Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
P. Bamberger, Y. Maus,

Theoretical Computer Science - Bridging Course
Summer Term 2017
Exercise Sheet 5

Hand in (electronically or hard copy) by 12:15 pm, November 27th, 2017

Exercise 1: Turing machines *(1+1+4+2 Points)*

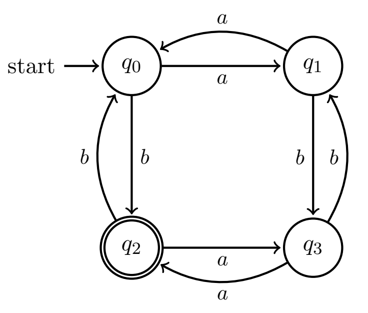
(a) Give a **comparison** of the set of languages recognized by **deterministic** Turing machines with the set of languages recognized by **non-deterministic** Turing machines.

(b) State **two differences** between **deterministic** and **non-deterministic** Turing machines.

(c) One can define a variant of the Turing machine which allows **three** actions of the read/write-head: $\boldsymbol{\{L, R, S\}}$, where $\boldsymbol{S}$ means that the head stands still during that step.

Let M_1 be a Turing machine **that uses head movements** $\boldsymbol{\{L, R, S\}}$. Give an **explicit** construction procedure that transfers M_1 into a Turing machine M_2 **that uses only head movements** $\boldsymbol{\{L, R\}}$ and recognizes the same language, i.e. $L(M_1) = L(M_2)$.

(d) **Briefly** explain how to construct (or construct) a Turing machine for the language defined by the following deterministic finite automaton over the alphabet $\{a, b\}$.



Sample Solution

(a) As proven in the lecture, deterministic and non deterministic Turing machines are equally powerful in terms of the languages they recognize. Hence **both language classes are equal**. *Remark: The bold sentence suffices to receive full points.*

(b) Non-exhaustive list of possible namings of differences:

1. Non-deterministic Turing machines might define multiple transitions for the same state and symbol read by the read/write-head.
2. The transition *function* δ of deterministic Turing machines maps to $Q \times \Gamma \times \{L, R\}$. The transition function of non-deterministic Turing machines maps to $P(Q \times \Gamma \times \{L, R\})$.

3. Non-deterministic Turing machines *probably* recognize more languages in polynomial time than deterministic ones (if $\mathcal{P} \neq \mathcal{NP}$).
4. Acceptance is defined differently. In the case of non-deterministic Turing machines the existence of an accepting path suffices to recognize an input, even though other paths for that input may not be accepting.

(c) We take the TM machine M_1 and subsequently remove all transitions which contain the neutral movement until we obtain a Turing machine according to the standard definition. For each state r add an additional state $\tilde{r}$. If the transition $\delta(q, a) = (r, b, S)$ occurs, replace it by the transitions $\delta(q, a) = (\tilde{r}, b, L)$ and $\delta(\tilde{r}, x) = (r, x, R)$ for all $x \in \Sigma$.

Reasoning: (not required for full points) After execution of these two transitions the Turing machine will be in the same configuration as it would have been when executing the original transition. Each step of M_1 can be simulated with at most two steps by M_2.

(d) We take the states $q_0, \ldots, q_3$ as the states of the Turing machine and add an accept and reject state (important: do not let q_2 be the accept state of the TM). When the head reads a symbol, it moves to the right and changes its state according to the DFA's transition function (without writing on the tape). When it reaches the first blank symbol (i.e. the end of the input), it enters the accept state when it was in state q_2 (the accept state of the DFA) and the reject state otherwise.

Exercise 2: Constructing a Turing Machine *(6 Points)*

Let $L = \{\langle w \rangle, \langle w+1 \rangle \mid w \in \mathbb{N}\}$, e.g., the word $\langle 6 \rangle, \langle 7 \rangle = 110, 111$ is contained in L. Design a Turing machine which accepts L. You do not need to provide a formal description of the Turing machine but your description has to be detailed enough to explain every possible step of a computation.
Remark: Here $\langle w \rangle$ is the binary encoding of the number w, e.g., the number 6 is going to be the string 110.

