); therefore, it can perform at most $O(n^c)$ steps (give or take some constants due to the number of states and to the head positions). Therefore, we can sequentially emulate all the computations of $\mathcal{N}$ with a DTM that uses the original logarithmic space **plus** a $O(n^c)$-bit (i.e., polynomial) string that keeps track of the current sequence of non-deterministic choices.