Theory of Computer Science

G. Röger
F. Pommerening
Spring Term 2020

University of Basel
Computer Science

Exercise Sheet 10 — Solutions

Exercise 10.1 (Transitivity of Reductions; 2 marks)

Show for any languages A, B and C: if $A \leq B$ and $B \leq C$, then $A \leq C$.

Solution:

Let $A \subseteq \Sigma_A^*$, $B \subseteq \Sigma_B^*$ and $C \subseteq \Sigma_C^*$. Since $A \leq B$, there is a total and computable function $f : \Sigma_A^* \to \Sigma_B^*$, with $x \in A$ iff $f(x) \in B$. From $B \leq C$ we analogously know that there is a function $g : \Sigma_B^* \to \Sigma_C^*$ with $x \in B$ iff $g(x) \in C$.
We define $h : \Sigma_A^* \to \Sigma_C^*$ as $g \circ f$ (i.e. $h(x) = g(f(x))$ for all $x \in \Sigma_A^*$). The function is total and computable, since the composition of total and computable functions is also total and computable. We now know that $x \in A$ iff $f(x) \in B$ iff $g(f(x)) \in C$ iff $h(x) \in C$. We conclude that A can be reduced (with h) to C: $A \leq C$.

Exercise 10.2 (Undecidability; 3 marks)

In each part, give an example of a language L_i with the given properties (without justification), or explain why no such language exists (with a short explanation).

(a) L_1 is undecidable and L_1 and $\overline{L_1}$ are semi-decidable.

(b) L_2 is a type-0 language and decidable.

(c) L_3 is a type-0 language and undecidable.

Solution:

(a) Impossible: if a language and its complement are semi-decidable, the language is decidable. We can for example interleave the computation steps of the semi-decision procedures (called "dove-tailing").

(b) For example any regular language such as $\{\mathtt{a}^n\mathtt{b}^m \mid n, m \geq 0\}$.

(c) Any semi-decidable (and thus type-0) but undecidable language, e.g. the halting problem.

Exercise 10.3 (Rice's Theorem; 2 marks)

For which of the following languages does Rice's theorem show that the language is undecidable? For each language where Rice's theorem can be used, specify the subset of Turing-computable functions $\mathcal{S}$ for which you use the theorem.
Hint: You do not have to write down any proofs. If Rice's theorem is applicable, specify the set $\mathcal{S}$, otherwise give a short reason (1 sentence) why Rice's theorem is not applicable.

(a) $L = \{w \in \{0,1\}^* \mid M_w \text{ does not terminate with a valid output for any input } \}$

(b) $L = \{w \in \{0,1\}^* \mid M_w \text{ computes the successor function or the predecessor function } \}$

(c) $L = \{w \in \{0,1\}^* \mid M_w \text{ requires an even number of steps on the input } \mathtt{0011} \}$

(d) $L = \{w \in \{0,1\}^* \mid \text{No input of } M_w \text{ leads to a valid output containing } \mathtt{0} \}$

Solution:

(a) Rice's theorem is applicable with the set of functions $\mathcal{S} = \{\Omega\}$, where Ω is the function that is undefined for all inputs.

(b) Rice's theorem is applicable with the set of functions $\mathcal{S} = \{succ, pred\}$.

(c) Rice's theorem is not directly applicable since the number of steps is a property of the computation and not a property of the computed function.

(d) Rice's theorem is applicable with the following set of functions:

$$\mathcal{S} = \{f \in \mathcal{R} \mid \text{for all } x \in \{0,1\}^*\colon\ f(x) \text{ is either undefined or does not contain } \texttt{0}\}$$

Exercise 10.4 (Undecidable Grammar Problems; 1.5+1.5 marks)

The *emptiness problem*, the *equivalence problem* and the *intersection problem* for general grammars are defined as:

- EMPTINESS: Given a general grammar G, is $\mathcal{L}(G) = \emptyset$?
- EQUIVALENCE: Given two general grammars G_1 and G_2, is $\mathcal{L}(G_1) = \mathcal{L}(G_2)$?
- INTERSECTION: Given two general grammars G_1 and G_2, is $\mathcal{L}(G_1) \cap \mathcal{L}(G_2) = \emptyset$?

You can use without proof that EMPTINESS is undecidable. (As a bonus exercise, you can also prove this statement with Rice's theorem.)

(a) Show that EQUIVALENCE is undecidable, by reducing EMPTINESS to it.

Solution:

Let $G_\emptyset$ be any grammar with $\mathcal{L}(G_\emptyset) = \emptyset$, e.g. a grammar without rules. Let f be the function $f(G) = (G, G_\emptyset)$ for all G.

$$\begin{aligned} G \in \text{EMPTINESS} &\text{ iff } \mathcal{L}(G) = \emptyset \\ &\text{ iff } \mathcal{L}(G) = \mathcal{L}(G_\emptyset) \\ &\text{ iff } (G, G_\emptyset) \in \text{EQUIVALENCE} \\ &\text{ iff } f(G) \in \text{EQUIVALENCE} \end{aligned}$$

The function f is total and computable and reduces EMPTINESS to EQUIVALENCE. Since EMPTINESS is undecidable, EQUIVALENCE must be undecidable as well.

(b) Show that INTERSECTION is undecidable, by reducing EMPTINESS to it.

Solution:

For an alphabet Σ let G_{Σ^*} be a grammar with $\mathcal{L}(G_{\Sigma^*}) = \Sigma^*$. Let f be the function $f(G) = (G, G_{\Sigma^*})$ for all G and Σ where G is a grammar with the alphabet Σ.

$$\begin{aligned} G \in \text{EMPTINESS} &\text{ gdw. } \mathcal{L}(G) = \emptyset \\ &\text{ gdw. } \mathcal{L}(G) \cap \Sigma^* = \emptyset \\ &\text{ gdw. } (G, G_{\Sigma^*}) \in \text{INTERSECTION} \\ &\text{ gdw. } f(G) \in \text{INTERSECTION} \end{aligned}$$

The function f is total and computable and reduces EMPTINESS to INTERSECTION. Since EMPTINESS is undecidable, INTERSECTION must be undecidable as well.