ETH zürich
Departement Informatik

**Theoretische Informatik**

Prof. Dr. Juraj Hromkovič
Dr. Hans-Joachim Böckenhauer
https://courses.ite.inf.ethz.ch/theo_inf_21

# Exemplary Solutions – Sheet 7

Zürich, November 19, 2021

## Solution to Exercise 19

(a) We prove the statement by contradiction. We suppose that $L_{\mathrm{H}}^{\complement} \in \mathcal{L}_{\mathrm{RE}}$. Hence, there exists a Turing machine $M_{\mathrm{H}}^{\complement}$ that accepts $L_{\mathrm{H}}^{\complement}$.

Moreover, the following simple Turing machine $M_{\mathrm{H}}$ accepts $L_{\mathrm{H}}$: for every input $x \in \{0,1,\#\}$, it checks if $x = \mathrm{Kod}(M)\#w$ for some $w \in \{0,1\}^*$. If not, then it rejects the input $x$ and otherwise it simulates $M$ on $w$. If the simulation ends, then $M_{\mathrm{H}}$ accepts the input $x$.

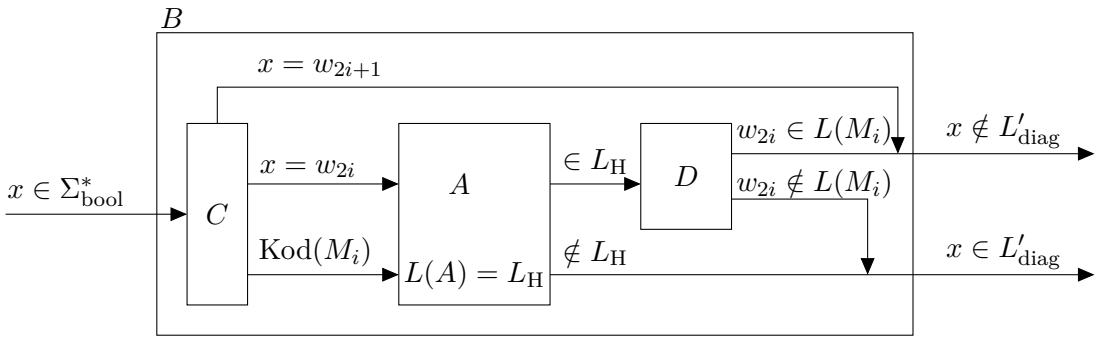
Now we prove that $L(M_{\mathrm{H}}) = L_{\mathrm{H}}$. If $x \in L_{\mathrm{H}}$, then $x = \mathrm{Kod}(M)\#w$ and $M_H$ simulates $M$ on $w$ and this simulation ends, thus $M_{\mathrm{H}}$ accepts and $x \in L(M_{\mathrm{H}})$. On the other hand, if $x \notin L_{\mathrm{H}}$, then there are two possibilities. Either $x$ does not have the form $\mathrm{Kod}(M)\#w$, in this case $M_{\mathrm{H}}$ immediately rejects the input $x$. Otherwise, $M_{\mathrm{H}}$ again simulates $M$ on $w$, but the simulation does not end this time, otherwise $x \in L_{\mathrm{H}}$ would hold. Because the simulation never ends, $M_{\mathrm{H}}$ does not accept $x$, thus $x \notin L(M_{\mathrm{H}})$.

Now we construct an algorithm $S$ that uses the assumed Turing machine $M_{\mathrm{H}}^{\complement}$ to decide $L_{\mathrm{H}}$. On an input $x$, $S$ simulates both Turing machines $M_{\mathrm{H}}$ and $M_{\mathrm{H}}^{\complement}$ on $x$ in parallel. If $M_{\mathrm{H}}$ accepts the input $x$ at some point during the simulation, then $S$ immediately accepts its own input. On the other hand, if $M_{\mathrm{H}}^{\complement}$ accepts the input $x$ at some point during the simulation, then $S$ immediately rejects its own input.

