Tutorial 9

Exercise 1: Constructing an NTM

Give a halting NTM $(Q, \Sigma, \Gamma, s, t, r, \delta)$ for the language $\{ww \mid w \in \mathbb{B}^*\}$ by specifying all components of the NTM formally, as done in the sample solution of Exercise 5 on Exercise Sheet 7 (which was shown in the slides of Lecture 8). Explain your solution.

You may assume that the tape is doubly-infinite. (We have shown in Exercise 2 on Exercise Sheet 8 that we can do so without loss of generality.)

Hint: You may reuse parts of the DTM from the slides.

Solution:
We begin by explaining the intuition: Given an input $x = x_0x_1 \cdots x_{n-1}$, the NTM nondeterministically guesses some $j \in \{0, 1, \ldots, n\}$ by splitting w into $x_0 \cdots x_{j-1}\#x_j \cdots x_{n-1}$. Now, the NTM deterministically checks whether the resulting word is of the form $\{w\#w \mid w \in \mathbb{B}^*\}$. Note that we already constructed a halting DTM for that language in the previous exercise, so we will reuse it for the latter part.

We implement the split as follows:

1. If the tape is empty, the NTM accepts immediately, as ε is in the language.
2. Otherwise, the NTM goes to a new state p_s and moves its head to the right through x, leaving the cells unchanged, until it nondeterministically inserts a $\#$ on the tape (by overwriting a symbol and then copying the remaining part of x one cell to the right, using states p_0 and p_1).
3. If this nondeterministic choice is not made before x ends, the NTM rejects.
4. Then, the NTM moves the head to the first letter of the tape content and behaves like the halting DTM for $\{w\#w \mid w \in \mathbb{B}^*\}$ (state p_ℓ and the states of the DTM constructed for $\{w\#w \mid w \in \mathbb{B}^*\}$).

   To simplify the finding of the first letter, we assume the NTM has a doubly-infinite tape. Then, the first letter can be found by going left to the first empty cell and then one step to the right again.

Formally, we define the halting NTM $(Q, \Sigma, \Gamma, s', t, r, \delta)$ with

- $Q = \{s', p_s, p_0, p_1, p_\ell, \underbrace{s, q_0, q_1, q_0^\#, q_1^\#, q_\ell, q_\ell^\#, q_s, t, r}_{\text{old DTM}}\}$,

  (Note that the initial state of the NTM is s' while s (the initial state of the DTM) is just a normal state.)
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \#, \underbrace{X}_{\text{old DTM}}, \textvisiblespace\}$,
- and the following $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$, with $b \in \mathbb{B}$:

  (Only the first five/four states/blocks are new. The rest is copied from the old DTM. The letter X cannot occur in the new states/blocks (and $\#$ can only occur in p_ℓ), so the corresponding transitions are arbitrary.)

(new states/transitions below here...)

$$\delta(s', 0) = (p_s, 0, +1)$$
$$\delta(s', 1) = (p_s, 1, +1)$$
$$\delta(s', \#) = (r, \#, +1)$$
$$\delta(s', \sqcup) = (t, \sqcup, +1)$$
$$\delta(s', X) = (r, X, +1)$$

$$\delta(p_s, 0) = \{(p_s, 0, +1), (p_0, \#, +1)\}$$
$$\delta(p_s, 1) = \{(p_s, 1, +1), (p_1, \#, +1)\}$$
$$\delta(p_s, \#) = (r, \#, +1)$$
$$\delta(p_s, \sqcup) = (r, \sqcup, +1)$$
$$\delta(p_s, X) = (r, X, +1)$$

$$\delta(p_b, 0) = (p_0, b, +1)$$
$$\delta(p_b, 1) = (p_1, b, +1)$$
$$\delta(p_b, \#) = (r, \#, +1)$$
$$\delta(p_b, \sqcup) = (p_\ell, b, -1)$$
$$\delta(p_b, X) = (r, X, +1)$$

$$\delta(p_\ell, 0) = (p_\ell, 0, -1)$$
$$\delta(p_\ell, 1) = (p_\ell, 1, -1)$$
$$\delta(p_\ell, \#) = (p_\ell, \#, -1)$$
$$\delta(p_\ell, \sqcup) = (s, \sqcup, +1)$$
$$\delta(p_\ell, X) = (r, X, +1)$$

