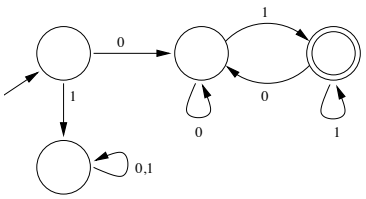
CS–172 Computability & Complexity, Spring 2021

Homework 1 Solutions

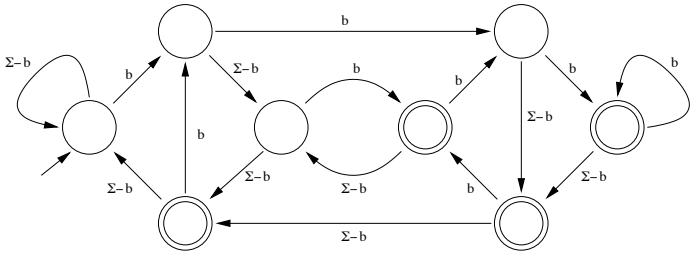
Note: These solutions are not necessarily model answers. Rather, they are designed to be tutorial in nature, and sometimes contain more explanation than is required for full points. Also, bear in mind that there may be more than one correct solution. The maximum total number of points available is 23.

1. DFAs for these three languages are as follows:

(a) The set of all 0-1 strings that begin with 0 and end with 1. *2pts*

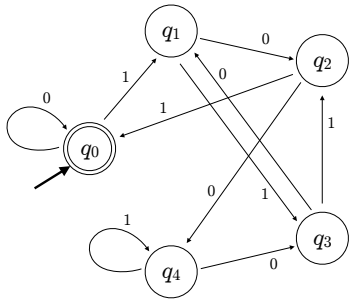


(b) The set of all words over the English alphabet whose third-last letter is 'b'. In the machine below, each state "remembers" the sequence of the last three symbols seen, distinguishing only between 'b' and non-'b' letters; thus it has eight states (corresponding to all 3-letter strings over an alphabet of two letters). Here Σ denotes the English alphabet. *2pts*



Note: As we know from the ideas in Note 1, any DFA for this language must have at least eight states!

(c) The set of all binary encodings of multiples of 5. In the machine below, the current state will be q_i if the portion of the input read so far is equal to i mod 5 (for $0 \leq i \leq 4$). Thus q_0 is the only accepting state. *2pts*



2. As suggested in the Hint, we describe the sets of input strings w that cause the DFA to end in each of the states (starting from the initial state q_0): *5pts*

- state q_0: all strings w in which every 0 is followed by a 1;
- state q_1: all strings w that end in 0 in which all other 0s are followed by a 1;
- state q_2: all strings w with a pair of consecutive 0's.

Note that these definitions cover all possible input strings, so all we have to do to show the correctness of the DFA is prove that if we run the DFA on a string w then it will terminate in state q_i **if** w satisfies the property ascribed to q_i above. (If some input strings were not covered, then we would have to prove the **only if** direction as well, as it might be possible for the DFA to accept more strings than we claim it does.)

The proof is by induction on the length of the input string w.

Base case: $|w|$ = 0, that is, $w = \epsilon$ (the empty string). The DFA terminates in state q_0, whose property is satisfied by the empty string: there are no 0's that are not followed by a 1. The definitions for the other two states are satisfied vacuously.

Induction step: We make the induction hypothesis that our definitions hold for $w' = \{0,1\}^n$ with $n \geq 0$; we now prove that they also hold for $w = w'a$ with $a \in \{0,1\}$. We consider the states in which the DFA might have determined after processing w':

- state q_0: If $a = 0$ then the DFA transitions to state q_1. By the induction hypothesis every 0 in w' is followed by a 1; $w'a$ now also ends in a 0 so the condition for q_1 is met. If $a = 1$ then the DFA stays in state q_0. Again by the induction hypothesis every 0 in w' is followed by a 1; this remains true with the addition of an extra 1.
- state q_1: If $a = 0$ then the DFA progresses to state q_2. By the induction hypothesis w' ends in 0; the addition of a gives the string a pair of consecutive 0's, and so the condition for q_2 is met. If $a = 1$ then the DFA transitions back to state q_0. As above, w' ends in 0 but otherwise has the properties of the strings that end up in q_0. Adding a 1 meets the requirement that every 0 in the whole string is followed by a 1, including the last 0.
- state q_2: By the induction hypothesis, w' already has a pair of consecutive 0's, so any string containing w' will have this property; therefore the DFA stays in state q_2 as it should.

We have now proved that the DFA will end up in state q_2 given any string w with a pair of consecutive 0's. Since the other two states are accepting, the DFA accepts iff w does not contain a pair of consecutive 0's.

