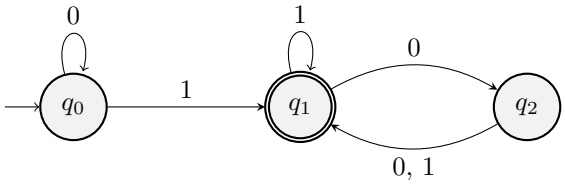
COMS 3261 Handout 2B: Deterministic Finite Automata Practice Solutions

Angel Cui and Owen Keith

Fall 2024

1

Determine which of ε, 11, 010, 10, 0101 is accepted by this DFA.

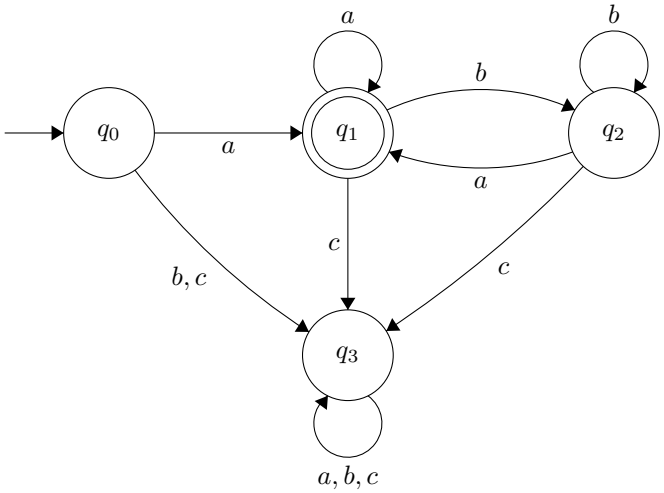


We start at the start state and "follow" the given string step-by-step through the DFA and check if it ends in an accept state. Specifically,

- ε is not accepted since the sequence of states in its computation is (q_0), and q_0 is not an accept state.
- 11 is accepted since the sequence of states in its computation is (q_0, q_1, q_1), and q_1 is an accept state.
- 010 is not accepted since the sequence of states in its computation is (q_0, q_0, q_1, q_2) and q_2 is not an accept state.
- 10 is not accepted since the sequence of states in its computation is (q_0, q_1, q_2) and q_2 is not an accept state.
- 0101 is accepted since the sequence of states in its computation is $(q_0, q_0, q_1, q_2, q_1)$ and q_1 is an accepting state.

2

The DFA state diagram below is defined on the alphabet $\Sigma = \{a, b, c\}$. Write out its formal definition (as a 5-tuple). When specifying the transition function δ, draw a table.



So we have $D = \{Q, \Sigma, \delta, q_0, F\}$ where:

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b, c\}$
- δ is described as

	a	b	c
q_0	q_1	q_3	q_3
q_1	q_1	q_2	q_3
q_2	q_1	q_2	q_3
q_3	q_3	q_3	q_3

- q_0 is the start state
- $F = \{q_1\}$ is the set of accept states

Note that q_3 is a dead/reject/bad state.[1] We could also not draw this q_3 state because for DFAs, we have a convention that when there are missing transitions, it means that they will go to a dead/reject/bad state. However, in the formal specification of a DFA, we have to include all states.

[1] A "dead" state is a state q that isn't an accept state, and for every symbol σ, we have $\delta(q, \sigma) = q$

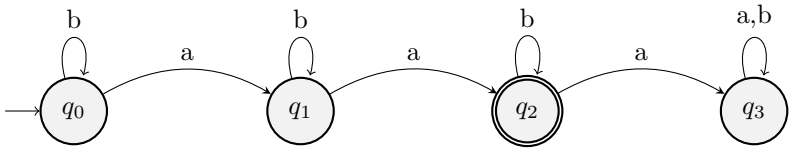
3

Draw DFAs that recognize the following languages:

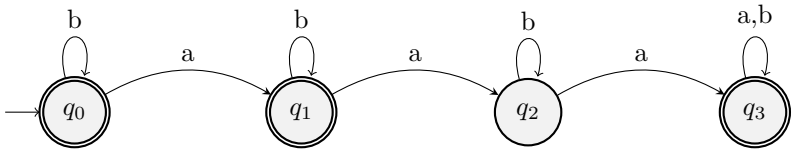
(a) $L = \{w \in \{a, b\}^* \mid w \text{ does not contain exactly two } a's\}$

We saw in class that the complement of a regular language is regular, i.e., if L is regular, then $\overline{L}$ is regular. Sometimes, it is easier to construct a DFA for $\overline{L}$, then by the construction we saw in class, flip all accept and reject states to arrive at a DFA for L.

