CSE 105, Fall 2019 - Homework 2 Solutions

Due: Monday 10/21 midnight

Instructions

Upload a single file to Gradescope for each group. All group members' names and PIDs should be on each page of the submission. Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

For questions that only ask for diagrams, justifications are not required but highly recommended. It helps to show your logic in achieving the answers and partial credit can be given if there are minor mistakes in the diagrams.

Reading Sipser Sections 1.1 - 1.3

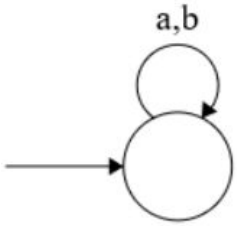
Key Concepts Deterministic finite automata (DFA), state diagram, computation trace, accept / reject, language of an automaton, regular language, union of languages, concatenation of languages, star of a language, closure of the class of regular languages under certain operations, nondeterministic finite automata (NFA), nondeterministic computation, ε arrows, equivalence of NFAs and DFAs.

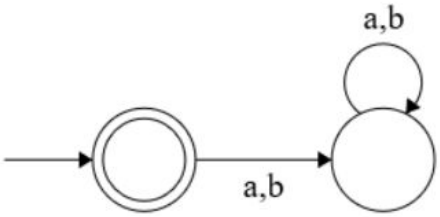
Problem 1 (10 points)

For each of the below parts, draw the minimal state diagram of the DFA that recognizes the given language.

a. $L = \textit{the empty language } \emptyset$ with $\Sigma = \{a, b\}$

Common Mistake: Using extra states/epsilon transition/accept empty string

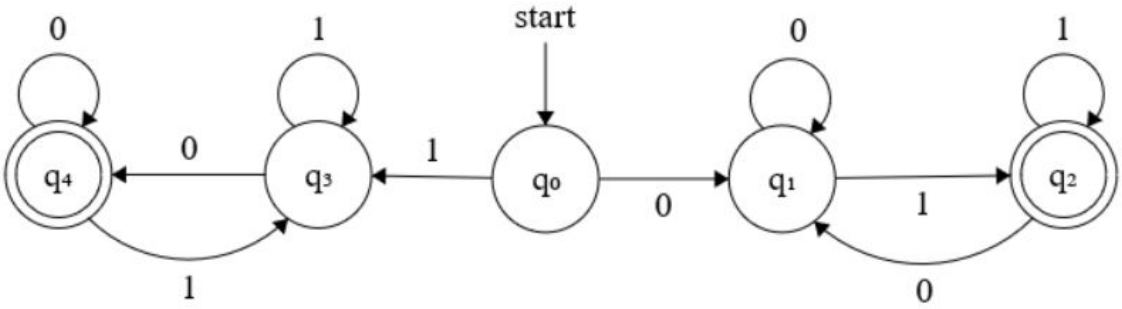




b. $L = \textit{the language that accepts only the empty string } \varepsilon$ with $\Sigma = \{a, b\}$

Common Mistake: transition to different states with a,b from start state, not optimal

c. $L = \{w \in \Sigma^* \mid w \textit{ does not contain an equal number of occurrences of the substrings } 01 \textit{ and } 10\}$ with $\Sigma = \{0, 1\}$ Common Mistake: Using extra states; transitions from accept states to start state; epsilon transition



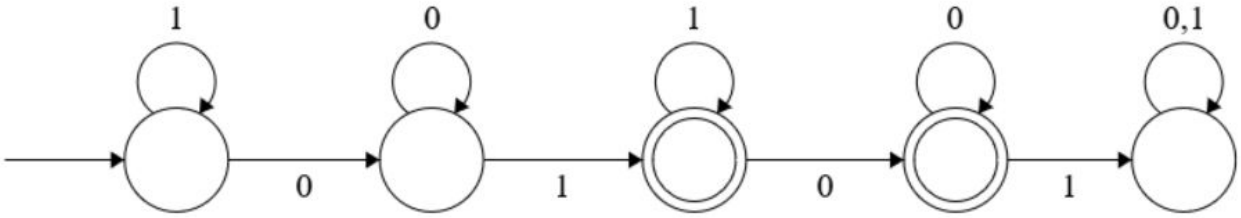
Problem 2 (10 points)

(a) Draw the state diagram of the DFA that recognizes the language

$L = \{w \in \{0,1\}^* \mid w \textit{ contains exactly one occurrence of the substring } 01\}$

For full credit your DFA must have no more than five states.