(old states/transitions from here on...)

$$\delta(s, 0) = (q_0, X, +1)$$
$$\delta(s, 1) = (q_1, X, +1)$$
$$\delta(s, \#) = (q_s, \#, +1)$$
$$\delta(s, \sqcup) = (r, \sqcup, +1)$$
$$\delta(s, X) = (r, X, +1)$$

$$\delta(q_b, 0) = (q_b, 0, +1)$$
$$\delta(q_b, 1) = (q_b, 1, +1)$$
$$\delta(q_b, \#) = (q_b^\#, \#, +1)$$
$$\delta(q_b, \sqcup) = (r, \sqcup, +1)$$
$$\delta(q_b, X) = (r, X, +1)$$

$$\delta(q_b^\#, X) = (q_b^\#, X, +1)$$
$$\delta(q_b^\#, b) = (q_\ell, X, -1)$$
$$\delta(q_b^\#, 1 - b) = (r, 1 - b, -1)$$
$$\delta(q_b^\#, \sqcup) = (r, \sqcup, -1)$$
$$\delta(q_b^\#, \#) = (r, X, +1)$$

$$\delta(q_\ell, X) = (q_\ell, X, -1)$$
$$\delta(q_\ell, \#) = (q_\ell^\#, \#, -1)$$
$$\delta(q_\ell, 0) = (r, 0, -1)$$
$$\delta(q_\ell, \sqcup) = (r, \sqcup, -1)$$
$$\delta(q_\ell, 1) = (r, 1, -1)$$

$$\delta(q_\ell^\#, 0) = (q_\ell^\#, 0, -1)$$
$$\delta(q_\ell^\#, 1) = (q_\ell^\#, 1, -1)$$
$$\delta(q_\ell^\#, X) = (q_r, X, +1)$$
$$\delta(q_\ell^\#, \sqcup) = (r, \sqcup, -1)$$
$$\delta(q_\ell^\#, \#) = (q_\ell^\#, \#, -1)$$

$$\delta(q_s, 0) = (r, 0, -1)$$
$$\delta(q_s, 1) = (r, 1, -1)$$
$$\delta(q_s, X) = (q_s, X, +1)$$
$$\delta(q_s, \sqcup) = (t, \sqcup, -1)$$
$$\delta(q_s, \#) = (r, \#, +1)$$

Exercise 2: Test your understanding

Consider the following claim:

Claim: If L is a computable language and $L' \subseteq L$, then L' is a computable language too.

Is this claim true? If yes, prove it. If not, give a counterexample.

Solution:
This claim is wrong. Let $L = \Sigma^*$. This is clearly a computable language. (Why? Think about it and then check the argument below.)

However, any language L' is a subset of L. Pick any non-computable language L' (e.g., HP), which contradicts the claim.

$L = \Sigma^*$ is computable because there is a halting DTM M with $L(M) = L$. The DTM simply accepts any input immediately.

Exercise 3: Closure properties

Prove that the classes of computable languages and computably-enumerable languages are closed under concatenation.

Recall that this means that if L_1 and L_2 are computable (computably-enumerable) languages, then $L_1 \cdot L_2$ is also computable (computably-enumerable).

Solution:
Let L_1 and L_2 be two computable languages. By definition, there are halting DTM's M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We construct the following 3-tape NTM M accepting $L_1 \cdot L_2$. On input w:

1. Nondeterministically split the input string into two parts $w = w_1 w_2$ and copy w_1 on the second tape and w_2 on the third tape. (See also Exercise 1.)
2. On the second tape, simulate M_1 on w_1.
3. If M_1 accepts, go to step 4; otherwise, M rejects.
4. On the third tape, simulate M_2 on w_2.
5. If M_2 accepts, M accepts; otherwise, M rejects.

