Theory of Computer Science

G. Röger — University of Basel
Spring Term 2020 — Computer Science

Exercise Sheet 9

Due: Wednesday, April 29, 2020

Exercise 9.1 (Turing machines; 2 marks)

We defined the tape of a Turing machine to be infinite in both directions. An alternative definition uses a tape that is only infinite in one direction. Formally, this can be achieved by changing the definition of a *step* and leaving all other definitions the same: in the definition on slide C7.16, we change the third case from

$$\langle \varepsilon, q, b_1 \dots b_n \rangle \vdash_M \langle \varepsilon, q', \Box c b_2 \dots b_n \rangle \qquad \text{if } \langle q', c, L \rangle \in \delta(q, b_1), n \geq 1$$

to

$$\langle \varepsilon, q, b_1 \dots b_n \rangle \vdash_M \langle \varepsilon, q', c b_2 \dots b_n \rangle \qquad \text{if } \langle q', c, L \rangle \in \delta(q, b_1), n \geq 1.$$

Turing machines with this step model behave in the way as our Turing machines except if they try to move the head to the left of the first position. In this case the head just remains on the first position.

Using a doubly infinite tape does not make our Turing machines more expressive than machines that use a tape that is only infinite in one direction. Explain how a given Turing machine $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \Box, E \rangle$ with a two-sided infinite tape can be transformed to a machine M' with single-sided infinite tape, such that M' accepts the same language as M.

Note: A proof sketch is sufficient here, but you should explain what kind of additional symbols and states M' requires and what the main idea of the transformation is.

Exercise 9.2 (Multiplication is Turing-computable; 2 marks)

Describe the main idea of a proof showing that multiplication of binary numbers is Turing-computable, i.e., describe a Turing machine that computes mul^{code} for the function $mul : \mathbb{N}_0^2 \to_{\text{p}} \mathbb{N}_0$ with $mul(n_1, n_2) = n_1 \cdot n_2$.

Note: You may use the fact that addition is Turing computable and a high-level description of the Turing machine is sufficient.

Exercise 9.3 (Composition of computable functions is computable; 2 marks)

Let $f : \Sigma^* \to \Sigma^*$ and $g : \Sigma^* \to \Sigma^*$ be Turing-computable partial functions for an alphabet Σ. Show that the *composition* $(f \circ g) : \Sigma^* \to \Sigma^*$ is also Turing-computable.

In general the composition of two functions is defined as $(f \circ g)(x) = f(g(x))$. Specifically, the value $(f \circ g)(x)$ is undefined if $g(x)$ is undefined.

Exercise 9.4 (Enumerable Functions; 2 marks)

Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Specify total and computable functions $f : \mathbb{N}_0 \to \Sigma^*$ which recursively enumerate the following languages. Additionally specify the function values $f(0)$, $f(1)$, $\dots$, $f(5)$. You may use all computable functions which were discussed in the lecture.

(a) $L_1 = \{\mathtt{b}^n \mathtt{a}^{2n} \mid n \in \mathbb{N}_0, n \text{ is even}\}$

(b) $L_2 = \{w \in \Sigma^* \mid \mathtt{a} \text{ occurs in } w \text{ exactly once}\}$

Exercise 9.5 (Decidability; 2 marks)

Which of the following statements are true? Justify your answers with one or two sentences each.

(a) If L is decidable, then L is also finite.

(b) If L is regular then L is also decidable.

(c) If L_1 and L_2 are decidable, then $L_1 \cap L_2$ is decidable.

(d) If L is context-sensitive, then L is also semi-decidable.