# CS340: Theory of Computation
# Assignment 1 Solutions

Yatharth Goswami
Roll No. 191178

September 2, 2021

# I Problem 1 Solution

In this problem, we were asked to design DFA for each of the languages provided.

(a) This part required us to construct a DFA for a language which accepts strings with alternating $a$ and $b$ with atleast two $a'$s.

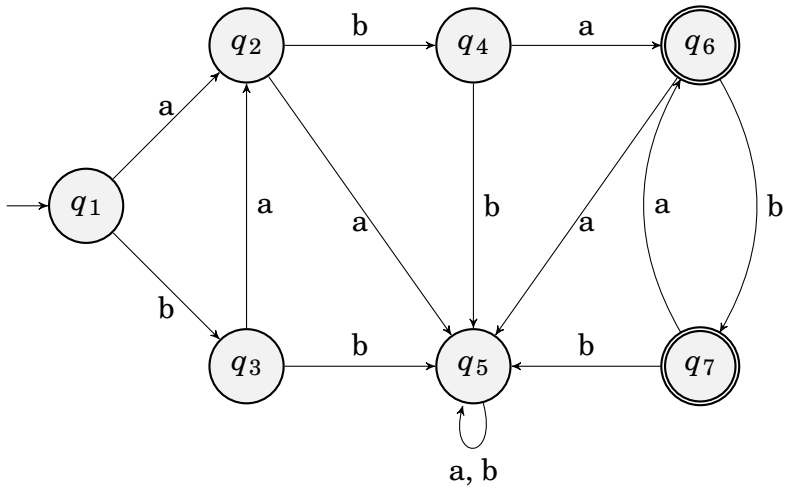


Figure 1: DFA for 1(a)

(b) This part required to construct a DFA for a language which accepts strings containing "ababb" as a substring.

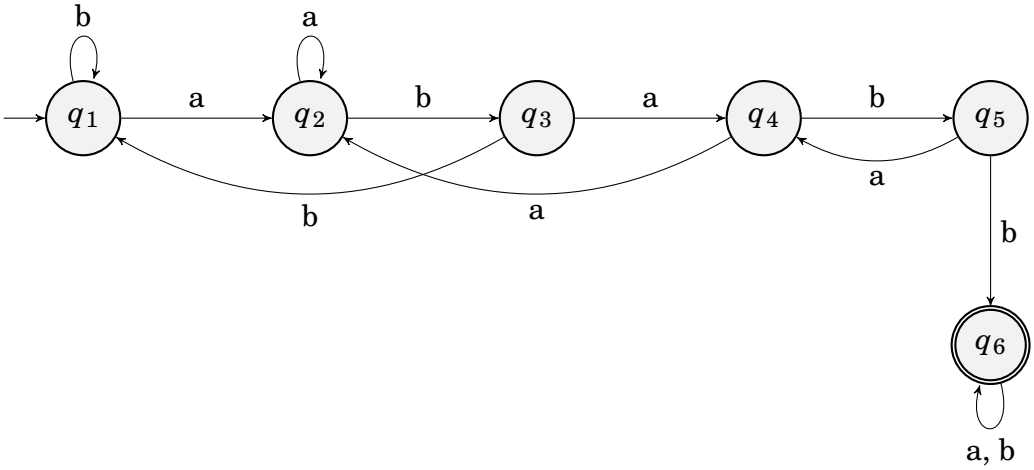


Figure 2: DFA for 1(b)

(c) This part required to construct a DFA for a language which accepts strings having atmost 2 occurences of 3 consecutive 1's.

