Tutorial 10

Exercise 1: Computability

Which of the following languages are computable? If you claim that a particular language is computable, describe a halting DTM for the language.

1. $L_1 = \{\ulcorner M \urcorner \mid M \text{ is a DTM and } M \text{ has (strictly) more than } 5 \text{ states}\}$
2. $L_2 = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM and } M \text{ accepts } w \text{ in (strictly) less than 1000 computational steps}\}$
3. $L_3 = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM and } M \text{ accepts } w \text{ in a finite number of computational steps}\}$
4. $L_4 = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ is a DTM and } M \text{ applied to } w \text{ halts either in the accepting or rejecting state}\}$

Solution:

1. The language L_1 is computable. Consider the following halting DTM M_1 for L_1:
   (a) If the input does not encode a DTM (this can be done by a halting DTM as argued in the lecture), then reject.
   (b) Otherwise, the input is of the form $\ulcorner M \urcorner$. In particular, the input starts with a prefix $1^n\#$ where $n \in \mathbb{N}$ is the number of states of M.
   (c) If n is greater than 5, then accept; otherwise, reject.
   (This can be determined by checking that the first six letters of the input are all 1's.)
2. The language L_2 is computable. Consider the following halting DTM M_2 for L_2:
   (a) If the input is not of the form $\langle \ulcorner M \urcorner, w\rangle$, then reject.
   (b) Otherwise, simulate M on w for 1000 steps.
   (c) If M accepts during the simulation, accept.
   (d) Otherwise, reject.
3. Note that "M accepts w in a finite number of steps" is equivalent to "M accepts w". So, the language L_3 is equal to the language AP and hence we know that it is not computable.
4. Note that a DTM only halts in the accepting or in the rejecting state. All other states are nonterminating. Hence, "M applied to w halts either in the accepting or rejecting state" is equivalent to "M applied to w halts." So, the language L_4 is equal to the language HP and hence we know that it is not computable.

Exercise 2: Computable functions

Argue that the following functions are computable by (informally) describing a DTM that computes them.

1. $f(w) = X^{|w|}$, assuming that $X \notin \Sigma$ (i.e., X cannot occur as a letter in w).
2. $$g(w) = \begin{cases} \ulcorner M' \urcorner & \text{if } w = \ulcorner M \urcorner \text{ for some DTM } M, \text{ where } M' \text{ is obtained from } M \text{ by} \\ & \text{replacing every occurrence of the rejecting state } r \text{ in a transition by} \\ & \text{a looping state (in which } M' \text{ never halts) (and adding relevant transitions)} \\ \varepsilon & \text{otherwise} \end{cases}$$

Recall the definition of a computable function from the lecture. In short: there exists a halting DTM that always halts in the accepting state and with the head on the first symbol of the output $f(w)$.

Solution:

1. This function can be implemented by a DTM with five states: the usual states s, t, r, a state q_r going from left to right through the input replacing every letter (but the first one) by X, and a state q_ℓ to move the head back to the first letter of the tape. As this letter is still unchanged, we replace it by X and move the head left, which means it is still at the first cell.

   Formally, we define the transition function δ as follows:

   - For every letter $a \in \Sigma$ (in the input alphabet), define $\delta(s, a) = (q_r, a, +1)$ (leaving the first letter unchanged). Also, define $\delta(s, \textvisiblespace) = (t, \textvisiblespace, -1)$ to cover the special case of the empty input.
   - For every letter $a \in \Sigma$, define $\delta(q_r, a) = (q_r, X, +1)$ (replacing each other input letter by X). Also, define $\delta(q_r, \textvisiblespace) = (q_\ell, \textvisiblespace, -1)$ which starts the process of moving the head back to the first cell.
   - Define $\delta(q_\ell, X) = (q_\ell, X, -1)$ (to move the head left toward the first cell again) and $\delta(q_\ell, a) = (t, X, -1)$ for every letter $a \in \Sigma$ (replacing the first letter that has been left unchanged so far).
   - For all other values, the transition function can be defined arbitrarily, as they are unreachable.

   Note that the last step moves the head to the left, but since this happens on the first cell of the tape, the head actually stays on that cell.

   Thus, on any input w, this process terminates with the tape only containing $X^{|w|}$ and the head on the first cell again.

2. First, let us recall the definition of $\ulcorner M \urcorner$:

$$1^{|Q|}\,0\,1^{|\Gamma|}\,0\,s\,0\,t\,0\,r\,0\,w_\delta$$

   where w_δ is the list of encodings of transitions. Each $\delta(q, b) = (p, a, d)$ is encoded by

$$q\,0\,\mathit{rep}_\Gamma(b)\,0\,p\,0\,\mathit{rep}_\Gamma(a)\,0\,\mathit{dir}(d)\,0$$

   where $\mathit{dir}(-1) = 1$ and $\mathit{dir}(+1) = 11$. Here, every state is assumed to be of the form 1^n for some $n \geq 1$.

   The DTM computing g works as follows: First, it checks whether the input is an encoding of a DTM M by checking whether it has the format above. If not, it empties the tape and accepts (case ε).

   If the input indeed encodes a DTM, let n be the number of states of the encoded DTM (i.e., the number of 1's before the first 0 in the input). Our DTM first adds another 1 in the beginning of the input (this adds another state).

   Now, the DTM moves its head to the 0 before the beginning of w_δ, i.e., to the fifth 0 in the input.

