**T-79.148** **Spring 2004**
**Introduction to Theoretical Computer Science**
**Tutorial 12**
**Solutions to the demonstration problems**

4 **Problem**:

Prove, without appealing to Rice's theorem, that the following problem is undecidable:

Given a Turing machine $M$; does $M$ accept the empty string?

**Solution**:

First we define a language $L = \{M \mid M \text{ halts with the input } \varepsilon\}$. Now, $L$ is recursive if and only if the decision problem in the exercise statement is decisive. Next we show that the language $H = \{Mw \mid M \text{ halts with input } w\}$ can be recursively reduced to $L$ (denoted $H \leq_m L$) so $L$ is at least as difficult as $H$. Since $H$ is not recursive, $L$ may not be recursive, either.

The concept of a recursive reduction is defined as follows: Let $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$ be languages. Now $A \leq_m B$ if and only if there exists a recursive function$f : \Sigma^* \to \Gamma^*$ such that

$$\forall w \in \Sigma^* : w \in A \Leftrightarrow f(w) \in B \ .$$

In this case we want to find a function $f$ such that $f(Mw) \in L$ if and only if $Mw \in H$. In practice this means that we want to find a systematic way to construct a Turing machine $M'$ that halts with an empty input exactly when $M$ halts with $w = w_1 w_2 \cdots w_n$.

Fortunately, this is an easy thing to do: $M'$ starts by writing $w$ to its tape and after that it simulates $M$. Now $M'$ stops only if $M$ stops.

Formally, $f$ can be defined as:

$$f(\langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}\rangle, w_1 w_2 \cdots w_n) = \langle Q', \Sigma, \Gamma, \delta', q'_0, q_{\text{acc}}, q_{\text{rej}}\rangle,$$

where

$$\begin{aligned} Q' &= Q \cup \{q'_i \mid 0 \leq i \leq n\} \\ \delta' &= \delta \cup \{\langle q'_i, \varepsilon, q'_{i+1}, w_{i+1}, R\rangle \mid 0 \leq i < n\} \\ &\quad \cup \{\langle q'_n, x, q'_n, x, L\rangle \mid x \in \Gamma \cup \{<\}\} \\ &\quad \cup \{\langle q'_n, >, q_0, >, R\rangle\} \end{aligned}$$

Since we add only a finite number of states and transitions to $M$ ($n$ has to be finite), $f$ is trivially recursive.

5. **Problem**: Prove the following connections between recursive functions and languages:

(i) A language $A \subseteq \Sigma^*$ is recursive ("Turing-decidable"), if and only its characteristic function

$$\chi_A : \Sigma^* \to \{0,1\}, \qquad \chi_A(x) = \begin{cases} 1, & \text{if } x \in A; \\ 0, & \text{if } x \notin A \end{cases}$$

is a recursive ("Turing-computable") function.

(ii) A language $A \subseteq \Sigma^*$ is recursively enumerable ("semidecidable", "Turing-recognisable"), if and only if either $A = \emptyset$ or there exists a recursive function $g : \{0,1\}^* \to \Sigma^*$ such that

$$A = \{g(x) \mid x \in \{0,1\}^*\}.$$

**Solution**: We start by defining five simple helper machines:

- **1** writes '1' to the input tape, moves the read/write head to right and stops.
- **0** writes '0' to the tape and stops.
- $C$ empties the input tape, moves the head to the beginning of the tape and stops.
- NEXT reads the input $x \in \Sigma^*$ and replaces it with the lexicographic successor of $x$.
- $Cmp^{i,j}$ compares the contents of the input tapes $i$ and $j$ of a multi-tape Turing machine and accepts if they are identical.

Since the machines are simple, they are not presented here.

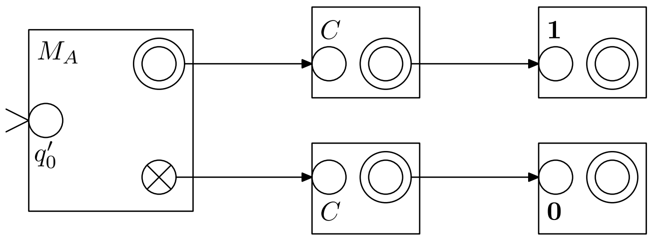
(i) [$\Rightarrow$] Let $A \subseteq \Sigma^*$ be a recursive language. Then there exists a Turing machine $M_A$:

$$M_A = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}} \rangle$$

such that

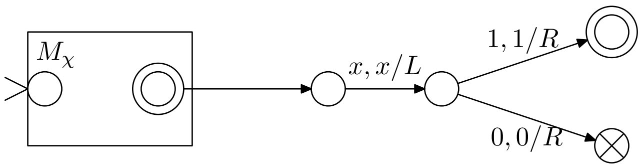
$$\forall w \in \Sigma^* : w \in L \Leftrightarrow (q_0, w) \vdash^*_{M_A} (q_{\text{acc}}, \alpha) \quad \text{ja}$$
$$w \notin L \Leftrightarrow (q_0, w) \vdash^*_{M_A} (q_{\text{rej}}, \alpha)$$