Note: Some students had trouble with this inductive proof: note that, in addition to writing down the characterizations of each state, you also have to prove that these characterizations are correct by showing that they are maintained inductively at each step of the computation.

3. In all the parts below, we assume that $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA for the language L, and we explain how to modify M to construct an FA M' for the new language. We say that a state q_1 is *reachable* from a state q_2 in a FA M if there is a finite sequence of transitions that takes M from q_1 to q_2.

(a) Let $F \subseteq F' \subseteq Q$ be the set of states of M from which some state in F is reachable. Then M' is the same as M, except that all states in F' are now accepting states. (Note that M' remains a DFA.) *2pts*

To see that this construction is correct, note that a state q belongs to F' iff there exists a string $z \in \Sigma^*$ such that $\delta^*(q, z) \in F$. (Here we use δ^* to denote the common multi-step transition function of M

and M', as introduced in class.) But now, by construction,

$$M' \text{ accepts } w \Leftrightarrow \delta^*(q_0, w) \in F' \Leftrightarrow \exists z \in \Sigma^*(\delta^*(q_0, wz) \in F) \Leftrightarrow w \in \text{prefix}(L).$$

Note: Some students made all states in Q accepting, which is incorrect since M' could then accept strings that cannot be extended to a string in L.

(b) Add a new start state q_0' to M', and an ε-transition from q_0' to all states in Q that are reachable from q_0. Otherwise M' is the same as M. The idea here is that the ε-transitions effectively allow M' to jump to whatever state it could have been in after reading the missing initial segment of a word in L. (Note that M' is now an NFA.) *2pts*

To see that this construction is correct, let Q' denote the states reachable from q_0. Then

$$\begin{aligned} M' \text{ accepts } w &\Leftrightarrow (\exists q' \in Q')(\exists z \in \Sigma^*)((\delta^*(q_0, z) = q') \wedge (\delta^*(q', w) \in F)) \\ &\Leftrightarrow (\exists z \in \Sigma^*)(\delta^*(q_0, zw) \in F) \\ &\Leftrightarrow w \in \text{suffix}(L). \end{aligned}$$

Note: It is incorrect to simply add an ε-transition from the original start state q_0 to all states reachable from q_0. In this case, M' could use the ε-transitions multiple times by returning to q_0 during the computation. Recall how we used ε-transitions in our proofs of closure properties to make the constructions clearly correct. Another common mistake was to add ε-transitions from the new start state to every state in Q, rather than just to those states that are reachable from q_0.

(c) Construct M' by adding to M, for each accepting state $q \in F$ and each symbol $a \in \Sigma$, a transition from q to itself on input a. This means that, if M' ever reaches an accepting state on an initial segment x of the input, it will accept no matter what follows x. (Note that M' will now in general be an NFA.) *2pts*

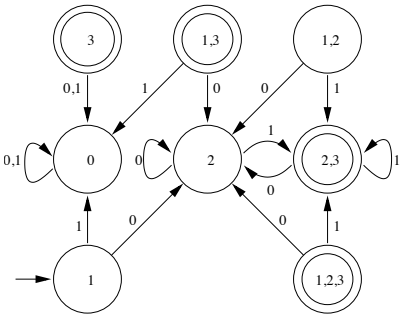
(d) Add a new start state q_0' to M'; also, add a transition from q_0' to itself on every input symbol $a \in \Sigma$, and an ε-transition from q_0' to q_0. The role of q_0' is to allow M' to ignore any initial segment y of w. (Again, M' is an NFA.) *2pts*

Note: In this case, unlike part (b), it is OK to simply add the self-loop transitions to q_0 itself rather than introducing the new start state. However, this requires more justification than the construction above, which is obviously correct. For the simpler construction, we have to check that no strings not in $\text{prepend}(L)$ are accepted, with the issue being possible returns to q_0 during the computation which allow M to use the self-loops multiple times. To check this claim, note that any accepting computation of M' must leave q_0 for the last time and end in an accepting state; we then identify x with this final portion of the input (which must belong to L since it takes M from q_0 to F) and y with the rest of the input. (If q_0 itself is an accepting state the same argument works with $x = \varepsilon$.)

4. The following DFA shows the full construction. *4pts*

First, note that it is easy to see that the given NFA accepts the same language as in problem 1(a). Moreover, if we prune away states unreachable from the start state in the above DFA, we end up with the same DFA as in problem 1(a). (The construction is not guaranteed to find such a simple DFA, but in this case it does.)

Note: The most common error here was to give incorrect transitions out of state $\{2, 3\}$ in the DFA. Make sure you understand these transitions!


3
1,3
1,2
0,1
1
0
0
1
0,1
0
0
2
1
2,3
1
0
1
0
0
1
1
1,2,3