We have $\overline{L} = \{w \in \{a, b\}^* \mid w \text{ contains exactly two } a's\}$, so a DFA for $\overline{L}$ would be as below. It is not hard to see that the following is a DFA for $\overline{L}$.



And so, the DFA for L would be:



(b) $L = \{w \in \{a, b\}^* \mid w \text{ has even length and an odd number of } a\text{'s}\}$

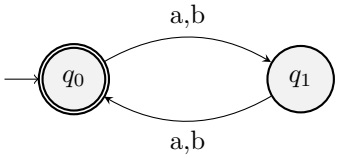
Here, L is the intersection of two simpler languages. Namely, $L = L_1 \cap L_2$ where

$$L_1 = \{w \in \{a, b\}^* \mid w \text{ has even length}\}$$

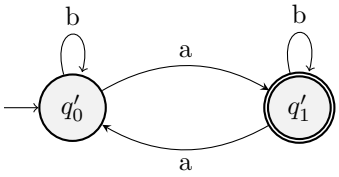
$$L_2 = \{w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s}\}$$

Let's construct the DFAs for L_1 and L_2, then apply the construction we saw in class to arrive at a DFA for L.

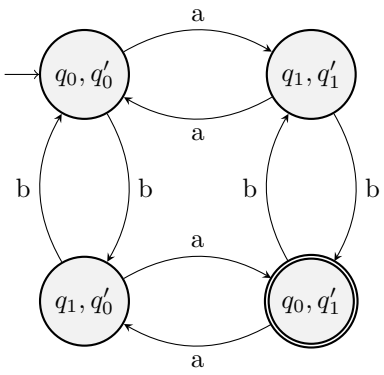
The DFA for L_1 is:



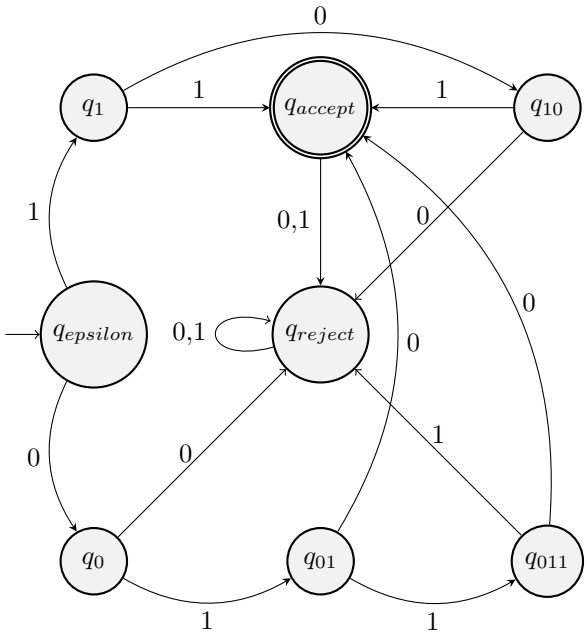
The DFA for L_2 is:



Now by the Cartesian product construction in class, the DFA for L would be:



(c) $L = \{11, 101, 010, 0110\}$



This DFA has a different state for each proper prefix of any string in the language (note that we can do that because the language is finite, so there are finitely many such prefixes), and an additional accepting state and a rejecting state. When the input is a string in L the DFA will end the computation in q_{accept}, and when the string is not in L it will end up in q_{reject}.

