Computability - Solution of Exercise 2

Recall the recursive definition of the extended transition function $\delta^* : Q \times \Sigma^* \to Q$, by a given transition function $\delta : Q \times \Sigma \to Q$, as follows.
For $q \in Q$ we have $\delta^*(q, \epsilon) = q$, and for $q \in Q$, $u \in \Sigma^*$, and $a \in \Sigma$, we have $\delta^*(q, ua) = \delta(\delta^*(q, u), a)$.

1. (a) A non-deterministic automaton that recognizes L_n appears is in Figure 1. The automaton guesses the n-th place before the end of the input word, and checks that it contains 'a'. Then it counts that indeed after the 'a' there are n-1 letters in the input word, and accepts only if this is the case.

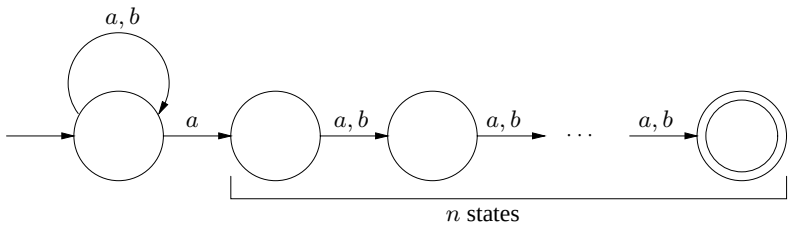

Figure 1: A non-deterministic automaton for L_n.

It is clear from the structure of the automaton, that it has an accepting run on a word w if and only if the n-th letter from the end of w is 'a'. The automaton has $n + 1 = O(n)$ states.

(b) We prove the claim by contradiction. Suppose that there exists a deterministic automaton A_{det} that recognizes L_n, with less than 2^n states. The number of possible words of length n over Σ is 2^n, thus there are two different words w_1 and w_2 of length n over Σ such that the runs of $\mathcal{A}_{det}$ on w_1 and w_2 end in the same state q (by the pigeonhole principle). Let i be the first index in which w_1 and w_2 differ. Without loss of generality, the i-th letter of w_1 is a, and the i-th letter of w_2 is b. Then we can represent w_1 as $u \cdot a \cdot v_1$ and w_2 as $u \cdot b \cdot v_2$, where $|u| = i - 1$, and $|v_1| = |v_2| = n - i$. Now consider the word $w = w_1 \cdot u = u \cdot a \cdot v_1 \cdot u$. Since $|v_1 \cdot u| = n - 1$, the i-th letter from the end of w is 'a', thus $w \in L_n$. Therefore, the (single) run of $\mathcal{A}_{det}$ on w is accepting. Recall that the run of $\mathcal{A}_{det}$ on $w_1 = u \cdot a \cdot v_1$ ends in the state q. Thus, in $\mathcal{A}_{det}$, the state $\delta^*(q, u)$ is accepting. Now let $w' = w_2 \cdot u = u \cdot b \cdot v_2 \cdot u$. We know that the run of $\mathcal{A}_{det}$ on w_2 ends in q, and that $\delta^*(q, u)$ is accepting. Thus, $\mathcal{A}_{det}$ accepts w'. However, the n-th letter from the end of w' is b, thus $w' \notin L_n$, and we have reached a contradiction.

2. (a) For a regular language L, the language $Pref(L)$ is regular. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a DFA that accepts L. We define $\mathcal{A}_{pref} = \langle Q, \Sigma, \delta_{pref}, q_0, F' \rangle$ that accepts $Pref(L)$, where $F' = \{q \in Q | \exists w \in \Sigma^* \ s.t. \ \delta^*(q, w) \in F\}$.
For $x \in Pref(L)$, there is a y such that $xy \in L$. The state $\delta^*(q_0, x)$ must be in F', since $\delta^*(q_0, xy) \in F$. Therefore, $x \in L(A_{pref})$.
For $x \in L(A_{pref})$, $\delta^*(q_0, x) \in F'$, hence there exists a w such that $xw \in L$. Therefore, $x \in Pref(L)$.

(b) Constructing an automaton is trivial, what follows is an alternative proof. For a regular language L, the language $Suff(L)$ is regular. It is easy to see that $[Pref(L^R)]^R = Suff(L)$. The left-hand-side is regular, since regular languages are closed under both the reverse operation (question 3), and under the prefix operation (the above item).

3. Let $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ be a DFA accepting L. We build an NFA $A' = \langle Q, \Sigma, F, \delta^{-1}, Q_0\rangle$ accepting L^R as follows: the initial states of A' are the *accepting states* of A. The transition function of A' is δ^{-1} the inverse relation of δ defined as: $\delta^{-1}(s, \sigma) = \{r \in Q : \delta(r, \sigma) = s\}$. The accepting states of A' are the *initial states* of A.

   Let $w = w_1 \cdot w_2 \cdots w_n$ be a word in L. Look at the run of A on w. This is a sequence of states $r_0, r_1, \ldots, r_n$ where $r_0 \in Q_0$, $r_{i+1} = \delta(r_i, \sigma_{i+1})$, and r_n is in F (why?). Therefore, $r_n, r_{n-1}, \ldots, r_0$ is computation of A' on input w^R.

