Exercises for Computability and Complexity, Spring 2018, Sheet 1

Note: time and location of tutorial session, where this sheet will be discussed, will be announced.

Exercise 1 Give a transition table for a TM that computes the function $f(n) = 2n$. The TM should have the tape alphabet $\{0, 1, \triangleright, \sqcup\}$ and numbers are coded as binary strings by writing them to base 2.

Solution. That's an easy one. Multiplying n by 2 means to append a 0 at the binary representation of n. A table for such a TM:

$p \in K$	$\sigma \in \Sigma$	$\delta(q, \sigma)$	comment
s	$\triangleright$	$(s, \triangleright, \rightarrow)$	get started
s	0	$(s , 0, \rightarrow)$	reading a 0, just move on to the right
s	1	$(s, 1, \rightarrow)$	reading a 1, just move on to the right
s	$\sqcup$	$(h, 1, -)$	hitting the first blank, replace it by 1, halt

Exercise 2 If one would admit TMs with countably many states, would this extend the set of TM-computable functions on the integers? In other words, is there a function $f: \mathbb{N} \rightarrow \mathbb{N}$ which can be computed by some TM with countably infinitely many states, but not by any ordinary TM? Sketch a proof for your answer.

Solution. With infinitely many states one can indeed "compute" more functions than with finitely many states. (In fact, with such a machine one could "compute" *every* function on the integers.) To see why, let $f: \mathbb{N} \rightarrow \{0, 1\}$ be *any* function with binary values on the integers (that is, f picks a subset of the integers – and any subset can be thus picked by some such f – that is, there must be uncountably many such f, which in turn means that almost all of these f are not Turing-computable). Arrange an infinite-state TM M with state set $K \supseteq \{s_1, s_2, \ldots\}$ such that on input n, M first goes to s_n (how can this be done? needs a subroutine) and then outputs $f(n)$ due to a hardwired answer-table-lookup rule of the form $\delta(s_n, a) = (h, f(n), -)$.

Exercise 3 (deferred to Exercise sheet 2)

Exercise 4 Show that $L = \{w \in \{1\}^* \mid |w| \text{ is a power of } 2\} \in \mathbf{TIME}(O(n \log n))$, by describing in words (and maybe sketches of interesting configurations) a TM (with possibly several tapes) that does this job.

Solution. Set up a 2-tape TM, as follows. The first tape contains the input word, is read-only, and the cursor here never moves left. While the first cursor moves right, on the second tape a binary-coded count of the number of 1's visited is constructed. Whenever the first cursor moves to the right, the count on tape 2 is updated (which may take some operations where the first cursor does not move). The update is a combination of the add-1 and shift-right, single-tape TMs from the lecture notes, which per add-1 operation may require 2 full back-and-forth traversals of the word b written on tape 2 up to that point, that is, 4 |b| TM cycles. When the last 1 on tape 1 has been processed, our TM enters a final round of checking whether the 2nd tape word b is of the form 10....0. If yes, the input is accepted, if no, not. This final check can

be clearly effected in another $|b|$ steps. Since $|b| \leq \log_2(|w|)$, we find that our TM uses at most $\log_2(|w|)(4\ n) + \log_2(|w|) = O(n \log n)$ steps.