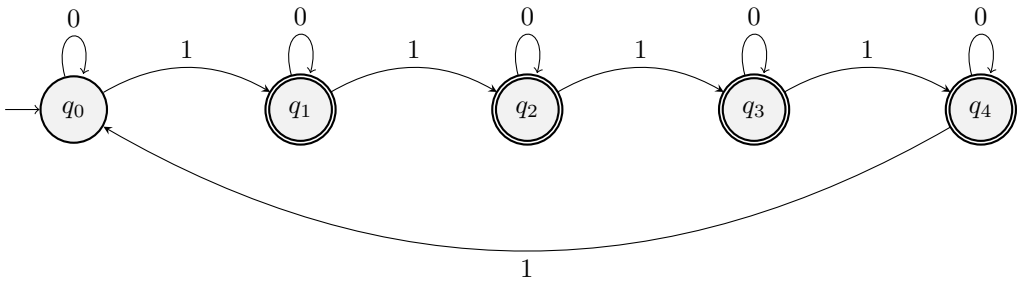
Here, q_{reject} is a dead/reject/bad state, which can also be omitted from the graph for simplicity. We could have removed q_{reject} and all transitions to it by convention mentioned above.

Additionally, you can always name the states in a way that can help make things more clear.

4

Given the DFA, specify the language it recognizes:

(a) DFA M:



How to approach this:

First, we can inspect which states are accept states.

Here, $\{q_1, q_2, q_3, q_4\}$ are the accept states.

We can then inspect some strings that are accepted by the DFA and some strings that are rejected by the DFA. For example, *0, 11111, 011111, 0101010101, 1111111111* are some of the strings that are rejected by this DFA. *01, 010, 011, 0101, 1, 10, 11, 111111111* are some of the strings that are accepted by the DFA.

We can find that the strings that are rejected by the DFA always have {0, 5, 10, 15, 20, ...} number of 1.

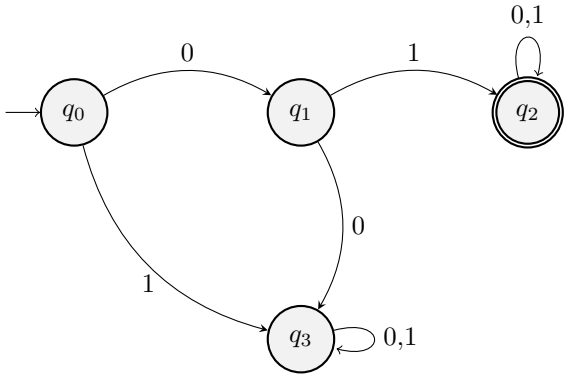
Solution:

This DFA recognizes the language below:

$L = \{w \in \{0,1\}^* | \text{the number of 1's in } w \text{ is not an integer multiple of 5}\}$

We can see that for any string, if the computation of M gets to q_i, it means we have seen i mod 5 1's so far. The only strings not accepted are the ones where the number of 1s is an integer multiple of 5 (Note that 0 is also an integer multiple of 5).

(b) DFA N:



How to approach this:

Similarly, we can first inspect which states are accept states.

Here, $\{q_2\}$ is the accept state. And we can easily tell that q_3 is a dead/reject/bad state.

We can then inspect some strings that are accepted by the DFA and some strings that are rejected by the DFA. For example, *0, 00, 1, 11, 000, 001, 100, 101* are some of the strings that are rejected by this DFA. *01, 010, 011, 0101* are some of the strings that are accepted by this DFA.

We can tell that the DFA is only accepting strings that start with a prefix *01.*

Solution:

This DFA recognizes the language below:

$L = \{w \in \{0,1\}^* | w \text{ starts with the prefix } 01\}$

Any string that starts with *01* will have it's computation start with the states $(q0, q1, q2)$ and then remain in $q2$ for the rest of the string, thus it is accepted. Any string that does not start with *01*, either starts with *1* , and then its computation goes from $q0$ to $q3$ and stays in $q3$; or starts with *00* and then the computation starts with $(q0, q1, q3)$, and then stays in $q3$.

Note:

Note that here q_3 is a dead/reject/bad state and can be omitted like below: