ETH zürich
Departement Informatik

Theoretische Informatik

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theoInf20

Exemplary Solutions – Sheet 8

Zürich, November 13, 2020

Solution to Exercise 20

(a) From two Turing machines M and M_c that accept L and $L^{\complement}$, respectively, we construct a new Turing machine A that decides L. The machine A simulates, for every input $w \in \Sigma^*$, the two machines M and M_c on w in parallel. If $w \in L$, then the machine M accepts the word w in finite time and then A accepts as well. If $w \notin L$, then M_c accepts the word w in finite time and then A rejects the input. In both cases, A outputs the correct answer in a finite number of steps.

(b) To show that $(L_{\mathrm{U},\lambda})^{\complement}$ is not recursively enumerable, we show $L_{\mathrm{U},\lambda} \notin \mathcal{L}_{\mathrm{R}}$ and $L_{\mathrm{U},\lambda} \in \mathcal{L}_{\mathrm{RE}}$. The statement to be proved then follows immediately using the claim from part (a).

The language $L_{\mathrm{U},\lambda}$ is a semantically nontrivial decision problem about Turing machines, since it is clearly nonempty, does not contain the codes of all Turing machines, and any two Turing machines A and B that accept the same language satisfy $\mathrm{Kod}(A) \in L_{\mathrm{U},\lambda} \iff \mathrm{Kod}(B) \in L_{\mathrm{U},\lambda}$. Rice's Theorem now immediately implies $L_{\mathrm{U},\lambda} \notin \mathcal{L}_{\mathrm{R}}$.

To show that $L_{\mathrm{U},\lambda}$ is recursively enumerable, we construct a Turing machine A that accepts $L_{\mathrm{U},\lambda}$. The machine A first checks if its input is valid, i.e., $\mathrm{Kod}(M)$ for some Turing machine M. If not, then A rejects its input. Otherwise, A simulates the computation of M on λ. If M halts on λ in the accepting state, then A accepts as well. If M rejects the empty word, then A rejects as well. If M does not terminate on λ, then A does not terminate either. Hence, A accepts the language $L_{\mathrm{U},\lambda}$.

Solution to Exercise 21

(a) To show $L_{\mathrm{H}} \leq_{\mathrm{R}} (L_{\mathrm{diag}})^{\complement}$, we assume that $M_{(L_{\mathrm{diag}})^{\complement}}$ is a TM deciding $(L_{\mathrm{diag}})^{\complement}$ and use it to construct a TM $M_{L_{\mathrm{H}}}$ deciding L_{H}.

A properly encoded input for the halting problem L_{H} contains two components, the code of a TM M and an input word x. From this we construct a machine $N_{M,x}$ that proceeds on *every* input as follows:

$N_{M,x}$ ignores its own input and simulates the computation of a TM M' on x. The TM M' is equal to M except that all transitions that lead to q_{reject} in M lead to q_{accept} in M'.

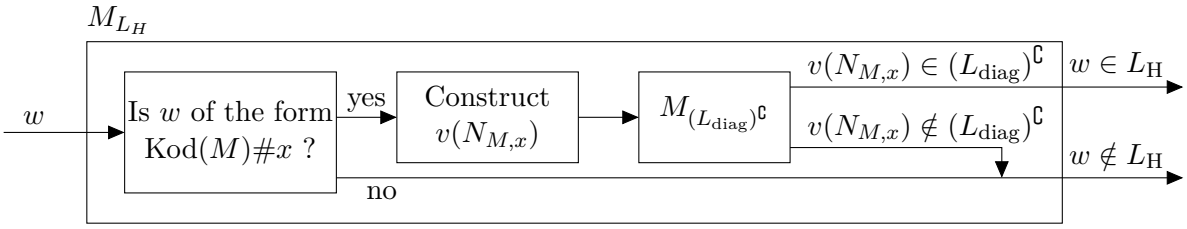
From $N_{M,x}$, we obtain the word $v(N_{M,x})$ that determines the index of $N_{M,x}$, i.e., $v(N_{M,x}) = w_i$ such that $N_{M,x} = M_i$ holds in the enumeration from the definition of L_{diag}. Then all properly encoded inputs $\text{Kod}(M)\#x$ clearly satisfy

$$\begin{aligned}
\text{Kod}(M)\#x \in L_{\text{H}} &\iff M \text{ halts on } x \\
&\iff M' \text{ reaches } q_{\text{accept}} \text{ on } x \\
&\iff N_{M,x} \text{ accepts every input} \\
&\iff N_{M,x} \text{ accepts } v(N_{M,x}) \\
&\iff v(N_{M,x}) \in (L_{\text{diag}})^{\complement}.
\end{aligned}$$

