TU Dresden, Fakultät Informatik
Daniel Borchmann, Markus Krötzsch Winter Term 2015/2016

Complexity Theory

Exercise 1: Mathematica Foundations, and Decidability and Recognisability

17 October 2017

Exercise 1.1. Let M be a set. Show that there is no function

$$f\colon M \to 2^M$$

such that f is surjective.

Solution. Consider the set D defined by

$$x \in D \iff x \notin f(x).$$

If now f were surjective, then there would exist some x' such that $f(x') = D$. But then

$$x' \in D \iff x' \notin f(x') = D,$$

a contradiction. (One can also draw a nice diagram here, akin to the classical diagram for proving the uncountability of $\mathbb{R}$.)

Exercise 1.2. Show the following claims.

1. $|\mathbb{N}| = |\mathbb{N} \times \mathbb{N}|$.
2. $|\mathbb{N}| = |\mathbb{Q}|$.
3. $|\mathbb{N}| \neq |\mathbb{R}|$.

Solution.

1. Make a grid of all pairs of natural numbers and traverse in diagonals starting from the corner.
2. Same, but this time the grid consists of all fractions instead of pairs of natural numbers.
3. See other exercise: a subset S of $\mathbb{N}$ can be interpreted as the positions of the ternary expansion of a number in $[0, 1]$ such that there is a 1 at position i if $i \in S$ and 0 otherwise (and never a 2). Thus
$$|\mathbb{N}| < |2^{\mathbb{N}}| \leq |[0,1]| \leq |\mathbb{R}|.$$

Exercise 1.3. Show the following claims.

1. There exist non-regular languages.
2. There exist undecidable languages.
3. There exist non-Turing-recognizable languages.

Solution. The argument is always the same: over a fixed alphabet there are only countably many automata and Turing machines with that input alphabet, but uncountably many languages. Therefore, most of the languages cannot be regular/decidable/recognizable.

Exercise 1.4. Show that every simple undirected graph with two or more nodes contains two nodes that have equal degrees.

Solution. Consider a graph with n nodes (recall: graphs are not directed here and do not contain any loops). Then each node can have degree $0, 1, \dots, n-1$. If some node has degree 0, then no node can have degree $n-1$. If some node has degree $n-1$, then no node can have degree 0. Thus, either all nodes have degree $0, \dots, n-2$ or $1, \dots, n-1$. In either case, there are only $n-1$ possible values of a degree for an overall of n nodes. Therefore, two nodes must have the same degree.

Exercise 1.5. Let $A = \{ s \}$, where

$$s := \begin{cases} 0 & \text{if life will never be found on Mars,} \\ 1 & \text{if life will be found on Mars someday.} \end{cases}$$

Is A decidable? (For the purpose of this problem, assume that the question whether life will be found on Mars has an unambiguous "yes" or "no" answer.)

Solution. Either $A = \{ 0 \}$ or $A = \{ 1 \}$, and both sets are decidable. However, we cannot decide which case is true (at least not now). But this does not matter: to show that A is decidable it is sufficient to show that there *exists* a Turing machine deciding it. The way this Turing machine is found does not need to be constructive. (See also the proof of the theorem saying that decidable sets are exactly those that can be enumerated by a Turing machine in non-decreasing length of the output words).

Exercise 1.6. Show that the class of Turing-decidable languages is closed under

(a) union,

(b) concatenation,

(c) intersection,

(d) star.

Solution. To decide the union of two decidable sets we just run the corresponding deciders in parallel and accept if any of those accepts, and reject if both reject. Intersection works the same way, but this time we accept only if both deciders accept. For concatenation we just guess a separation point and check the corresponding subwords for membership. The same we can do for star.

$*$ **Exercise 1.7.** Show that the class of Turing-recognizable languages is closed under homomorphism.

Solution. To see closure under homomorphisms let $f\colon \Sigma^* \to \Delta^*$ be a homomorphism and let $L \subseteq \Sigma^*$ be some Turing-recognizable set. Then L can be enumerated by some Turing machine M, and if we apply f to each output word of M we obtain an enumerator for $f(L)$ (note that homomorphism are always computable).

Exercise 1.8. A *Turing machine with two-sided unbounded tape* is a single-tape Turing machine where the tape is unbounded on both sides. Argue that such machines can be simulated by ordinary Turing machines.

Solution. We only sketch the idea (this is why the exercise says "argue" instead of "prove"). Suppose we are given a Turing machine M with a two-sided unbounded tape. Assume the cells are numbered with elements from $\mathbb{Z}$. We can simulate M with an ordinary Turing machine M' that "wraps around" the tape of M at cell 0. M' then stores in every cell $j \in \mathbb{N}$ of its own tape the two cells j and $-(j+1)$ of M. Then M' can simulate one step of M in just one step, keeping track of whether to read the first or second part of the cell content, corresponding to whether M's head is at the right or left of cell 0.

