Tutorial 7

Exercise 1: Test your understanding

Answer the following questions (and justify your answers):

1. Can a DTM ever write the blank symbol $\sqcup$ on its tape?
2. Can the tape alphabet Γ be equal to the input alphabet Σ?
3. Can the head of a DTM ever stay on the same cell for two subsequent steps of a computation?
4. Can the state set of a DTM consist of only a single state?
5. Suppose that a DTM has its head at a cell with symbol a and is in a state q that is different from t and r. Does the DTM always make another transition? And if it does, how many possible (distinct) states can the DTM be in after that transition?
6. Is it always the case that if a language is computable then it is also computably-enumerable?

Solution:
Here are the correct answers:

1. A DTM can write a $\sqcup$, since $\sqcup \in \Gamma$ and the transition function has type $\delta : (Q \setminus \{t, r\}) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}$.
2. The tape alphabet Γ can never be equal to the input alphabet Σ since $\sqcup \in \Gamma$ but $\sqcup \notin \Sigma$.
3. In general, the direction is either -1 or $+1$. Hence, typically, the head either moves to the left or to the right. However, there is one special case: when the head is at the left-most cell (with number 1) and the direction is -1 (i.e., the head should move left), since there is no cell to the left, the head will stay at cell 1. Hence, the answer is yes (but only due to this special case).
4. The state set of a DTM will always contain at least two states, since t and r are different and must be in the set of states.
5. A DTM always makes another transition unless it is in either of the states t and r, and it will always be in exactly one state after a transition step. This is due to the fact that the DTM is deterministic, i.e., δ is a *function* and hence to every element of $Q \times \Gamma$ it assigns exactly one element of $Q \times \Gamma \times \{-1, +1\}$.
6. Yes, every computable language is also computably-enumerable. This follows directly from the definitions. Computable languages are accepted by halting DTMs, which are a special case of DTMs.

Exercise 2: On precise formulations

Some of the following definitions are correct and some wrong (and some are even pure nonsense). Mark the correct definitions and for the incorrect ones underline the part of the definition that is wrong and explain why.

1. A language L is computably-enumerable if the language halts in the state t whenever $w \in L$.
2. A language L is computably-enumerable if there exists a DTM M such that M, given input w, halts in the accepting state (t) if and only if $w \in L$.
3. A language L is computably-enumerable if there exists a DTM M that has an accepting state (t) and is a member of L.

4. A language L is computably-enumerable if there exists a DTM M such that for any given input w the run of the DTM M on w halts in the accepting state (t) if $w \in L$, and it either does not terminate or halts in r if $w \notin L$.

5. A language L is computably-enumerable if every DTM M, when run on a string w, halts in the accepting state (t) if and only if $w \in L$.

Solution:
Here are the correct answers:

1. A language L is computably-enumerable if the language halts in the state t whenever $w \in L$.
   WRONG: *A language does not halt. DTMs halt.*

2. A language L is computably-enumerable if there exists a DTM M such that M, given input w, halts in the accepting state (t) if and only if $w \in L$.
   CORRECT.

3. A language L is computably-enumerable if there exists a DTM M that has an accepting state (t) and is a member of L.
   WRONG: *A DTM cannot be a member of L, as L is a language and therefore contains words and not DTMs. Also, having an accepting state (t) does not imply anything (every DTM is required to have an accepting state).*

4. A language L is computably-enumerable if there exists a DTM M such that for any given input w the run of the DTM M on w halts in the accepting state (t) if $w \in L$, and it either does not terminate or halts in r if $w \notin L$.
   CORRECT.

5. A language L is computably-enumerable if every DTM M, when run on a string w, halts in the accepting state (t) if and only if $w \in L$.
   WRONG: *Not every DTM, but there should exist at least one such a DTM.*

Exercise 3: DTMs and their runs

Consider the following DTM $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ from Example 2.1.3 on page 75 of "Computability and Complexity" with

- $Q = \{s, t, r, h, e, g\}$,
- $\Sigma = \{a, b\}$,
- $\Gamma = \{a, b, \sqcup\}$, and
- δ given by the following table:

δ	a	b	$\sqcup$
s	$(h, \sqcup, +1)$	$(r, b, +1)$	$(t, \sqcup, +1)$
h	$(h, a, +1)$	$(h, b, +1)$	$(e, \sqcup, -1)$
e	$(r, a, +1)$	$(g, \sqcup, -1)$	$(r, \sqcup, +1)$
g	$(g, a, -1)$	$(g, b, -1)$	$(s, \sqcup, +1)$

Have a look at Examples 2.1.3, 2.1.5, and 2.1.11 before you attempt the exercise.

