Theory of Computer Science

G. Röger
Spring Term 2019

University of Basel
Computer Science

Exercise Sheet 7 — Solutions

Exercise 7.1 (Composition of computable functions is computable; 2 points)

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.

Solution:

General Ideas:
If f and g are Turing-computable, then there exist DTMs M_f and M_g which compute $f(y)$ given input y, respectively $g(x)$ given input x.
If we start the DTM M_g on an input x where $g(x)$ is defined, then after termination the tape content and the reading head are exactly in the configuration needed for the input for M_f (according to the definition of functions computed by a DTM). Thus it is in this case sufficient to combine M_f and M_g to a new DTM by changing all transitions of M_g which lead to an end state to the start state of M_f. Thus the DTM first computes $g(x)$ given input x. This is then the input y for f, which means the DTM computes $f(y) = f(g(x)) = (f \circ g)(x)$.
Technical Details:
We assume without restriction that the sets of states of M_f and M_g are disjoint (if not, we can rename states as needed) and that the tape alphabets are identical otherwise we can unite the two tape alphabets and arbitrarily choose all new transitions which need to be defined now (for example when M_g reads a symbol only occurring in the tape alphabet of M_f). We can do this arbitrarily because these transitions will never be used.
One technical problem arises though if M_g terminates in a configuration that does not represent a valid computation (if the head is on a wrong position or illegal symbols are on the tape). In this case the machine should not return a valid result, since $(f \circ g)(x)$ is undefined, but we cannot guarantee this without further demands to M_f.
The easiest way to remove this problem is to modify M_g in a way to ensure that it cannot stop in an invalid configuration. For example we could check at the end of the computation of M_g if the tape content has a correct form. One difficulty we have to deal with is to recognize where the visited part of the tape starts and ends. The easiest way to solve this is to never use $\square$ during the calculation of M_g but instead write a new symbol $\hat{\square}$ which behaves exactly as $\square$. Thus we ensure that when checking the result of the computation $\square$ marks both ends of the visited tape. During the final check we then replace $\hat{\square}$ with $\square$.

Exercise 7.2 (Enumerable Functions; 1.5+1.5 Points)

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Specify total and computable functions $f : \mathbb{N}_0 \to \Sigma^*$ which recursively enumerate the following languages.

(a) $L_1 = \{\mathtt{a}^x\mathtt{b}^y \mid x, y \in \mathbb{N}_0\}$

(b) $L_2 = L_A \cup L_B$ where L_A and L_B are languages over Σ that are recursively enumerated by the functions f_A and f_B.

Solution:

(a) $f_{L_1}(n) = \mathtt{a}^{decode_1(n)}\mathtt{b}^{decode_2(n)}$

(b) $f_{L_2}(n) = \begin{cases} f_A(\frac{n}{2}) & \text{if } n \text{ is even} \\ f_B(\frac{n-1}{2}) & \text{otherwise} \end{cases}$

Exercise 7.3 (Decidability and Semi-Decidability; 0.5+0.5+1+1+1 Points)
Which of the following statements are true, which are false? In each case, specify a short proof (1 sentence) or a counter example. You can use all results from the lecture.

(a) Every decidable language is of type 0.

Solution:

Correct. From the lecture, we know: if A is decidable then A and $\bar{A}$ are semi-decidable and thus of type 0.

(b) If A is decidable then $\bar{A}$ is also decidable.

Solution:

Correct. The characteristic function of $\bar{A}$ is $\chi_{\bar{A}}(w) = 1 - \chi_A(w)$. This function is computable if χ_A is computable.

(c) Every language that is accepted by a Turing machine is decidable.

Solution:

Incorrect. If a language is accepted by a Turing machine, it is semi-decidable but not necessarily decidable. We had several examples in the lecture for languages that are semi-decidable but not decidable, e.g. the special halting problem.

(d) Every language that can be described by a regular expression is decidable.

Solution:

Correct. For every regular language there exists a DFA that accepts exactly this language. It is easy to construct a program for a given DFA that simulates the DFA on a given input and returns `1` or `0` at the end, depending on whether the automaton ends in an accepting state or not.

(e) Every decidable language is context-free.

Solution:

Incorrect. For example, the language $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbb{N}\}$ is decidable (the Turing machine from the exercise in C7 was deterministic and can easily be modified to do this), but we know from the lecture (C6) that it is not context-free.

Additional information:
In general, we have the following connections:

- Every type-3 language (regular language) is a type-2 language (context-free language), but the reverse is not true in general.
- Every type-2 language (context-free language) is a type-1 language (context-sensitive language), but the reverse is not true in general.
- Every type-1 language is decidable, but not every decidable language is of type 1. (In contrast to the other statements, this was not shown in the lecture.)
- Every decidable language is semi-decidable (a type-0 language), but the reverse is not true in general.