T-79.1001 **Syksy 2006**
Introduction to Theoretical Computer Science (T)
Session 11
Answers to demonstration exercises

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:

   We can prove that a problem is undecidable by showing that we could use its solution to solve some other problem that we know to be undecidable. In this case we will use the *universal language* U as our existing undecidable problem and reduce it to the language $L_\varepsilon = \{c_M \mid \varepsilon \in L(M)\}$ where c_M denotes the encoding of a Turing machine M using some suitable binary encoding.

   Our proof has these steps:

   (a) The universal language $U = \{c_M c_x \mid x \in L(M)\}$ is known to be undecidable. (We take this as given).

   (b) Since U is undecidable, it is not possible to construct a *total* universal Turing machine M_U where $L(M_U) = U$.[1] A Turing machine is total if it halts for every possible input.

   (c) We show that if we have a Turing machine M_ε where $L(M_\varepsilon) = L_\varepsilon$, we can use it as a building block in constructing an universal Turing machine M_U^ε in such a way that we can guarantee that the other parts of M_U^ε are total.

   (d) Since we can create a universal Turing machine M_U^ε out from M_ε and we know that it is not possible to create a total UTM, we conclude that M_ε may not be total so L_ε is undecidable.

   Next, we examine the construction phase in detail.

   Suppose that we can construct a Turing machine M_ε for the language L_ε ($L(M_\varepsilon) = L_\varepsilon$. The machine M_ε gets as its input a binary encoding c_M of some Turing machine M and then it tells whether M accepts the empty string or not. We treat M_ε as a black box: it can use any method to determine the answer and we are not concerned of its interior workings.

   Next, we want to create a universal Turing machine M_U^ε in a way that it will use M_ε to do the hard part of the computation. A UTM gets two inputs, an encoding c_M of a TM M and an encoding c_x of an input string x.

   The UTM M_U^ε will work in two phases:

   (a) First it uses M and x to create a new Turing machine M_x. When this machine is started, it first writes the string x to its tape, rewinds its read/write-head, and then starts to simulate machine M.

   (b) Next, the UTM will use M_ε to check whether the new machine M_x accepts the empty string.

   In the first phase M_U^ε will alter the encoding c_M by adding $|x|+1$ new states for it. In the first $|x|$ states the machine will write one symbol of x to the tape and move the read/write head to right. The last new state rewinds the tape back to the beginning, and then takes

[1] It is possible to construct universal Turing machines but they are not total.

an transition to the original initial state of M. We can implement this phase with a total Turing machine because both c_M and c_x have a finite length by Turing machine definition. The machine M_x does essentially the same computation with an empty input as M does with input x.

If the language L_ε is decidable, then we can make M_ε total. However, in this case M_U^ε is also total. Since this is impossible, we know that M_ε may not be total and L_ε is not decidable but only semi-decidable.

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.

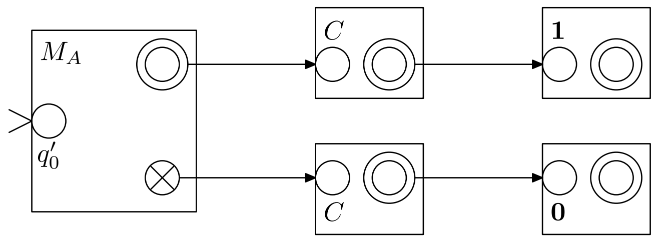
   (ii) A language $A \subseteq \Sigma^*$ is recursively enumerable ("semidecidable", "Turing-recognisable"), if and only if either $A = \emptyset$ or there exists a recursive function $g : \{0,1\}^* \to \Sigma^*$ such that
   $$A = \{g(x) \mid x \in \{0,1\}^*\}.$$

   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.

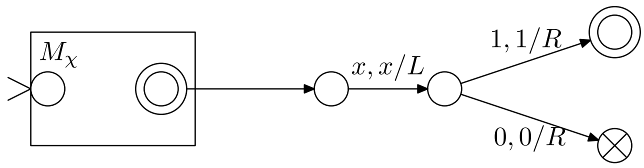
   Since the machines are simple, they are not presented here.

   (i) [⇒] 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
   $$\begin{aligned} \forall w \in \Sigma^* : w \in L \Leftrightarrow (q_0, w) \vdash_{M_A}^* (q_{\text{acc}}, \alpha) \quad \text{ja} \\ w \notin L \Leftrightarrow (q_0, w) \vdash_{M_A}^* (q_{\text{rej}}, \alpha) \end{aligned}$$
   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.

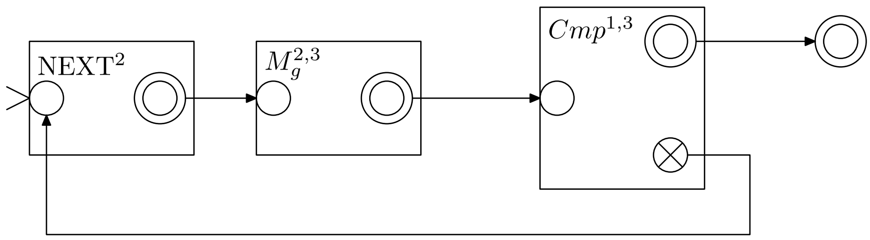
[⇐] Suppose that the function $\chi(w)$ is recursive. Then there exists a Turing machine M_χ that computes it. We can now construct a machine M as follows:



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.

[⇐] 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 $A \in RE$.

[⇒] 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 $g' : \{0,1\}^* \to \{0,1\}^*$ 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 xth element of n Σ^* 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.