1. Give the initial configurations of M on the inputs $w_0 = aaa$ and $w_1 = aab$, both using the notation from Slide 21 and the simplified notation from Slide 22.

2. Give the runs of M on w_0 and w_1. Use the simplified notation to write down the configurations.
3. Are w_0 and w_1 accepted or rejected by M?

Solution:

1. Initial configurations:
   (a) On aaa: $[s, \tau_0, 1]$ with $\tau_0(n) = \begin{cases} a & \text{if } n \leq 3, \\ \sqcup & \text{if } n > 3, \end{cases}$ or, in the simplified notation, $[s, aaa\sqcup \ldots, 1]$.
   (b) On aab: $[s, \tau_1, 1]$ with $\tau_1(n) = \begin{cases} a & \text{if } n \leq 2, \\ b & \text{if } n = 3, \\ \sqcup & \text{if } n > 3, \end{cases}$ or, in the simplified notation, $[s, aab\sqcup \ldots, 1]$.
2. Runs:
   (a) On aaa:
   $$[s, aaa\sqcup \ldots, 1] \vdash_M [h, \sqcup aa\sqcup \ldots, 2] \vdash_M [h, \sqcup aa\sqcup \ldots, 3] \vdash_M [h, \sqcup aa\sqcup \ldots, 4] \vdash_M$$
   $$[e, \sqcup aa\sqcup \ldots, 3] \vdash_M [r, \sqcup aa\sqcup \ldots, 4]$$
   (b) On aab:
   $$[s, aab\sqcup \ldots, 1] \vdash_M [h, \sqcup ab\sqcup \ldots, 2] \vdash_M [h, \sqcup ab \ldots, 3] \vdash_M [h, \sqcup ab\sqcup \ldots, 4] \vdash_M$$
   $$[e, \sqcup ab\sqcup \ldots, 3] \vdash_M [g, \sqcup a\sqcup \ldots, 2] \vdash_M [g, \sqcup a\sqcup \ldots, 1] \vdash_M [s, \sqcup a\sqcup \ldots, 2] \vdash_M$$
   $$[h, \sqcup \ldots, 3] \vdash_M [e, \sqcup \ldots, 2] \vdash_M [r, \sqcup \ldots, 3]$$
3. Both words are rejected by M, as the runs on them each end in a rejecting configuration.

Exercise 4: Constructing a simple DTM

Consider the language of words over the alphabet $\{0, 1\}$ whose first letter is equal to their last letter.

1. Write a formal definition of the language informally described above. In particular, think about corner cases and how to handle them.
2. Give a DTM that accepts the language you have defined in Step 1.
3. Give the run on input 101.
4. Give the run on input 100.
5. Give one run on each corner case you considered.

Solution:

1. The corner cases we need to consider are the empty word and words of length one:
   - The empty word does not have any letters (it has length zero after all), so it in particular has no first and no last letter. Hence, it is not in our language.
   - On the other hand, words of length 1 (e.g., "0") have a first letter and a last letter, which happen to be at the same position. Hence, they must be equal and such words are therefore in the language.

One possible formal definition of the language is

$$\{a_1 a_2 \cdots a_n \mid a_j \in \{0,1\} \text{ for all } 1 \leq j \leq n, n \geq 1, \text{ and } a_1 = a_n\}.$$

2. The DTM we construct works intuitively as follows:
   - If there is no first letter, reject.
   - Store the first letter using the states q_0 and q_1.
   - In the states q_0 and q_1, go to the right until the first $\sqcup$ is reached.
   - When reaching this $\sqcup$ in state q_b for $b \in \{0,1\}$, go left and into state chck_b. The head is then on the last letter of the input.
   - Compare that to the letter stored in the state chck_b and accept if they are equal, otherwise reject. As the stored letter is the first one, this final check compares the first to the last letter.