   Now, for each transition encoding

$$q\,0\,\mathit{rep}_\Gamma(b)\,0\,p\,0\,\mathit{rep}_\Gamma(a)\,0\,\mathit{dir}(d)\,0,$$

   it checks for each number encoded between the second and third 0 in such a block whether it is equal to r and, if so, replaces it by 1^{n+1} (the encoding of the new state). Finally, it appends the encodings

$$1^{n+1}\,0\,\mathit{rep}_\Gamma(b)\,0\,1^{n+1}\,0\,\mathit{rep}_\Gamma(b)\,0\,\mathit{dir}(+1)\,0,$$

   for each letter b of the tape alphabet, making sure that the machine encoded by the output does not terminate in the state 1^{n+1}.

   Once the DTM has completed that, it moves the head back to the initial cell (which requires some mechanism to mark that cell in order to find it again).

Exercise 3: Computable vs. computably-enumerable

Let $L \subseteq \Sigma^*$ be a language. Show that L is computable if and only if L and its complement $\overline{L} = \Sigma^* \setminus L$ are computably-enumerable.

Solution:
To prove the equivalence of both statements, we prove the two implications separately, i.e.,

1. If L is computable, then L and its complement $\overline{L} = \Sigma^* \setminus L$ are computably-enumerable.
2. If L and its complement $\overline{L} = \Sigma^* \setminus L$ are computably-enumerable, then L is computable.

We start with the first implication. If L is computable, then so is $\overline{L}$ (apply closure of computable languages under complementation (see Lecture 9)). Every computable language is also computably-enumerable. Hence, L and $\overline{L}$ are indeed computably-enumerable, as required.

Now, consider the second implication. As L and $\overline{L}$ are computably-enumerable, there are one-tape DTM's M and $\overline{M}$ such that $L(M) = L$ and $L(\overline{M}) = \overline{L}$, i.e.,

- If $w \in L$, then M on input w halts and accepts.
- If $w \notin L$, then M on input w either halts and rejects or does not halt at all.
- If $w \notin L$, then $\overline{M}$ on input w halts and accepts.
- If $w \in L$, then $\overline{M}$ on input w either halts and rejects or does not halt at all.

Consider the following two-tape DTM D: On input w, simulate the run of M on w on the first tape and the run of $\overline{M}$ on w on the second tape. The simulations are run in parallel step-by-step. If the simulation of M on w reaches the accepting state, then D accepts. If the simulation of $\overline{M}$ reaches the accepting state, then D rejects.

It remains to show that D is a halting DTM for L.

If $w \in L$, the simulation of M on w halts by reaching its accepting state (and the simulation of $\overline{M}$ will not reach its accepting state). Hence, D halts and accepts w.

Dually, if $w \notin L$, then the simulation of $\overline{M}$ on w halts by reaching its accepting state (and the simulation of M will not reach its accepting state). Hence, D halts and rejects w.

Altogether, we have $L(D) = L$. As one of the cases "$w \in L$" and "$w \notin L$" is true for every input, we conclude that D halts on every input, i.e., it is a halting DTM.

Exercise 4: The complement of the halting problem

Show that $\overline{\mathrm{AP}}$ (the complement of the language AP from the lecture) is not computably-enumerable.

Solution:
We already know that AP is computably-enumerable. Toward a contradiction, assume that $\overline{\mathrm{AP}}$ is also computably-enumerable. Then, both AP and $\overline{\mathrm{AP}}$ are computably-enumerable and therefore AP is computable (by Exercise 3). However, we know that AP is not computable, i.e., we have derived a contradiction. Hence, $\overline{\mathrm{AP}}$ cannot be computably-enumerable.

Exercise 5: Reductions

Show that the language $\mathrm{NEP} = \{\ulcorner M \urcorner \mid L(M) \neq \emptyset\}$ is not computable by giving a reduction from a suitable non-computable problem (e.g., AP).

Solution:
We will show a reduction from the acceptance problem AP, i.e., $\mathrm{AP} \leq_m \mathrm{NEP}$.

Given (an encoding of) a DTM M and an input w, we define the DTM $\mathcal{E}_{M,w}$, which behaves as follows:

1. It removes its own input from the tape.

2. It writes w on the tape.

3. It simulates M on w, i.e., if the simulation reaches the accepting state (the rejecting state) of M, then $\mathcal{E}_{M,w}$ accepts (rejects) as well.

Note that the behavior of $\mathcal{E}_{M,w}$ does not depend on its input, as this is erased immediately.

Hence, we have the following: M accepts w if and only if $\mathcal{E}_{M,w}$ accepts an input w'.

In particular, $\mathcal{E}_{M,w}$ is either the empty language (if M does not accept w, i.e., if $\langle \ulcorner M \urcorner, w \rangle$ is in AP) or the language Σ^* with all words (if M accepts w).

Formally, let f be the computable function

$$f(x) = \begin{cases} \ulcorner \mathcal{E}_{M,w} \urcorner & \text{if } x \text{ is of the form } \langle \ulcorner M \urcorner, w \rangle \\ \varepsilon & \text{otherwise.} \end{cases}$$

Then, we have, as argued above:

$$x \in \mathrm{AP} \iff f(x) \in \mathrm{NEP},$$

i.e., we have shown $\mathrm{AP} \leq_m \mathrm{NEP}$. Hence, NEP is not computable as well.