Schedule S1, (Computer Science, CS and Philosophy, Maths and CS)
Schedule B, MSc in Computer Science

Hilary Term 2019

Computational Complexity

Exercise class 1: background material, TMs, (un)decidability

1. Arrange the following functions in increasing order of order-of-growth. (Do any of them have the same order of growth?)

   (a.) n^{10} (b.) $(n^2)^{\log n}$ (c.) $(1.1)^n$ (d.) $(\log n)^{\log n}$

   (e.) $n^{\log n}$ (f.) $100n^2$

2. A *Turing machine with two-side unbounded tapes* is a Turing acceptor where the tapes are unbounded to both sides. Show that such machines can be simulated by our standard model of Turing machines.

   Note: You do not have to give the formal definition of the Turing machine. A precise description of what the machine does and how it simulates the original machine is sufficient.

3. Prove that if L_1 and L_2 are decidable languages over the alphabet Σ, then so are

   (a) $L_1 \cup L_2$

   (b) $L_1 \cap L_2$

   (c) $\bar{L}_1 := \{w \in \Sigma^* : w \notin L_1\}$

   (d) $L_1; L_2 := \{wv \in \Sigma^* : w \in L_1, v \in L_2\}$

   (e) $L_1^* := \{w \in \Sigma^* : w = w_1 w_2 w_3 \dots w_k, \quad w_i \in L_1 \text{ for all } 1 \leq i \leq k\}$

   (This result can be summarised by the statement that the class of decidable languages is closed under *union*, *intersection*, *complement*, *concatenation*, and *star*)

4. Show that the existence of an algorithm to decide HALT would imply the existence of an algorithm for deciding any recursively enumerable language.
   (This result may be summarised by the statement that HALT is *complete* for the class of recursively enumerable languages.)

5. Show that Emptiness is undecidable.
   Emptiness is the problem to decide for a given Turing machine if it rejects all inputs.

6. A polynomially bounded Turing machine is one which, on input w, with $|w| = n$, uses no more than $f(n)$ cells on its tape, for f a polynomial function. Is halting decidable for polynomially bounded Turing machines? If no, explain why not, if yes, explain how to decide if a polynomially bounded Turing machine M halts on input w.

7. Suppose the decision version of the Clique problem

CLIQUE	
Input:	Graph G, $k \in \mathbb{N}$
Problem:	Does G have a clique of size $\geq k$?

can be solved in time $T(n)$ for some function $T : \mathbb{N} \to \mathbb{N}$ with $T(n) \geq n$. Prove that the optimisation version

OPT-CLIQUE	
Input:	Graph G
Problem:	Compute a clique in G of maximum order

can be solved in time $O(n^c \cdot T(n))$, for some $c \in \mathbb{N}$.