Sample Solution

We first check whether the string is in the format $\{0, 1\}^*, \{0, 1\}^*$. Then we add 1 to the left hand string and then we simply compare the left and right string symbol by symbol and accept if they are equal, otherwise we reject.
The tricky part is transforming the left string such that it is the binary encoding of $w + 1$.
For that we move to the very right of it; assume that after this the TM is in some state q_1. The meaning of q_1 is that we need to add one to the number which is encoded by the string which starts at the very left and reaches until the current head position.
If we read a 0 in q_1 we simply change it to a 1. In this case we are done and can move to the comparing step. However, if we read a 1 we cannot add 1 to the respective position. Just as when doing *written addition* we thus have to carry the addition over to the next position. That is, we change the current symbol on the tape from 1 to 0, do not change the state and move the read/write head one position to the left. Whenever we reach a 0 in this process we'll be done. But when adding 1 to a string like $s = 11111$ we will never reach a zero, but instead we reach the left end of the string. Adding 1 to s results in 100000, i.e., if we reach the blank symbol in state q_1 we also need to substitute it with a 1 and we can continue with the comparing step.
In the comparison step we always overwrite the symbols which we have already compared with a new symbol Z to find the corresponding position when comparing the next two symbols.

Exercise 3: Turing Machine *(flexible 1+1+1+1+1+1 Points)*

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{reject}, q_{accept})$ be a **deterministic** Turing machine over alphabet $\Sigma = \{0, 1\}$ with tape alphabet $\Gamma = \{0, 1, \sqcup\}$ that **always halts**. Furthermore, for all of the below exercises assume that M is **linearly space restricted**. You may assume that M never moves outside a range of $5n$ consecutive cells on the input tape when started on an input $w \in \Sigma^*$ of length n.

(a) Upper bound the **different read/write-head positions** that M could assume when running on an input of length n.

(b) Upper bound the **combinations of read/write-head positions and possible states** that M could assume on each position. Use the number of states $m := |Q|$ of M.

(c) Upper bound the **possible number of strings** that M could write on the tape when running on an input of length n with the tape restriction mentioned above.

(d) Upper bound the **possible number of configurations** of M on an input of length n with the tape restriction mentioned above.

(e) Explain why M can **never encounter a configuration twice**, when started on an input of size n. Keep in mind that M is a **deterministic decider**.

(f) Now, assume that the Turing machine M holds on every input. Upper bound its **maximum number of steps** on an input of length n.

Remark: *Whenever we ask for upper bounds we want a bound as tight as possible. Giving looser upper bounds might yield partial points.*

Sample Solution

(a) As the Turing machine only uses $5n$ cells the read/write-head is on one of these cells, i.e., there are $5n$ possible positions for the head.

(b) There are $5 \cdot m \cdot n$ combinations.

(c) As the words that M could write are contained in the $5n$ cells, i.e., have length at most $5n$, there are $|\Gamma|^{5n} = 3^{5n}$ possible words.

(d) A configuration consists of a word on the input tape, a state and a head position. Thus there are $5 \cdot m \cdot n \cdot 3^{5n}$ possible configurations.

(e) Assume that M hits the same configuration twice. Then it will also hit the configuration a third time, as it will behave exactly the same when hitting the configuration the second time as it did when hitting it the first time. One can repeat this argument to show that the Turing machine will end up in an infinite cycle, i.e., it does not always halt, a contradiction to the assumption in the exercise.

(f) Every step of the Turing machine changes the configuration of the machine. As a deterministic Turing machine that always halts cannot encounter the same configuration twice and there are $5 \cdot m \cdot n \cdot 3^{5n}$ configurations we know that the Turing machine performs at most $5 \cdot m \cdot n \cdot 3^{5n}$ steps.