CSE 103 Computational Models
Fall 2021 Prof. Fremont **HW 2**

Due October 13 at 11:59pm

(5 questions, 235 points total)

1. (50 pts.) Proving a DFA has a particular language

Suppose we have a robot which moves around in a 1D world, moving either left or right at each time step. We can represent its trajectory as a string over the alphabet $\Sigma = \{\leftarrow, \rightarrow\}$: for example the string $\leftarrow\leftarrow\rightarrow$ would mean that the robot moves left twice, then right once. Suppose that if the robot moves more than k steps left or right of its starting position, it will fall off a cliff and get stuck. Let L_k be the language of all trajectories where the robot does not fall; then for example L_1 would contain $\rightarrow\leftarrow$ but not $\rightarrow\leftarrow\rightarrow\rightarrow\leftarrow$, since in the latter trajectory the robot falls off the cliff after the 4th movement.

We can build a DFA recognizing L_k, but we can't draw it as a graph since k is arbitrary and we'll need a different DFA M_k for each $k \geq 0$. So we'll define $M_k = (Q_k, \Sigma, \delta_k, q_0, F_k)$ mathematically, as follows:

- $Q_k = \{q \in \mathbb{Z} \mid |q| \leq k+1\} = \{-(k+1), -k, \ldots, -1, 0, 1, \ldots, k, k+1\}$
- $\Sigma = \{\leftarrow, \rightarrow\}$
- $\delta_k(q,s) = \begin{cases} q+1 & |q| \leq k \text{ and } s = \rightarrow \\ q-1 & |q| \leq k \text{ and } s = \leftarrow \\ q & |q| > k \end{cases}$
- $q_0 = 0$
- $F_k = \{q \in \mathbb{Z} \mid |q| \leq k\} = \{-k, \ldots, -1, 0, 1, \ldots, k\}$

Intuitively, this DFA works by using states to keep track of where the robot currently is, updating the state at each time step and getting stuck in a non-accepting state if the robot ever moves too far from the start. Notice how we've used integers as states so that we can add or subtract 1 to them to represent the robot moving from one position to the next: you can use any objects as states in a DFA.

Lets formally verify this intuition, and use it to prove the correctness of our DFA.

(a) (40 pts.) Prove that $\widehat{\delta}_k(q_0, x)$ (the state reached when running M_k on input x) is the position of the robot after following the trajectory x, where we consider positions $k+1$ and $-(k+1)$ to indicate having fallen off the right and left cliffs respectively.
(Hint: Use induction on the length of x.)

Solution. *Proof:*
Base case if $x = \varepsilon$:

$$\widehat{\delta}_k(q_0, \varepsilon) = q_0 \tag{1}$$

If there are no moves then the robot's position is the same as the starting position. Which means that the extended transition function being the position of the robot after sequence of moves x holds. We now proceed with the assumption that $\widehat{\delta}_k(q_0, x)$ is the position after sequence of moves x. Then:

Inductive step:

$$\widehat{\delta}_k(q_0, xm) \text{ for any } x \in \Sigma^* \text{ and } m \in \Sigma \tag{2}$$

$$= \delta_k(\widehat{\delta}_k(q_0, x), m) \tag{3}$$

if we assume x to be a string of moves from Σ^* *then x can be* ε

$$= \delta_k(\widehat{\delta}_k(q_0, \varepsilon), m) \tag{4}$$

$$= \delta_k(q_0, m) \Rightarrow \text{ the position after taking move } m \text{ from } q_0 \tag{5}$$

Therefore the position after the sequence of moves xm can be represented by $\widehat{\delta}_k(q_0, xm)$*. Accordingly the position after the sequence of moves x can be represented by* $\widehat{\delta}_k(q_0, x)$

(b) (10 pts.) Using the result of part (a), prove that this DFA actually works, that is, that $L(M_k) = L_k$. (Hint: Use the lemma stated in class that M_k accepts x if and only if $\widehat{\delta}_k(q_0, x) \in F_k$.)

Solution. *Using the lemma from class* ($L(M_k) = \{x \in \Sigma^* | \widehat{\delta}_k(q_0, x) \in F_k)\}$*), and the fact that* $\widehat{\delta}_k(q_0, x)$ *results in the position after sequence x we can say that the lemma holds since:*

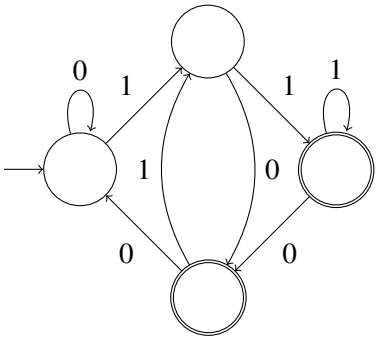
$$x = \varepsilon \tag{6}$$

$$\widehat{\delta}_k(q_0, \varepsilon) = q_0 \in F_k \tag{7}$$

If $x = \varepsilon$ *then the position will be* q_0 *which falls in the set of final states. Therefore* $L(M_k) = L_k$ *since the resulting state is in the language* L_k*.*