$*$ **Exercise 1.9.** Show that single-tape Turing machines that cannot write on the portion of the tape containing the input string recognize only regular languages.

Solution. This is a rather difficult exercise and is not meant to be presented in class. Instead, it is meant for very good students to be kept interested in the exercises. The solution presented here is adapted from the solution of Exercise 3.17 of the Instructors Manual of Sipser's Book.

We first observe that a trivial approach that rejects the claim does not work: we cannot copy the input to another tape and work on this copy as usual. The problem here is that for copying the input, we need to keep track of what the last symbol is that we have copied so far. For this we would need to mark this position, which is not possible. (Note that, however, if we would have another tape, then we could copy the input, showing that the above claim is indeed false for multi-tape Turing machines.)

This failed approach however contains the first clue to the solution: when we have read the input, we can only remember a finite amount of information about the input, using a state of the Turing machine M. We put this into a real solution by constructing a finite automaton A that accepts the same language as M. For this we need to construct the state space, the transition function, and the final state of A.

For the state space of A we define for some string s the function

$$F_s\colon (Q \cup \{\,\mathsf{first}\,\}) \to (Q \cup \{\,\mathsf{accept}, \mathsf{reject}\,\}).$$

This function is determined by the way M works on input s. $F_s(\mathsf{first})$ is the state M reaches when M has been started on input s from the beginning and is about to move off of the right end of s for the first time. If M accepts or rejects (either explicitly or by looping) before leaving the input s, then $F_s(\mathsf{first})$ is accept or rejects, respectively. Furthermore, for state $q \in Q$, $F_s(q)$ is the state the machine M enters when it is about to move off of the end of s for the first time when it was started on the right end of s in state q. Again $F_s(q)$ is accept or reject if M accepts or rejects before leaving the input s.

There are only finitely many functions from $Q \cup \{\,\mathsf{first}\,\}$ to $Q \cup \{\,\mathsf{accept}, \mathsf{reject}\,\}$, and thus there are only finitely many functions F_s. Then M behaves the same on inputs s and t if $F_s = F_t$. In particular, if $F_s = F_t$, then M accepts s if and only if M accepts t.

The state space of A is now the set of all such functions F_s. This state space is finite, by the argument of the previous paragraph. Upon reading a symbol a while in state F_s, the machine A changes its state to F_{sa}. The initial state of A is F_ε, which is given by $F_\varepsilon(\mathsf{first}) = q_0$ (the initial state of M) and $F_\varepsilon(q) = q$ for all $q \in Q$. The final states of A are all those functions F_s where s is a string accepted by M.

Exercise 1.10. Let $\mathsf{ALL}_{\mathsf{DFA}} = \{\, \langle A \rangle \mid A \text{ is a DFA that accepts every word} \,\}$. Show that $\mathsf{ALL}_{\mathsf{DFA}}$ is decidable.

Solution. Given a DFA, we can obtain a DFA for the complementary language by interchanging final and non-final states. For this complementary DFA we can check whether the language it accepts is empty. If so, the original DFA accepted every word. If not, then not.

Exercise 1.11. Let $\mathsf{E}_{\mathsf{TM}} = \{\, \langle M \rangle \mid M \text{ is a TM such that } \mathcal{L}(M) = \emptyset \,\}$. Show that $\overline{\mathsf{E}_{\mathsf{TM}}}$ is Turing-recognizable.

Solution. Given a Turing machine M we can recognize whether the language accepted by M is empty by just iterating over all $(i, j) \in \mathbb{N} \times \mathbb{N}$ and running M for i steps on the j-th input word. If $\mathcal{L}(M) \neq \emptyset$, then we will eventually find some word w_j that is accepted in i steps, and we accept. If $\mathcal{L}(M) = \emptyset$, then this computation will loop and we will not accept.

Exercise 1.12. Let C be a language. Prove that C is Turing-recognizable if and only if a decidable language D exists such that $C = \{\, x \mid \exists y. \langle x, y \rangle \in D \,\}$.

Solution. Suppose such a language D exists. Then we can obtain an enumerator for C by enumerating all elements of D and dropping the second component y. Thus C can be enumerated and must be Turing-recognizable.

Conversely assume that C is Turing-recognizable. Then there exists a Turing machine M recognizing C. Define D to be the set of all pairs of (x, y) where $x \in C$ and y is an accepting computation history of M on input x. Then D is decidable and satisfies the requirements of the claim.