   Formally, we define $M = (Q, \Sigma, \Gamma, s, t, r, \delta)$ with
   - $Q = \{s, t, r, q_0, q_1, \mathrm{chck}_0, \mathrm{chck}_1\}$,
   - $\Sigma = \{0, 1\}$,
   - $\Gamma = \{0, 1, \sqcup\}$,
   - and δ given by the following table:

δ	0	1	$\sqcup$
s	$(q_0, 0, +1)$	$(q_1, 1, +1)$	$(r, \sqcup, +1)$
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(\mathrm{chck}_0, \sqcup, -1)$
q_1	$(q_1, 0, +1)$	$(q_1, 1, +1)$	$(\mathrm{chck}_1, \sqcup, -1)$
chck_0	$(t, 0, +1)$	$(r, 1, +1)$	$(r, \sqcup, +1)$
chck_1	$(r, 0, +1)$	$(t, 1, +1)$	$(r, \sqcup, +1)$

3. Run on 101:

$$[s, 101\sqcup\cdots, 1] \vdash_M [q_1, 101\sqcup\cdots, 2] \vdash_M [q_1, 101\sqcup\cdots, 3] \vdash_M$$
$$[q_1, 101\sqcup\cdots, 4] \vdash_M [\mathrm{chck}_1, 101\sqcup\cdots, 3] \vdash_M [t, 101\sqcup\cdots, 4]$$

4. Run on 100:

$$[s, 100\sqcup\cdots, 1] \vdash_M [q_1, 100\sqcup\cdots, 2] \vdash_M [q_1, 100\sqcup\cdots, 3] \vdash_M$$
$$[q_1, 100\sqcup\cdots, 4] \vdash_M [\mathrm{chck}_1, 100\sqcup\cdots, 3] \vdash_M [r, 100\sqcup\cdots, 4]$$

5. Run on ε: $[s, \sqcup\cdots, 1] \vdash_M [r, \sqcup\cdots, 2]$.
   Run on 0: $[s, 0\sqcup\cdots, 1] \vdash_M [q_0, 0\sqcup\cdots, 2] \vdash_M [\mathrm{chck}_0, 0\sqcup\cdots, 1] \vdash_M [t, 0\sqcup\cdots, 2]$.
   Run on 1: $[s, 1\sqcup\cdots, 1] \vdash_M [q_1, 1\sqcup\cdots, 2] \vdash_M [\mathrm{chck}_1, 1\sqcup\cdots, 1] \vdash_M [t, 1\sqcup\cdots, 2]$.

Exercise 5: Constructing a (slightly more complex) DTM

Consider the language $L = \{w\#w \mid w \in \{0,1\}^*\}$.

1. Give a halting DTM for L. Explain your solution in natural language.
2. Give the accepting run on $\# \in L$.
3. Give the accepting run on $011\#011 \in L$.
4. Give the rejecting run on $01\#00 \notin L$.

Solution:
See the slides of the next lecture.

Exercise 6: Challenge

Show that the following problem is computable:

$$P = \{n \in \mathbb{N} \mid \text{the decimal expansion of } \pi \text{ contains } n \text{ consecutive } 0\text{'s}\}$$

Describe your algorithm on an intuitive level without going into technical details.

Solution:
We begin with the following observation: If n is in P, then also every smaller number. For example, if $5 \in P$ (because the decimal expansion contains the sequence 00000), then also $4 \in P$ (as 0000 is a subsequence of the previous one). Hence, by taking the contraposition: if n is not in P, then also no larger number.

The only subsets of $\mathbb{N}$ satisfying this property are

- $\mathbb{N}$ itself, and
- the finite sets $\mathrm{Pref}(n) = \{0, 1, \ldots, n\}$ for $n \in \mathbb{N}$.

Note that all these sets are trivially computable.

So, P is either equal to $\mathbb{N}$ or equal to some $\mathrm{Pref}(n)$ (note that we do not know which case it is). So, P is computable (but we do not know which halting DTM is the one that computes our language).