2. (35 pts.) Defining a DFA symbolically

For a given integer $k \geq 1$, let the language L_k consist of the strings over the binary alphabet $\Sigma = \{0,1\}$ whose kth symbol from the end is 1 (so for example $0100 \in L_3$). We saw an NFA for L_3 in class; we can also build a DFA for L_k (directly, without using the NFA-to-DFA conversion we'll cover later). For example, here is a DFA that recognizes L_2:



(a) (30 pts.) Define a DFA M_k recognizing L_k. Since k is arbitrary, you can't draw a graph; define the states Q_k, transition function δ_k, etc. mathematically, as we did in question (1). (You do not have to prove that your DFA recognizes L_k, but if you'd like more practice with induction, give it a try!)

(Hint: The states in the set Q don't have to be numbers; any distinct objects are fine. You could have states labeled by English words, for example, and then it would be well-defined to say something like "all states whose first letter is Z". So if it's convenient, feel free to have the "names" of the states store some useful information rather than just being numbers or letters.)

Solution.

$$Q = \{s_q \textit{ for some } q \in \Sigma^k\} \tag{8}$$

$$\Sigma = \{0,1\} \tag{9}$$

$$\delta_k(s_q, x) = \begin{cases} s_{q0} & \textit{if } q \in \{0,1\}^{k-1} \textit{ and } x = 0 \\ s_{q1} & \textit{if } q \in \{0,1\}^{k-1} \textit{ and } x = 1 \end{cases} \tag{10}$$

$$q_0 = s_{1^k} \tag{11}$$

$$F = \{s_z \textit{ for some } z \textit{ that starts with } 0\} \tag{12}$$

(b) (5 pts.) How many states does your automaton have?

Solution. *The automaton has 2^k states. with $\frac{2^k}{2}$ accepting states.*

3. (35 pts.) Does a DFA accept everything?

(a) (20 pts.) Describe an efficient algorithm to decide whether a DFA D accepts all strings, i.e. whether $L(D) = \Sigma^*$. You may use standard graph algorithms as subroutines.

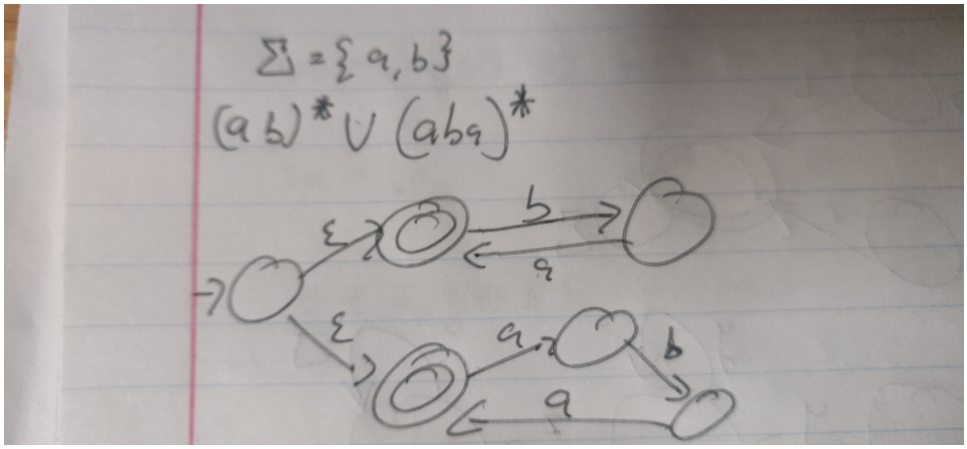
Solution. *The algorithm to check if a DFA accepts all strings is to check if the DFA contains loops, howver a DFA containing loops only has a posibility of accepting all strings so if we check to see if the DFA has no loops then we know that the DFA does not accept all strings.*

(b) (5 pts.) What is the asymptotic runtime of your algorithm, assuming D has n states and $|\Sigma| = m$?

Solution. *The runtime of the solution is $O(n)$ since we have to check every state in the DFA to see if it has a connection to an already checked state. Similar to Djikstra's algorithm we have to propogate through the entire graph. But we only check for loops which follows Floydd's Tortise and Hare algorithm for finding loops/cycles in a linked list.*

(c) (10 pts.) Would your algorithm also work for an NFA? If so, explain why. If not, give an example of an NFA where your algorithm returns the wrong answer.