Common Mistake: DFA not accepting strings in the form of 1*0*1*0*;

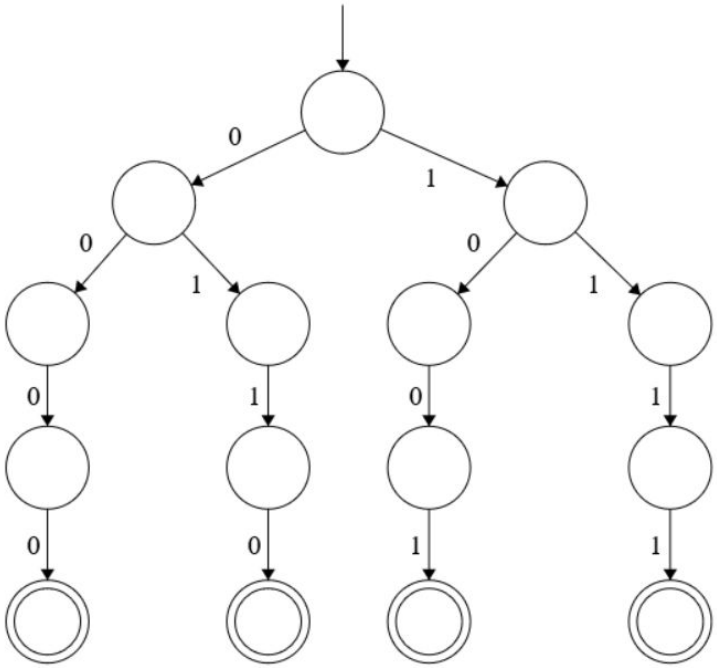


b. Draw the state diagram of the NFA that recognizes the language

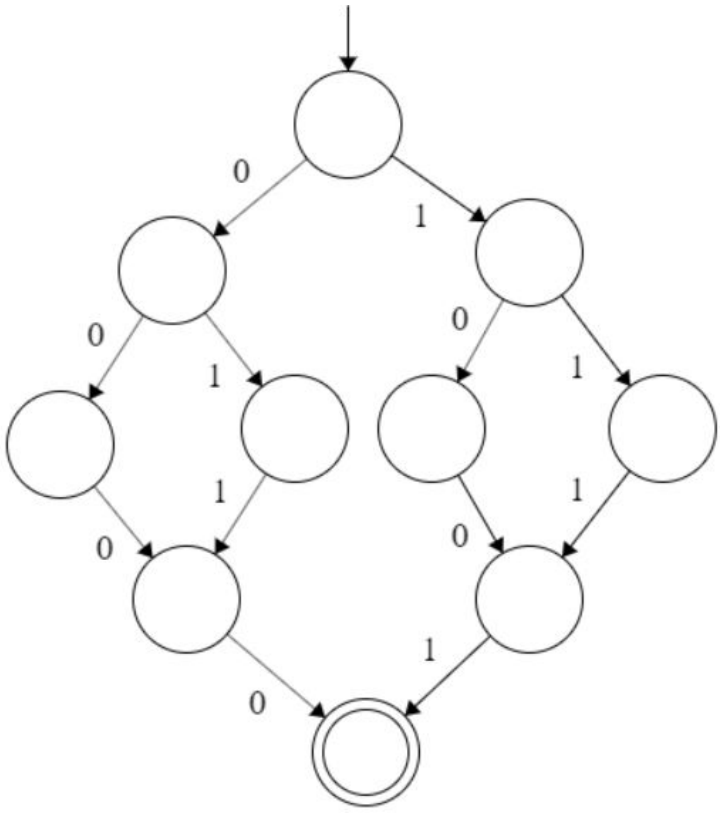
$L = \{w \in \Sigma^* \mid w \textit{ is a palindrome of length } 4\}$

For full credit your NFA should have no more than fifteen states and the minimal number of transitions in the diagram.

Common Mistake: extra epsilon transitions; extra 0,1 transitions to trap state; intersection of the 4 palindrome paths causing NFA accepting non-palindrome strings



This is an alternative answer to b with only 10 states.

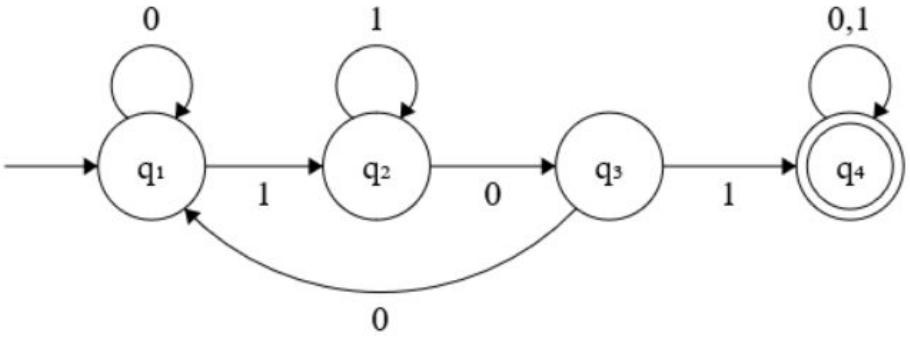


Problem 3 (10 points)

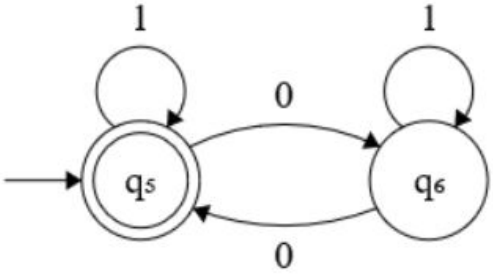
Recall, for a language $L \subseteq \Sigma^*$ its complement is the set of strings over Σ not in L, denoted as $\overline{L} = \{w \notin L\} \subseteq \Sigma^*$. Also, recall that the set difference is defined as $L_1 - L_2 = \{w \mid w \in L_1,\ w \notin L_2\}$

$$A = \{w \in \{0,1\}^* \mid w \textit{ contains } 101 \textit{ as a substring}\}$$
$$B = \{w \in \{0,1\}^* \mid w \textit{ has an even number of zeros}\}$$

A

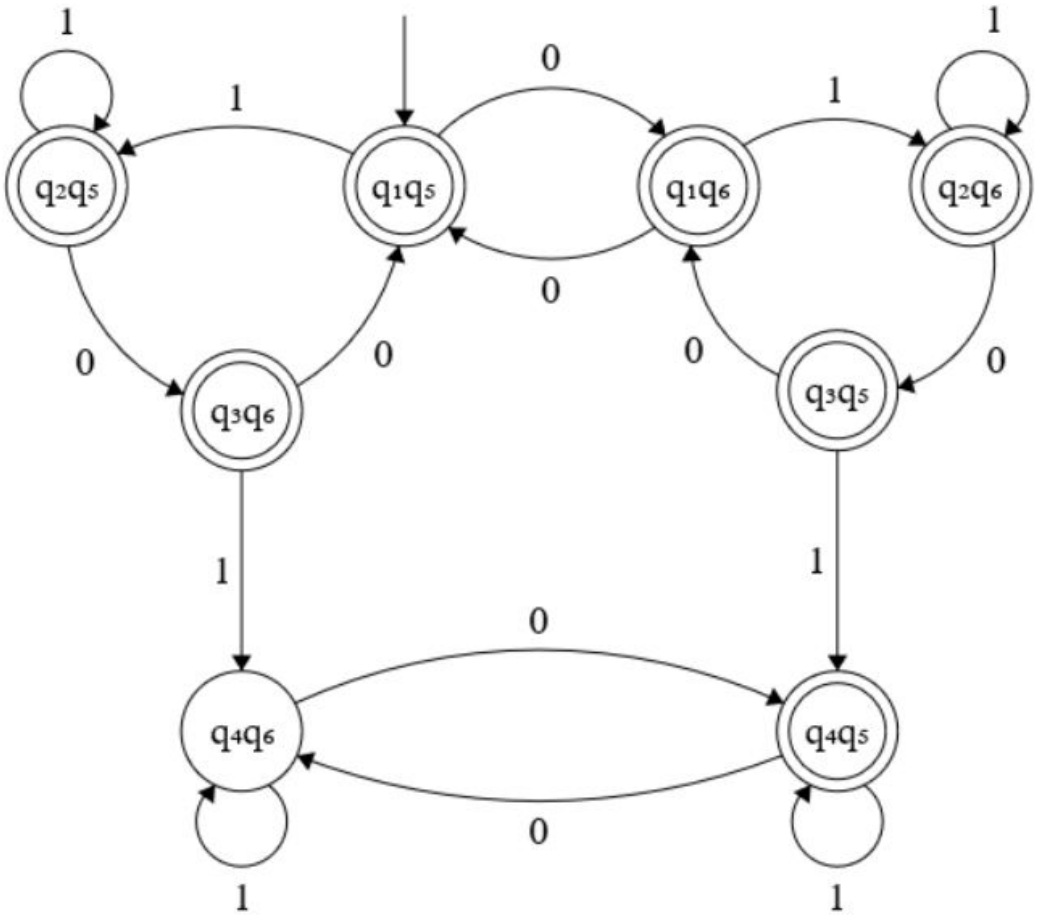


B



(a) Draw the state diagram of the DFA of the following language: $\overline{A} \cup B$

For full credit, each DFA should have no more than 8 states.