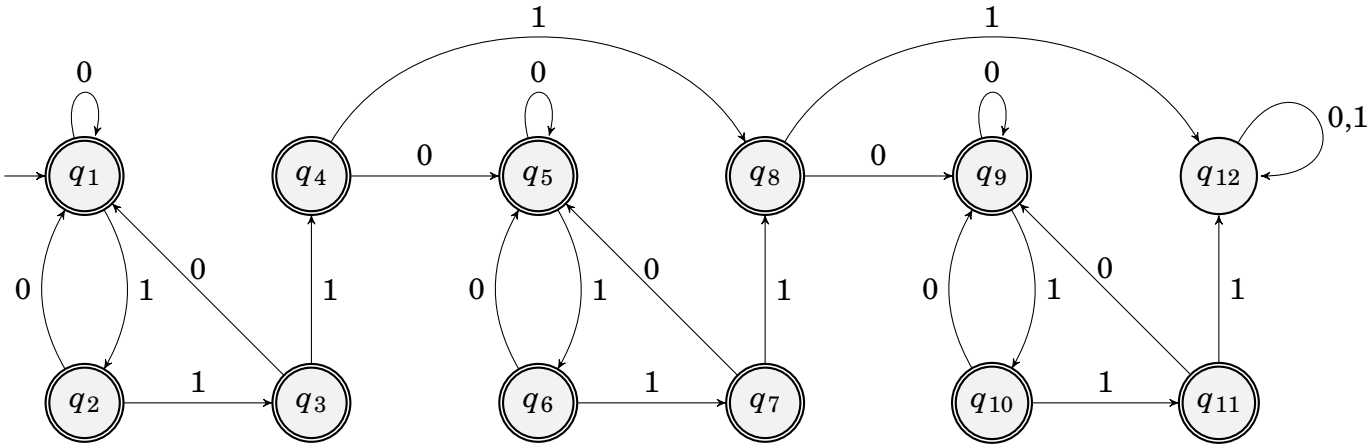


Figure 3: DFA for 1(c)

# II Problem 2 solution

This problem required us to provide the minimum state DFAs for the following regular languages.

(a) $\epsilon + (0+1)0(0+1)^*$

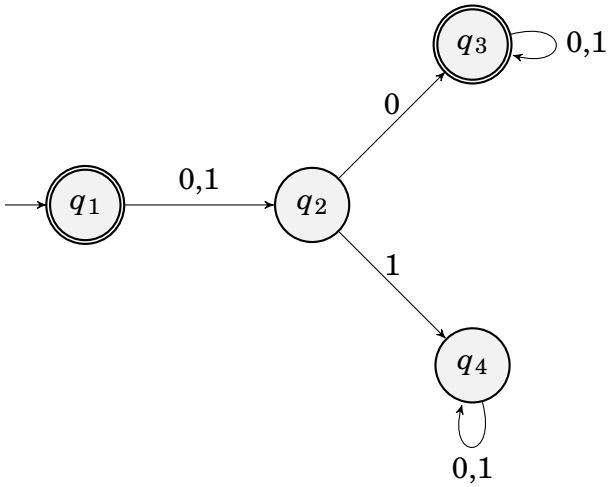


Figure 4: DFA for 2(a)

(b) $a^*b^* + b^*a^*$

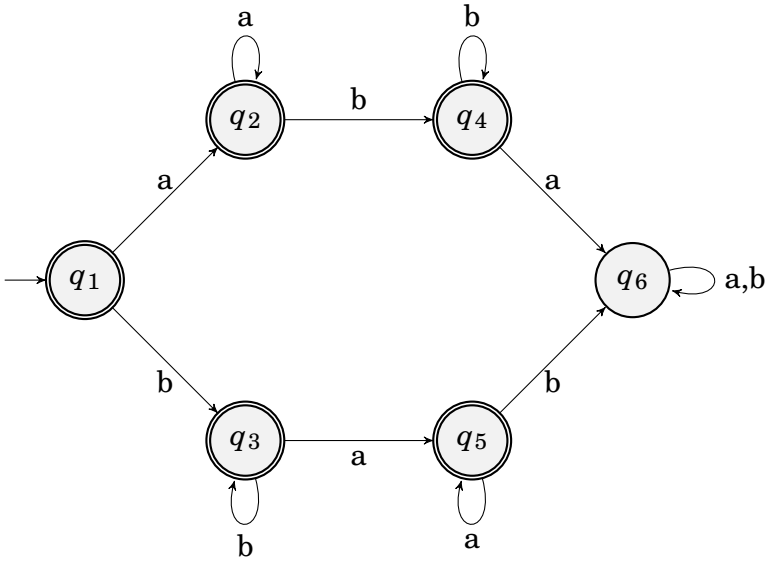


Figure 5: DFA for 2(b)

# III Problem 3 Solution

In this problem, we were given two regular languages $A$ and $B$ and were asked to prove that

$f(A,B) = \{w \in \Sigma^* | w = a_1b_1a_2b_2\ldots a_kb_k,$ where $a_1, a_2, \ldots, a_k \in A$ and $b_1, b_2, \ldots, b_k \in B$ and each $a_i, b_i \in \Sigma^*\}$

is also regular.