We construct a machine $M$ by combining $M_A$ with machines **1**, **0**, $C$ as follows:



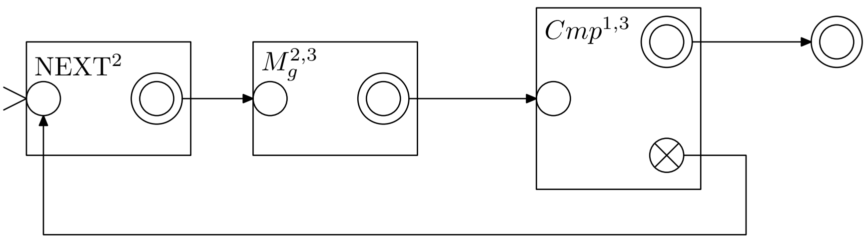

If $w \in L$, then $M_A$ accepts $w$. After that $M$ clears the tape and writes 1 to the tape. Otherwise 0 is written. Since $A$ is recursive, $M_A$ halts always so also $M$ halts and it computes the function $\chi(w) = \begin{cases} 1, w \in A \\ 0, w \notin A \end{cases}$ that is the characteristic function of $A$.

[$\Leftarrow$] Suppose that the function $\chi(w)$ is recursive. Then there exists a Turing machine $M_\chi$ that computes it. We can now construct a machine $M$ as follows:




Now $M$ accepts $w$ whenever $\chi(w) = 1$ and rejects it when $\chi(w) = 0$, so $M$ decides the language $A$ and $A$ is recursive.

(ii) If $A = \emptyset$, then trivially $A \in RE$ and $g(x) = 0$ is its characteristic function.
If there exists a function $g$ that fulfills the conditions, then there exists a Turing machine $M_g$ that computes $g$. We can trivially modify it so that it becomes a 2-tape machine $M_g^{1,2}$ that computes $g$ but stores the result in the second tape instead of the first. We now construct a 3-tape machine as follows:



The machine gets its input from its first tape and it stays untouched for the whole computation. In each iteration $M_A$ replaces the bit string $x$ on the second tape by its lexicographic successor $y$, computes $g(y)$ and writes the output on the third tape. Finally, the contents of tapes 1 and 3 are compared and if they match, the word is accepted, otherwise the iteration proceeds into the next round.

[$\Leftarrow$] Consider the word $w \in A$. Suppose that a recursive function $g$ that fulfills the conditions exists. Then $w = g(x)$ for some $x = x_1 x_2 \cdots x_n$ where $n$ is finite. Since each finite string has a finite number of predecessors in the lexicographic order, NEXT eventually generates $x$, $M_g^{2,3}$ generates $w$ on the third tape and $M_A$ accepts the word. Thus, $M_A$ recognizes the language $A$ so $A \in RE$.

[$\Rightarrow$] Next, suppose that $A \in RE - \{\emptyset\}$. Then there exists a Turing machine $M_A$ that recognizes it. We now define a helper machine $M_{A,i}$ that simulates $M_A$ for $i$ steps. The machine $M_{A,i}$ accepts $x$ if $M_A$ accepts it using at most $i$ steps, and rejects it otherwise. We note that $M_{A,i}$ always halts.

We construct the function $g$ with the help of $M_{A,i}$. Every input $x$ and bound $i$ is encoded into bit strings using the function $c(x, y) = 0^x 1 0^y$. We define that $g(c(x, y)) = x$, if $M_{A,y}$ accepts $x$. We define that $g' : \{0,1\}^* \rightarrow \{0,1\}^*$ is the function:

$$g'(w) = \begin{cases} x, & w = 0^x 1 0^y \text{ and } M_{A,y}(x) \text{ accepts} \\ x_0, & \text{otherwise} \end{cases},$$

where $x_0 \in A$. Finally, $g(x) = d(g'(x))$ where $d$ is a function that maps a bit string $0^x$ into the $x$th element of n $\Sigma^*$ in the lexicographic order. The value of $g'$ may be computed in a finite time since $M_{A,y}(x)$ always halts. Thus, $g'$ is recursive and so also $g$ is.

Note that while $g$ always exists, it is not always possible to find it since in the general case it is an undecidable problem to find an element $x_0 \in A$ that is needed for the definition.