   On the other hand, if $r'_0, \ldots, r'_n$ is an accepting run of A' on input w, then $r'_n, r'_{n-1}, \ldots, r'_0$ is an accepting run of A on input w^R. Thus $L(A') = L^R$ and L^R is regular.

4. (a) Assume towards contradiction that there exists an NFA $A = \langle Q, \Sigma, Q_0, \delta, F\rangle$ such that its language $L(A)$ is $\{ww \mid w \in \{0,1\}^n\}$, and $|Q| < 2^n$.

   For each word $w \in \{0,1\}^n$, the word ww is in L, and is therefore accepted by some run $r_0^w, r_1^w, \ldots, r_n^w, r_{n+1}^w, \ldots, r_{2n}^w$ of A. Note that there are 2^n words in $\{0,1\}^n$. Therefore, since $|Q| < 2^n$, there must be two different words w and u in $\{0,1\}^n$ for which $r_n^w = r_n^u$. This means that $r_0^w, r_1^w, \ldots, r_n^w = r_n^u, r_{n+1}^u, \ldots, r_{2n}^u$ is an accepting run of A on wu. Note, however, that wu is not in the language of A, and we reach a contradiction.

   (b) The language $\overline{L_n}$ is the union of two cases: first, it might be that the word is not of length $2n$. Second, it might be that the word is of length $2n$ but is not of the type ww. In the second case, it is always the case that there exists two letters at distance n that are different (which is impossible if the word is of the form ww). We therefore choose non-deterministically between two $O(n)$ size automata. The first A_1 checks that a word is of length different then $2n$ and the second A_2 checks that there are two letters in distance n that are different. Note that A_2 will accept also words of length different then $2n$ but we do not care. The construction of A_1 and A_2 is easy and we leave it as an exercise.

5. See the deterministic automaton in Figure 2.

6. We construct an NFA for the language $L_{\frac{1}{2}}$. Since L is regular, there is a DFA $A = \langle Q, \Sigma, \delta, s_0, F\rangle$ that accepts it. Define the NFA $A' = \langle Q', \Sigma, \delta', s'_0, F'\rangle$, as follows:

   $Q' = Q \times Q \times Q$
   $Q_0 = \{(s_0, s, s) | s \in Q\}$
   $F' = \{(s, s, q) | q \in F, s \in Q\}$
   $\delta'((s, q, r), a) = \{(\delta(s, a), q, \delta(r, b)) | b \in \Sigma\}$

   Intuitively, the NFA A' runs in parallel on two tracks. The first track (which is represented by the first element of each state) simulates A from the starting state. The second track (which is represented by the third element of each state) simulates A from some state (which we nondeterministically guess). The second element of each state is fixed throughout the run path and is used to remember the state from which we started the second track. We accept if and only if we have a computation path on the first track that starts in s_0 and on reading the input w it terminates in the state s, that was guessed at the start of the run. At the same time we have a computation path (that we choose nondeterministically) on the second track, that starts in s and terminates in an accepting state. We can then conclude that there exists y such that $wy \in L$.

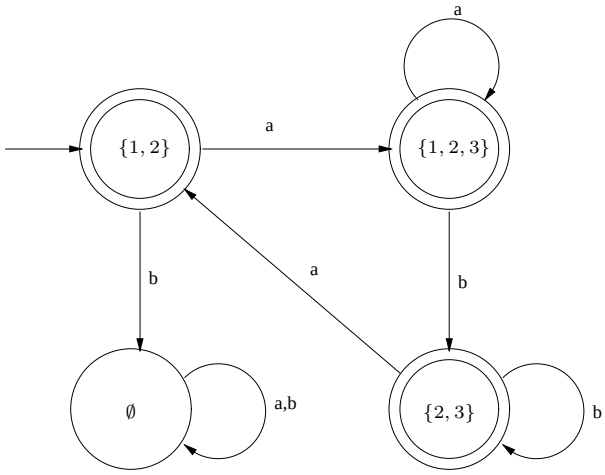

Figure 2: I'm deterministic!

Formally, we want to show that $L(A') = L_{\frac{1}{2}}$.
($\Rightarrow$) Assume that $w \in L(A')$. Then, there is an accepting run $(s_0, q, r_0), ..., (s_n, q, r_n)$ of A' on w. By the definition of A', we have that $r_0 = q$, $s_n = q$, and $r_n \in F$. Consider the second track of the run, by definition, $r_{i+1} = \delta(r_i, b_i)$ for some $b_i \in \Sigma$ ($0 \leq i < n$). Define $y = b_0 b_1 ... b_{n-1}$. Then, $wy \in L$. Indeed, by running A on wy we terminate in the state $r_n \in F$. Also, $|w| = |y|$ because for each step on the first track we make a step on the second. Therefore, $w \in L_{\frac{1}{2}}$.

($\Leftarrow$) Assume that $w \in L_{\frac{1}{2}}$. That is, there exists y such that $wy \in L$ and $|w| = |y|$. Denote the accepting run of A on wy by $r_0 r_1 \ldots r_{|w|} r_{|w|+1} \ldots r_{2|w|}$. Then the following sequence is an accepting run of A' on w:
$(r_0, r_{|w|}, r_{|w|}), (r_1, r_{|w|}, r_{|w|+1}) \ldots (r_{|w|}, r_{|w|}, r_{2|w|})$.