*Proof.* We will show that $f(A,B)$ is regular by constructing an NFA that accepts this language. By equivalence between NFA and DFA we can say that if there exists a NFA that accepts a certain language than there exists a corresponding DFA for it as well and hence the language will be regular. Since the languages A and B are regular, this implies that there will exist a NFA for both of them. Let's call them $N_1$ and $N_2$. Consider the NFA shown in figure 6, let's call it $N$. The red and blue boxes denote the NFAs $N_1 = (Q_1, \Sigma, \delta_1, q_1, q_{f1})$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, q_{f2})$ (with unique start and accept state). Let us first define the NFA $N$ formally. Let us denote $N$ by a tuple $(Q, \Sigma, \delta, q_0, F)$ where:

1. $Q = Q_1 \cup Q_2 \cup \{q_0, q_{acc}\}$
2. (a) $\delta(q_0, \epsilon) = \{q_1\}$
   (b) $\delta(q_0, a) = \{\}$ for all $a \in \Sigma$
   (c) $\delta(q_i, a) = \delta_1(q_i, a)$ for all $a \in \Sigma_\epsilon$ and $q_i \in Q_1 - \{q_{f1}\}$
   (d) $\delta(q_{f1}, \epsilon) = \delta_1(q_{f1}, \epsilon) \cup \{q_2\}$
   (e) $\delta(q_{f1}, a) = \delta_1(q_{f1}, a)$ for all $a \in \Sigma$
   (f) $\delta(q_j, a) = \delta_2(q_j, a)$ for all $a \in \Sigma_\epsilon$ and $q_j \in Q_2 - \{q_{f2}\}$
   (g) $\delta(q_{f2}, \epsilon) = \delta_2(q_{f2}, \epsilon) \cup \{q_1, q_{acc}\}$
   (h) $\delta(q_{f2}, a) = \delta_2(q_{f2}, a)$ for all $a \in \Sigma$
   (i) $\delta(q_{acc}, a) = \{\}$ for all $a \in \Sigma_\epsilon$
3. $F = \{q_{acc}\}$

We will prove that NFA $N$, accepts the language $f(A,B)$. Let us prove this claim formally. Let the set of strings accepted by $N$ be denoted by $L(N)$. We will show that $L(N) = f(A,B)$.

**Claim 3.0.1.** $f(A,B) \subseteq L(N)$

*Proof.* For proving the above claim, we need to show that every string in $f(A,B)$ is accepted by $N$. Consider any arbitrary string from $f(A,B)$, say $w = a_1b_1a_2b_2\ldots a_kb_k$. To show that this string gets accepted by the NFA, we will follow it's path in $N$. When $a_1$ will appear, the NFA will be at state $q_1$ and after $a_1$ is read, it will reach the state $q_{f1}$ since $a_1 \in A$ and when $b_1$ is read it will be at state $q_2$ and after reading $b_1$, it will be at the state $q_{f2}$, from which it uses $\epsilon$ transitions to go to $q_{ac}$ and $q_1$. Similarly, DFA accepts every pair of strings of the form $a_ibj$ where $a_i \in A$ and $b_j \in B$. So, we can say

inductively that if the NFA accepts the string $a_1b_1 \ldots a_{k-1}b_{k-1}$ and it accepts the string $a_1b_1 \ldots a_kb_k$ (by doing an $\epsilon$ transition to $q_1$ and reaching $q_{ac}$ after reading $a_kb_k$). Hence proved that $f(A,B) \subseteq L(N)$. □

**Claim 3.0.2.** $L(N) \subseteq f(A,B)$

*Proof.* We will see some properties of strings which are accepted by $N$. First there is no $w \in L(N)$ such that no prefix of $w$ is accepted by $A$. If we consider there does exist such a string, it would have never reached $q_{f1}$ and hence never $q_{acc}$. Hence, all strings $w$ have some prefix accepted by $A$. Now, considering the remaining part of the string leaving aside this prefix, we can say that there exists no $w$ such that no prefix in the rest of the part is accepted by $B$. If we consider, there does exist such a string, after reaching $q_{f1}$ it would have never reached $q_{f2}$ and hence $q_{acc}$. Hence, all string $w$ have some prefix of the form $a_1b_1$ where $a_1 \in A$ and $b_1 \in B$. Now, either $w$ ends here ($w = a_1b_1$) or it takes an $\epsilon$ transition back to $q_1$. In the first case we are done and in the second case we can recursively comment on the rest of the string being accepted by $N$. This leads to us finding the fact that all of these $w$ can be written in the form of $a_1b_1a_2b_2 \ldots a_kb_k$ for some appropriate value of $k$. Hence proved that $L(N) \subseteq f(A,B)$. □