Every word $x$ satisfies either $x \in L_{\mathrm{H}}$, or $x \in L_{\mathrm{H}}^{\complement}$. In the first case, $M_{\mathrm{H}}$ accepts the input $x$ in finite time and thus $S$ accepts its own input $x$. In the second case, $M_{\mathrm{H}}^{\complement}$ accepts $x$ in finite time and thus $S$ rejects its own input $x$.

This shows that $S$ halts on every input and accepts $x$ if and only if $x \in L_{\mathrm{H}}$ and rejects $x$ if and only if $x \in L_{\mathrm{H}}^{\complement}$, i.e., $x \notin L_{\mathrm{H}}$. Hence, $S$ is an algorithm that decides $L_{\mathrm{H}}$ and $L_{\mathrm{H}} \in \mathcal{L}_{\mathrm{R}}$. We know from the lecture that $L_{\mathrm{H}} \notin \mathcal{L}_{\mathrm{R}}$ and thus we get a contradiction.

(b) To show $L'_{\mathrm{diag}} \leq_{\mathrm{R}} L_{\mathrm{H}}$, we assume that $A$ is an algorithm that decides $L_{\mathrm{H}}$. Then we construct an algorithm $B$ that uses $A$ to decide $L'_{\mathrm{diag}}$. The algorithm $B$ is designed as shown in the following diagram:




For every input $x \in \Sigma^*_{\text{bool}}$, the subprogram $C$ computes $j \in \mathbb{N}$ such that $x = w_j$ is the $j$-th word over $\Sigma_{\text{bool}}$ in the canonical order. If $j$ is odd, i.e., $j = 2i + 1$ for some $i \in \mathbb{N}$, then $B$ rejects the input $x$. On the other hand, if $j = 2i$ for some $i \in \mathbb{N}$, then $C$ computes the code $\text{Kod}(M_i)$ of the $i$-th Turing machine in the canonical order. The subprogram $A$ for $L_{\text{H}}$ then gets $\text{Kod}(M_i)$ and $x = w_{2i}$ as input of the form $\text{Kod}(M_i)\#x$.

If $A$ rejects the input $\text{Kod}(M_i)\#x$, then $M_i$ does not halt on $w_{2i}$, i.e., $M_i$ does not accept the word $w_{2i}$ either. Hence, $w_{2i} \in L'_{\text{diag}}$ and $B$ accepts its input $x = w_{2i}$.

If $A$ accepts the input $\text{Kod}(M_i)\#x$, then $M_i$ halts on $w_{2i}$. In this case, the subprogram $D$ simulates $M_i$ on $w_{2i}$. This simulation is guaranteed to end in finite time. If the simulation shows that $M_i$ accepts the word $w_{2i}$, then $w_{2i} \notin L'_{\text{diag}}$ and $B$ rejects its input $x = w_{2i}$. Otherwise, $M_i$ rejects the word $w_{2i}$, then $w_{2i} \in L'_{\text{diag}}$ and $B$ accepts its input $x = w_{2i}$.

## Solution to Exercise 20

(a) To show $L_{\text{H}} \leq_{\text{m}} L_{\text{UU},\lambda}$, we provide an algorithm $F$ that transforms an input $x$ for $L_{\text{H}}$ into an input $f(x)$ for $L_{\text{UU},\lambda}$. The algorithm $F$ first checks if $x$ has the form $\text{Kod}(M)\#w$ for some Turing machine $M$ and some word $w \in \{0,1\}^*$. If not, then $F$ outputs $f(x) = \lambda$. Otherwise, $F$ modifies the Turing machine $M$ encoded in $x$ to a Turing machine $M'$ as follows. First, all transitions to the rejecting state in $M$ are redirected to the accepting state in $M'$. Second, $M'$ always replaces its input, which is initially on the tape, by $w$. Then, $F$ outputs $f(x) = \text{Kod}(M')\#\text{Kod}(M')$.