Solution. *An NFA will have epsilon transitions and therefore have a loop and along with that if we take into account that there are more possibilites for loops with the fact that NFA's can have multiple paths from start to accepting state given an input the algorithm to check for no loops should fail. This does mean that the algorithm is not perfect for NFAs. An example would be an NFA with epsilon transitions to detect if a string contains the substring "ab" or "aba" as the graph can look like the following with loops but not accept all strings:*



(13)

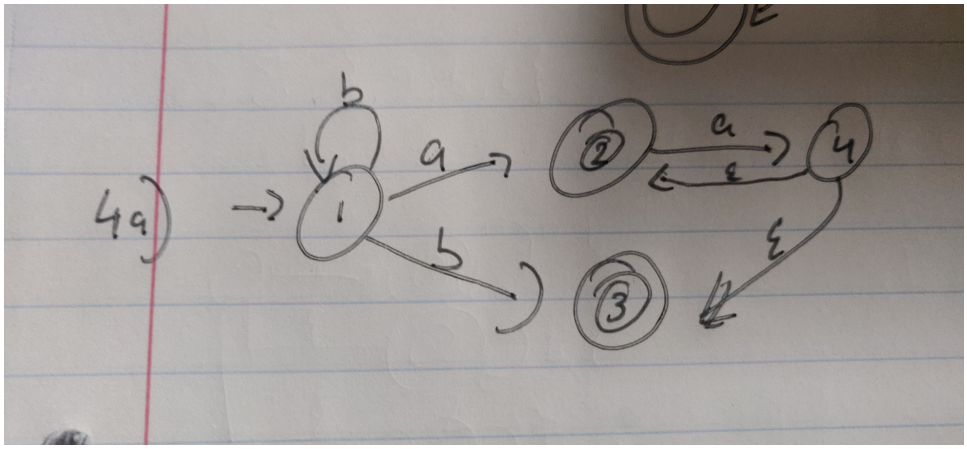
4. (55 pts.) Working with an NFA

Consider the NFA $M = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = \{1,2,3,4\}$
- $\Sigma = \{a,b\}$
- $\delta(q,s) = \begin{cases} \{2\} & q = 1 \text{ and } s = a \\ \{1,3\} & q = 1 \text{ and } s = b \\ \{4\} & q = 2 \text{ and } s = a \\ \{2,3\} & q = 4 \text{ and } s = \varepsilon \\ \emptyset & \text{otherwise} \end{cases}$
- $q_0 = 1$
- $F = \{2,3\}$

(a) (20 pts) Draw M as a graph.

Solution.



(14)

(b) (8 pts.) What are the ε-closures $E(\{1\})$, $E(\{2\})$, $E(\{4\})$, and $E(\{1,4\})$?

Solution.

$$E(1) \Rightarrow 1 \tag{15}$$
$$E(2) \Rightarrow 2 \tag{16}$$
$$E(4) \Rightarrow 2,3,4 \tag{17}$$
$$E(1,4) \Rightarrow E(1) \cup E(4) = 1,2,3,4 \tag{18}$$

(c) (15 pts.) Which of the following strings does M accept: $\varepsilon, aa, b, ba, aaabb$? Give an accepting path (just a sequence of states) for each accepted string.

Solution.

$$\varepsilon \Rightarrow (1,\varepsilon) \rightarrow ? \Rightarrow \textit{ Rejected} \tag{19}$$
$$aa \Rightarrow (1,a) \rightarrow (2,a) \rightarrow (4,\varepsilon) \rightarrow 2 \Rightarrow \textit{ Accepted} \tag{20}$$
$$b \Rightarrow (1,b) \rightarrow 3 \Rightarrow \textit{ Accepted} \tag{21}$$
$$ba \Rightarrow (1,b) \rightarrow (1,a) \rightarrow 2 \Rightarrow \textit{ Accepted} \tag{22}$$
$$aaabb \Rightarrow (1,a) \rightarrow (2,a) \rightarrow (4,\varepsilon) \rightarrow (2,a) \rightarrow (4,\varepsilon) \rightarrow (2,b) \rightarrow ? \Rightarrow \textit{ Rejected} \tag{23}$$

(d) (12 pts.) What is the language $L(M)$ of M?