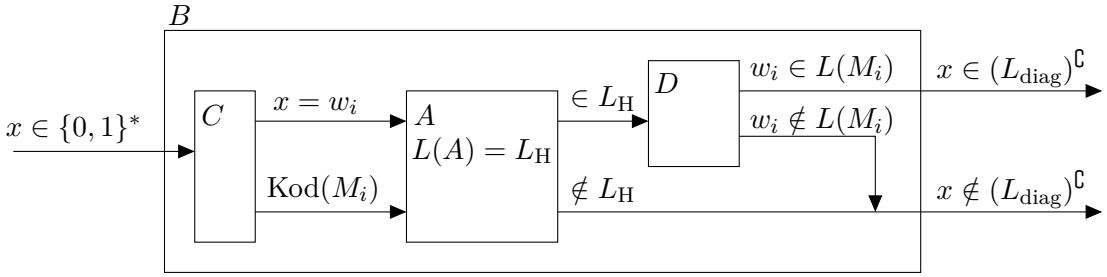
Hence, answering the question whether $v(N_{M,x}) \in (L_{\text{diag}})^{\complement}$ yields the desired result.

If the input w is not properly encoded, i.e., if it does not have the form $\text{Kod}(M)\#x$, we output "No". This is the desired result since $w \notin L_H$ holds in this case.

The reduction can be represented by the following diagram:



(b) To show $(L_{\text{diag}})^{\complement} \leq_{\text{R}} L_{\text{H}}$, we assume that A is an algorithm deciding L_{H}. Then we construct an algorithm B that uses A to decide the language $(L_{\text{diag}})^{\complement}$. The structure of the algorithm B is depicted in the following diagram:



On an input $x \in \Sigma^*_{\text{bool}}$, the subprogram C computes an $i \in \mathbb{N}$ with $x = w_i$ and the code $\text{Kod}(M_i)$ of the i-th TM. The subprogram A for L_{H} takes $\text{Kod}(M_i)\#x = w_i$ as input.

If A rejects the input $\text{Kod}(M_i)\#x$, then M_i does not halt on w_i, i.e., M_i does not accept the word w_i either. Hence, $w_i \notin (L_{\text{diag}})^{\complement}$ and B rejects its input $x = w_i$.

If A accepts the input $\text{Kod}(M_i)\#x$, then M_i halts on w_i. In this case, the subprogram D simulates the computation of M_i on w_i. This simulation is guaranteed to terminate in finite time. If the simulation yields that M_i accepts the word w_i, then $w_i \in (L_{\text{diag}})^{\complement}$ and B accepts its input $x = w_i$. Otherwise, M_i rejects the word w_i, hence, $w_i \notin (L_{\text{diag}})^{\complement}$ and B rejects its input $x = w_i$.

Solution to Exercise 22

We show $L_{\text{reach}} \notin \mathcal{L}_{\text{R}}$ by proving $L_{\text{U}} \leq_{\text{m}} L_{\text{reach}}$. To this end, we construct a TM A that transforms an input for L_{U} into an input for L_{reach}.

The TM A proceeds as follows:

- A checks if the input x has the form $\text{Kod}(M)\#w$.
- If the input does not have this form, then A outputs a word that is not in L_{reach}, e.g., the empty word λ.
- If the input has the form $\text{Kod}(M)\#w$, then A constructs a TM $N_{M,w}$ that ignores its own input and, on every input, simulates the TM M on w.
- A determines the number i of q_{accept} in $\text{Kod}(N_{M,w})$ and outputs $\text{Kod}(N_{M,w})\#0^i$.

Now we show that $x \in L_{\text{U}} \iff A(x) \in L_{\text{reach}}$.

Let $x \in L_{\text{U}}$. Then the input has the form $\text{Kod}(M)\#w$ for some $w \in L(M)$, i.e., the computation of M on w terminates in q_{accept}. Since $N_{M,w}$ simulates on every own input the computation of M on w, $N_{M,w}$ accepts every input, i.e., the computation of M terminates in $q_{\text{accept}} = q_i$, and, by definition of codes for Turing machines from Section 4.5 in the textbook, $N_{M,w}$ has at least $i+1$ states. Hence, $\text{Kod}(N_{M,w})\#0^i \in L_{\text{reach}}$.

Let $x \notin L_{\text{U}}$. Then either x is an invalid input or $w \notin L(M)$. If x is an invalid input, then $A(x) = \lambda \notin L_{\text{reach}}$.

Otherwise, $A(x) = \text{Kod}(N_{M,w})\#0^i \notin L_{\text{reach}}$ since the computation of $N_{M,w}$ on no input reaches q_{accept} (i.e., the i-th state).

As the many-one-reduction is a special case of a recursive reduction, $L_{\text{U}} \notin \mathcal{L}_{\text{R}}$ and $L_{\text{U}} \leq_{\text{m}} L_{\text{reach}}$ immediately imply $L_{\text{reach}} \notin \mathcal{L}_{\text{R}}$.