Tutorial 2

Exercise 1 (compulsory)

Can 3-tape Turing machines recognize a larger class of languages than 2-tape Turing machines?

- If your answer is positive, then provide an example of a language recognizable by a 3-tape Turing machine but not recognizable by any 2-tape Turing machine.
- If your answer is negative, give a short but complete argument supporting your claim.

Solution:
The answer is negative. Turing machines with 3 tapes recognize exactly the same class of languages as 2-tape Turing machines. Let L be a language recognized by a given 3-tape Turing machine. Thanks to Theorem 3.13 on page 177 we know that there is a single tape Turing machine recognizing L and any single tape machine can be trivially simulated by 2-tape Turing machine (we simply do not use the second tape). Hence the language L is recognized also by a Turing machine with 2 tapes.

Exercise 2 (compulsory)

One can imagine an extension of the Turing machine model such that the transition function is of the type

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{V}\}$$

where V denotes that the head is to be immediately (in one step) moved to the leftmost tape-cell. Is this extension more powerful than the original Turing machine model? Give precise arguments for your claim, at least on the implementation level description (see the text on page 170-171 after 'Examples of Turning machines').

Solution:
The answer is that the extended model is not more powerful than the original one. In order to prove this, we will show how the extended Turing machine can be simulated by our standard model of a Turing machine.

The idea is that the standard model can do exactly the same transitions as the extended one, except for the V-move. In order to simulate the V-move (which happens in one step), we need to move the head to the beginning of the tape one by one. Moreover, a Turing machine cannot determine if its head is located at the leftmost tape cell unless we use some trick. We will introduce a new tape symbol $\#$ which will mark the very first cell on the tape.

Let M be the extended Turing machine that we wish to simulate. The simulation proceeds as follows:

On input w:

1. Shift the input string w one cell to the right and write $\#$ in the leftmost cell.
2. Place the head of the machine on the cell immediately to the right of the symbol $\#$.
3. Perform the same steps as M, but replace V-transitions by moving the head to the left (one by one) until the machine reaches the end marker $\#$. Then move the head one cell to the right and continue the simulation of the machine M.

This is just one example of how this problem can be solved. Your solution, though different from the above presented one, can be still correct.

Exercise 3 (compulsory)

Which of the definitions of nondeterministic Turing machines given below is correct? Read them carefully before you answer.

1. A nondeterministic Turing machine M accepts input x if there exists more than one computation of M on x such that M halts in the state q_{accept}.
2. A nondeterministic Turing machine M accepts input x if no computation of M on x halts in the state q_{reject}.
3. A nondeterministic Turing machine M accepts input x if there exists a computation of M on x such that M halts in the state q_{accept}.
4. A nondeterministic Turing machine M accepts input x if there is exactly one accepting computation of M on x.
5. A nondeterministic Turing machine M accepts input x if for every computation of M on x we have that M halts in state q_{accept}.

Complete the definition below and be as precise as possible:

> A nondeterministic Turing machine M is called *decider* if ...

Solution:
Only the third definition is correct. The definition of a nondeterministic decider is as follows:

> A nondeterministic Turing machine M is called *decider* if for any given input $x \in \Sigma^*$ and any computation of M on x the machine halts (either in q_{accept} or q_{reject}).

Exercise 4 (compulsory, write down your solutions)

A *lexicographical enumerator* E is a two-tape TM with a special state q_{print} that prints strings on the second tape (by entering the state q_{print}) in the lexicographical order, meaning that once a string w is printed, all other strings that are printed afterwards are in lexicographical order strictly larger than w, in particular implying that their length is at least $|w|$.

Prove that the class of languages that are generated by lexicographical enumerators is equal to the class of decidable languages.

Solution:
We have to show two directions: a) if L is a language generated by a lexicographical enumerator then L is a decidable language, and b) if L is a decidable language then it is generated by some lexicographical enumerator.

To prove a) we consider two cases. If L is a finite language then it is clearly a decidable language (make sure you can argue why). Let L be an infinite language generated by the lexicographical enumerator E. We construct a decider M for language L in order to prove that L is decidable lenguage. We let

> M = "On input w:
>
> 1. Run the enumerator E (on two newly added tapes) until it prints a string w'.
> 2. If $w = w'$ then **accept**.
> 3. If $|w| < |w'|$ then **reject**.
> 4. Return to step 1 and continue running the enumerator."

Now we have to argue that M is a decider, i.e. that on any input w it halts after finitely many steps. Clearly, as L is an infinite language then E keeps printing infinitely many strings. However, there are only finitely many strings of length at most $|w|$ and hence eventually the enumerator E must print a string of length larger than $|w|$, implying that M will reject (unless it accepted even earlier should E print the string w). The fact that the strings are printed in lexicographical order guarantess that once E prints a string w' which is longer than w (and hence comes in lexicographical order after w), it is not possible that w is printed afterwards and the machine M can reject the input.

To prove b), let L be a decidable language that is recognized by a decider M. We construct a lexicographical enumerator E generating L as follows:

E = "Ignore input.

1. Let $w := \epsilon$.
2. Run M on w and if M accepted then print w.
3. Let $w := w'$ where w' is the next string is lexicographical order after w. Goto step 2."

Clearly, the enumerator prints all strings in lexicographical order as required and moreover, because M is a decider and does not loop on any string, all accepted strings will be eventually printed.

Exercise 5 (compulsory)

What is the statement of Church-Turing Thesis?

Solution:
Church-Turing Thesis: "The Turing machine model captures exactly the informal notion of algorithm."

Exercise 6 (optional but recommended)

- Problem 3.18 on page 189 (in international edition) or Problem 3.11 on page 189 (in standard edition) (easy).
- Problem 3.20 on page 189 (in international edition) or Problem 3.13 on page 189 (in standard edition) (medium difficult).
- Problem 3.17 on page 189 (in international edition) or Problem 3.10 on page 188 (in standard edition) (hard).