Now M is surely a halting NTM because both M_1 and M_2 are halting and $L(M) = L_1 \cdot L_2$. Any 3-tape halting NTM is equivalent to some single-tape halting DTM. Hence we have a halting DTM for the concatenation of L_1 and L_2.

The same construction also works if M_1 and M_2 are only computably-enumerable (e.g., the M_i are not necessarily halting), yielding an NTM that accepts the concatenation $L_1 \cdot L_2$ but is not necessarily halting.

Exercise 4: Spot the error

Below are two "proofs" that try to show that computably-enumerable languages are closed under complement. As we already know, computably-enumerable languages are not closed under complement. Hence, there must be errors in both proofs. Locate the errors by underlining them and explain (e.g., by giving a counterexample) where the problem is.

Proof 1:

Let L be a computably-enumerable language. By definition, there is a DTM M such that $L(M) = L$. Consider the following DTM M':

On input w:

1. Simulate M on w.
2. If M accepts, then M' rejects. If M rejects, then M' accepts.

Obviously, $L(M') = \overline{L}$. Hence, $\overline{L}$ is computably-enumerable.

Proof 2:

Let L be a computably-enumerable language. By definition, there is a DTM M such that $L(M) = L$. Consider the following DTM M':

On input w:

1. Simulate M on w.
2. If M accepts, then M' rejects. If M rejects or does not halt on w, then M' accepts.

Obviously, $L(M') = \overline{L}$. Hence, $\overline{L}$ is computably-enumerable.

Solution:

Proof 1:

Let L be a computably-enumerable language. By definition, there is a DTM M such that $L(M) = L$. Consider the following DTM M':

On input w:

1. Simulate M on w.
2. If M accepts, then M' rejects. If M rejects, then M' accepts.

Obviously, $L(M') = \overline{L}$. Hence, $\overline{L}$ is computably-enumerable.

The presented construction does not describe the complement of the language L. Consider a TM M over $\Sigma = \{a\}$ with three states s, t, and r such that $\delta(s, a) = (s, a, +1)$ and $\delta(s, \textvisiblespace) = (s, a, +1)$. On any given input w, the DTM M will not halt (it keeps moving the head to the right) and hence $L(M) = \emptyset$. Because M''s run on w will first try to simulate M on w, it will never halt either and hence $L(M') = \emptyset$, which is not equal to the complement of the empty language.

Proof 2:

Let L be a computably-enumerable language. By definition, there is a DTM M such that $L(M) = L$. Consider the following DTM M':

On input w:

1. Simulate M on w.
2. If M accepts, then M' rejects. If M rejects or does not halt on w, then M' accepts.

Obviously, $L(M') = \overline{L}$. Hence, $\overline{L}$ is computably-enumerable.

The problem here is that the DTM M' has no way to determine whether the DTM M on w halts or not (it would need to solve the halting problem for that).

For that reason, conditions of the type "if a DTM M on an input w does not halt, then do something" cannot be a part of any correct definition of a DTM because this test cannot be done algorithmically.

Exercise 5: The halting problem is computably-enumerable

Show that the halting problem

$$\mathrm{HP} = \{\langle \ulcorner M \urcorner, w\rangle \mid M \text{ halts on input } w\}$$

is computably-enumerable.

Solution:
We need to construct a DTM H such that $L(H) = \mathrm{HP}$, i.e., if $w \in \mathrm{HP}$, then H halts and accepts on input w, and if $w \notin \mathrm{HP}$, then H either halts and rejects or does not halt on input w.

We define H as follows: On input w:

1. If w is not of the form $\langle \ulcorner M \urcorner, w'\rangle$ such that M is a DTM and w' is an input for M, then reject.
2. Otherwise, simulate the run of M on w'.
3. If M halts, then accept.

This DTM H has the desired properties described above. In particular, H accepts $\langle \ulcorner M \urcorner, w'\rangle$ if M halts on w'. If M does not halt on w', then H does not halt either. Finally, if the input to H is of the wrong format (and therefore not in HP), then H rejects as well.