Now we prove the correctness of our reduction, i.e., $x \in L_{\text{H}} \iff f(x) \in L_{\text{UU},\lambda}$, for all $x \in \{0,1,\#\}^*$.

We first suppose that $x \in L_{\text{H}}$. Then $x = \text{Kod}(M)\#w$ holds for some Turing machine $M$ and some word $w \in \{0,1\}^*$ and $M$ halts on $w$. Furthermore, $M'$ halts on $\lambda$ because $M'$ first replaces $\lambda$ by $w$ and then proceeds exactly like $M$ except that $M'$ makes a transition to the accepting state if $M$ makes a transition to the rejecting state. In particular, $M'$ accepts $\lambda$ and thus $f(x) = \text{Kod}(M')\#\text{Kod}(M') \in L_{\text{UU},\lambda}$, by the definition of $L_{\text{UU},\lambda}$ with $M_1 = M_2 = M'$.

Next we suppose that $x \notin L_{\text{H}}$. Then either $x$ does not have the form $\text{Kod}(M)\#w$, for any Turing machine $M$ and any word $w \in \{0,1\}^*$, which implies $f(x) = \lambda \notin L_{\text{UU},\lambda}$, or $x$ has this form and $M$ does not halt on $w$. Because $M'$ on $\lambda$ first replaces $\lambda$ by $w$ and then proceeds exactly like $M$ as long as an accepting or rejecting state is not reached, $M'$ does not halt either and thus $f(x) = \text{Kod}(M')\#\text{Kod}(M') \notin L_{\text{UU},\lambda}$, by the definition of $L_{\text{UU},\lambda}$ with $M_1 = M_2 = M'$.

(b) To show $L_{\mathrm{U}}^{\complement} \leq_{\mathrm{m}} L_{\mathrm{diag}}$, we provide an algorithm $F$ that transforms an input $x$ for $L_{\mathrm{U}}^{\complement}$ into an input $f(x)$ for $L_{\mathrm{diag}}$. The algorithm $F$ first checks if $x$ has the form $\mathrm{Kod}(M)\#w$ for some Turing machine $M$ and some word $w \in \{0,1\}^*$. If not, then $F$ computes the code of a Turing machine $M^*$ rejecting every possible input, computes the index $j$ of this Turing machine, i.e., a number $j$ such that $M^*$ is the $j$-th Turing machine, and outputs the $j$-th word over $\{0,1\}$ in the canonical order. The output of $F$ is thus $f(x) = w_j$. On the other hand, if $x$ has the form $\mathrm{Kod}(M)\#w$ for some Turing machine $M$ and some word $w \in \{0,1\}^*$, then $F$ computes the code of a Turing machine $M'$ ignoring its own input and simulating $M$ on $w$, computes the index $i$ of $M'$, and outputs the $i$-th word over $\{0,1\}$ in the canonical order. The output of $F$ is thus $f(x) = w_i$.

Now we prove the correctness of our reduction, i.e., $x \in L_{\mathrm{U}}^{\complement} \iff f(x) \in L_{\mathrm{diag}}$, for all $x \in \{0,1,\#\}^*$.

We first suppose that $x \in L_{\mathrm{U}}^{\complement}$ and distinguish two cases. If $x$ does not have the form $\mathrm{Kod}(M)\#w$, for any Turing machine $M$ and any word $w \in \{0,1\}^*$, then $f(x) = w_j$, where $M_j$ is a Turing machine rejecting every possible input. The definition of $L_{\mathrm{diag}}$ implies $f(x) \in L_{\mathrm{diag}}$. On the other hand, if $x = \mathrm{Kod}(M)\#w$ for some Turing machine $M$ and some word $w \in \{0,1\}^*$, then $M$ does not accept $w$. Hence, $M'$ accepts no input because $M'$ always simulates $M$ on $w$. In particular, $M' = M_i$ does not accept $w_i$ either and thus $f(x) = w_i \in L_{\mathrm{diag}}$.