Using the above two claims, we can prove that both the sets have to be equal, or $L(N) = f(A,B)$. Hence, we found an NFA that accepts $f(A,B)$ and therefore it is regular. □

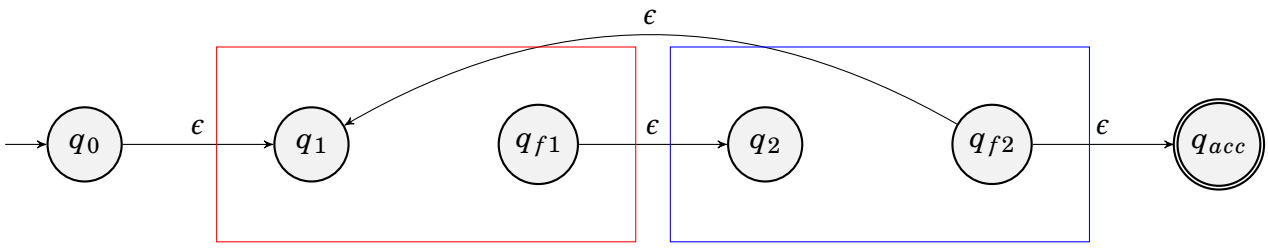


Figure 6: NFA for 3

# IV Problem 4 Solution

In this problem, we were asked to construct a minimum state DFA from the provided one. We will foloow the steps in the DFA minminsation algorithm to do that.

1. The first step in the algorithm is removing the unreachable states. We can see that the states $q_4, q_5, q_6, q_7$ will be removed in this way.

2. The next step in the algorithm involves removing the dead states, one from which any accept state can't be reached. There is just one accept state here and it can be reached from all the other states. Hence, no state will get removed in this step.

3. This step involves the merging of non-distinguishable states. We will follow the algorithm described in the class for this step.

   - The first part involves marking the pair of vertices with one finish state and one non-finish state. In this iteration the vertices $(q_0, q_3)$,$(q_1, q_3)$,$(q_2, q_3)$ will get marked and the state of the table will look like this.

     | | | | |
     |---|---|---|---|
     | $q_0$ | | | |
     | - | $q_1$ | | |
     | - | - | $q_2$ | |
     | X | X | X | $q_3$ |

   - The next part involves iterating over the non-marked states and marking them if one of the transitions lead to a pair of states which are already marked. Checking for the pair $(q_0, q_1)$ first. We see that both 0 and 1 will not lead to transitions to marked pair of states and hence this state will remain unmarked. However, the states $(q_0, q_2)$ and $(q_1, q_2)$ will go to an already marked pair on 0. Hence, both of these will get marked in this iteration. The state of the matrix after this iteration will be.

     | | | | |
     |---|---|---|---|
     | $q_0$ | | | |
     | - | $q_1$ | | |
     | X | X | $q_2$ | |
     | X | X | X | $q_3$ |

     In the next iteration of the algorithm, the state pair $(q_0, q_1)$ will also get marked because of the transition on symbol 1. Hence, the final state of the matrix will become.

     | | | | |
     |---|---|---|---|
     | $q_0$ | | | |
     | X | $q_1$ | | |
     | X | X | $q_2$ | |
     | X | X | X | $q_3$ |

     Since, all of the state pairs are marked therefore none of the pairs are non-distinguishable. Hence the algorithm ends here.

The final minimized DFA will be

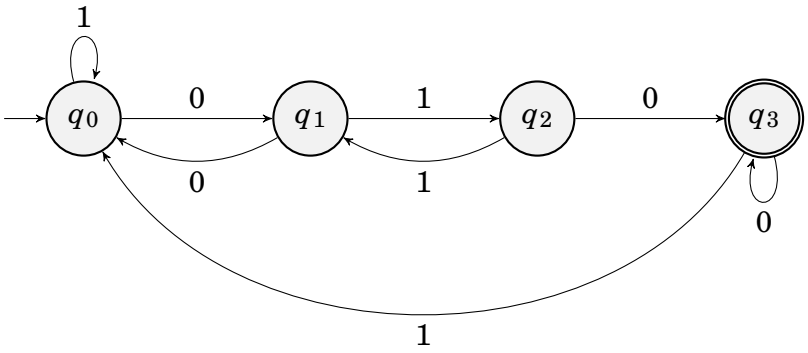


Figure 7: Minimised DFA for 4