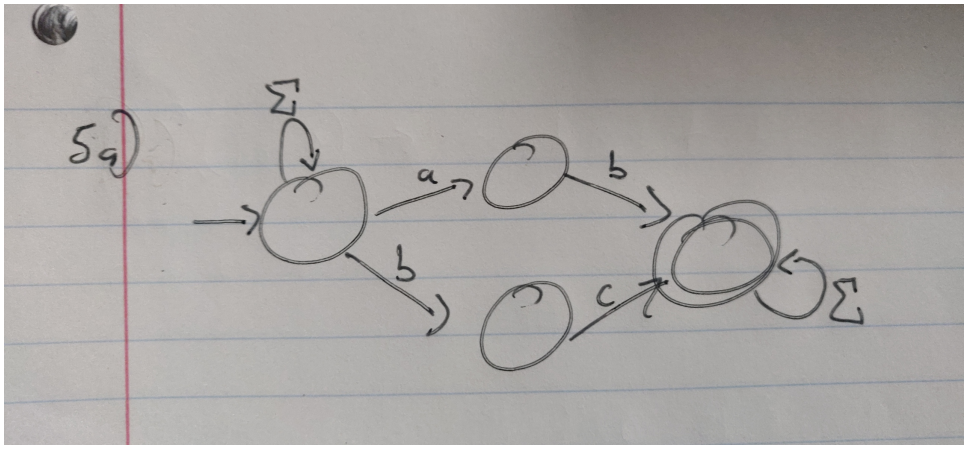
Solution. $L(M) =$ *any string that has zero or more b's followed by zero or more a's.*

5. (60 pts.) Designing NFAs

For each of the following languages, draw an NFA (as a graph) recognizing it. Your NFAs should have no more than 4 states.

(a) All strings over $\Sigma = \{a, b, c, \ldots, z\}$ containing either the substring ab or the substring bc.

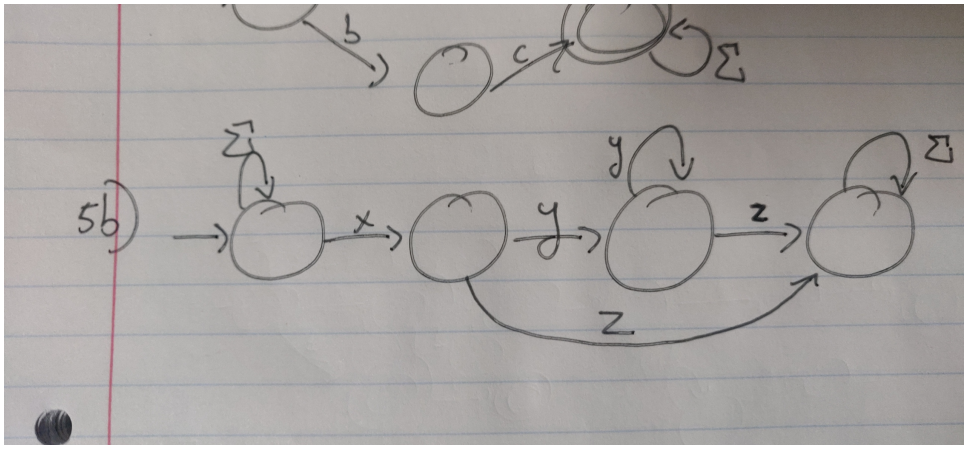
Solution.



(24)

(b) All strings over $\Sigma = \{a, b, c, \ldots, z\}$ which include a pair of x and z with only ys in between (e.g. $axyyzb$ and xz are OK but xaz and xy are not).

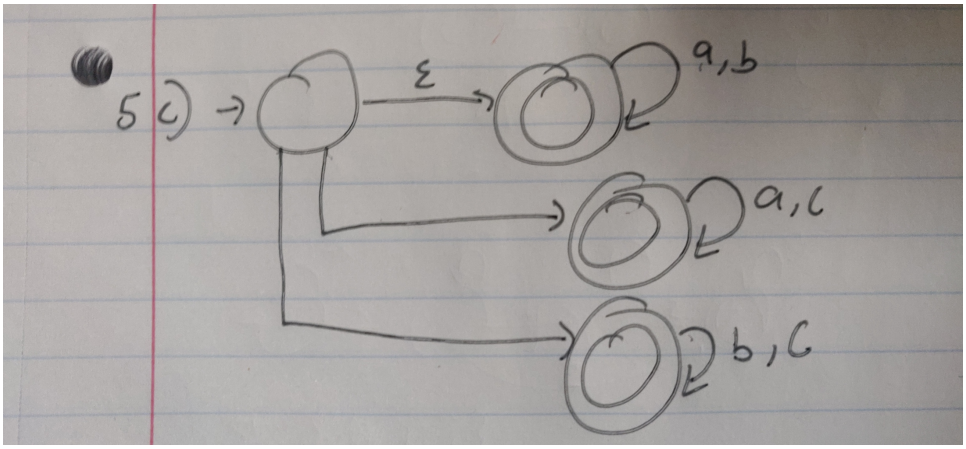
Solution.



(25)

(c) All strings over $\Sigma = \{a, b, c\}$ which do not contain all three symbols (e.g. *aab* and *bbba* are OK but *bca* is not).

Solution.



(26)