Next we suppose that $x \notin L_{\mathrm{U}}^{\complement}$, i.e., $x \in L_{\mathrm{U}}$. Then $x$ has the form $\mathrm{Kod}(M)\#w$ for some Turing machine $M$ and some word $w \in \{0,1\}^*$ and $M$ accepts $w$. Then $M'$ accepts every possible input because $M'$ always simulates $M$ on $w$. In particular, $M' = M_i$ accepts the word $w_i$. Hence, $f(x) = \mathrm{Kod}(M') \notin L_{\mathrm{diag}}$.

## Solution to Exercise 21

(a) To show $L_{\mathrm{union}} \in \mathcal{L}_{\mathrm{RE}}$, we construct a 3-tape Turing machine $A$ for $L_{\mathrm{union}}$. This suffices because multitape Turing machines and Turing machines are equivalent by Theorem 4.17. in the textbook. The 3-tape Turing machine $A$ first checks if the input is valid, i.e., if it consists of the codes of two Turing machines $M$ and $M'$ and one word $w$. If this is not the case, then $A$ rejects. Otherwise, $A$ first copies the input on the second working tape, then brings the reading head and the head on the second working tape to the beginning of the respective tape, and finally simulates in parallel $M$ on $w$ using the first working tape and $M'$ on $w$ using the second and third working tape, where the second working tape is used as the input tape of $M'$.

If one of the two simulated machines accepts $w$, then $A$ accepts its input as well. If both $M$ and $M'$ reject the word $w$, then $A$ rejects as well. If one of the two machines $M$ and $M'$ does not halt and the other one rejects or does not halt, then $A$ does not halt either. Hence, $A$ is a Turing machine that accepts the language $L_{\mathrm{union}}$.

Note that a sequential simulation of the two machines $M$ and $M'$ is not possible because $M$ might not halt on $w$, but $M'$ could accept $w$.

(b) To show $L_{\mathrm{U}} \leq_{\mathrm{m}} L_{\mathrm{union}}$, we must provide an algorithm that transform inputs for $L_{\mathrm{U}}$ into inputs for $L_{\mathrm{union}}$ so that exactly the accepting inputs for $L_{\mathrm{U}}$ are mapped to accepting inputs for $L_{\mathrm{union}}$. Every word over $\{0,1,\#\}$ that does not have the form $\mathrm{Kod}(M)\#w$, for any Turing machine $M$ and any word $w$, gets mapped

to $\lambda$ because $\lambda \notin L_{\text{union}}$. Every word of the form $\text{Kod}(M)\#w$ gets mapped to $\text{Kod}(M)\#\text{Kod}(M)\#w$, which is a valid input for $L_{\text{union}}$. Moreover, we have

$$\text{Kod}(M)\#\text{Kod}(M)\#w \in L_{\text{union}} \iff \text{Kod}(M)\#w \in L_{\text{U}}\,.$$

Hence, $L_{\text{U}} \leq_{\text{m}} L_{\text{union}}$ is proved.

(c) To show $L_{\text{union}} \leq_{\text{m}} L_{\text{U}}$, we provide a computable function that transforms valid inputs for $L_{\text{union}}$ into valid inputs for $L_{\text{U}}$. Invalid inputs (i.e., inputs that do not have the form $\text{Kod}(M)\#\text{Kod}(M')\#w$) get again mapped to $\lambda \notin L_{\text{U}}$. Every input of the form $\text{Kod}(M)\#\text{Kod}(M')\#w$ gets mapped to $\text{Kod}(A)\#w$, where $A$ is a Turing machine that simulates $M$ and $M'$ in parallel, as described in part (a). Then we have

$$\begin{aligned}\text{Kod}(M)\#\text{Kod}(M')\#w \in L_{\text{union}} &\iff w \in L(M) \cup L(M') \\ &\iff w \in L(A) \\ &\iff \text{Kod}(A)\#w \in L_{\text{U}}.\end{aligned}$$

Hence, $L_{\text{union}} \leq_{\text{m}} L_{\text{U}}$ is proved.