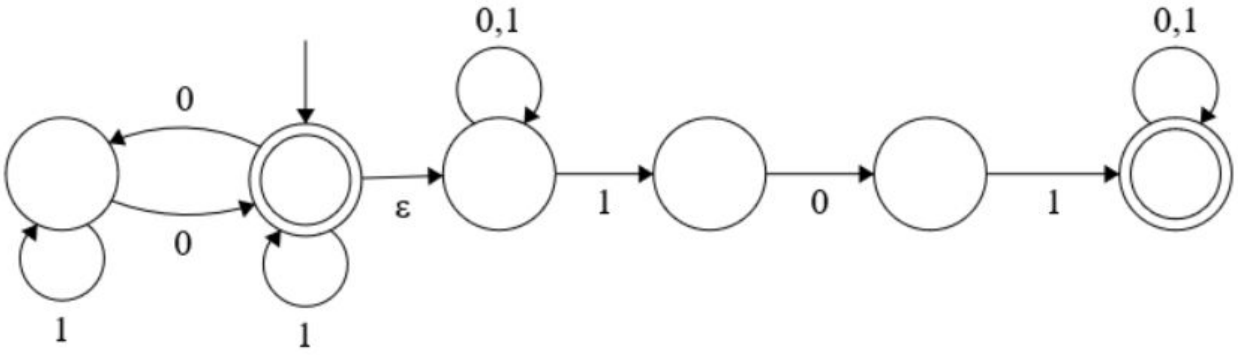


Common Mistake: The language can accept input strings which have 101, as long as the overall string has an even number of 0s.

(b) Draw the state diagram of the NFA of the following languages: (A)* ◦ B
For full credit your NFA should have no more than six states and the minimal number of transitions in the diagram.

The first thing to note is that in this case $A^* = A \cup \{\varepsilon\}$. For a language to be in A^* it must contain at least one occurrence of the string 101. If we concatenate several of these strings together (i.e. apply the Kleene star), we are still guaranteed to have at least one occurrence of 101 in the resulting string, and thus this string must have been in A to begin with.

Because both languages now recognise the empty string, we can rewrite $A^* \circ B$ as $A \cup B$.



Common Mistakes: a) The Kleene star operation allows you to have the input as an empty string for A. This means that the start state has to be shifted.
b) You can't have your start state at the start state of the NFA for A, and then make an ε transition to the start state of the NFA for B. This is because the first state of A will have the 0,1 transition onto itself, and the start state of B is an accept state. This would then cause any input string to be accepted.

Problem 4 (10 points)

Prove that any finite language (i.e. a language with a finite number of strings) is regular

Proof by Induction:

First we prove that any language L = {w} consisting of a single string is regular, by induction on |w|. (This will become the base case of our second proof by induction)

Base case: |w| = 0; that is, w = ε
In problem 1(b), we constructed a DFA that recognizes the language that contains only the empty string, and thus this language is regular.

Induction:
Let L be a language that recognizes a single string w over Σ. We can rewrite $w = w_1 w_2 ... w_n$ such that $w_i \in \Sigma$ for all i.

Suppose that a DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ exists that recognizes $L = \{w = w_1 w_2 ... w_n\}$. By definition L is regular. We must now prove that the language $L' = \{w' = w\Sigma = w_1 w_2 ... w_n w_{n+1}\}$ is regular.

Define $M' = \{Q', \Sigma, \delta', q_0', F'\}$ to be an NFA such that:

$Q' = Q \cup \{q_{n+1}\}$

$\delta'(q, c) =$

$\{\delta(q, c)\}$ if $q \notin F$

$\{q_{n+1}\}$ if $q \in F$ and $c = w_{n+1}$

$q_0' = q_0$

$F' = \{q_{n+1}\}$

We propose that M' recognizes L'. Since the first part of δ' follows the same path that w would take through L, we know that reading w will terminate at some state $q \in F$. Q' added a single state that is only reachable from the states in F on character w_{n+1}. If there are any additional characters to read, we will reject the string since q_{n+1} has no outgoing transitions. If we try to read a prefix of the string that does not equal w exactly, we will never end up in a state in F and thus never reach q_{n+1}. Thus, M' is an NFA that recognizes L' and by definition L' is regular.

Next, we prove that any finite language L is regular by induction on |L|. We prove the statement P(n) = “all languages of size n are regular”

Base Case: The previous proof handled the base case where |L| = 1. We also note that the empty language L = {} is regular by referring to the DFA in Problem 1(a).

Inductive Step: Suppose that we have proven that any language of size *n* is regular (we assume P(n) as our inductive hypothesis). Let L be a finite language containing *n+1* strings. Choose any string $s \in L,$ and let $L' = L - \{s\}$ be simply all of the strings in L except *s*. We can express L as the union $L' \cup \{s\}$. Furthermore, |L'| = n and |{s}| = 1 so by the inductive hypothesis L' is regular and by the base case {s} is regular. We know from lecture that the class of regular languages are closed under union, and thus L is regular. This proves P(n+1) and thus completes our proof by induction.

Common Mistakes: In proving all finite languages are regular, using induction on the size of the language |L|, some groups gave the wrong base case. The language of size zero, namely {}, cannot be used as a base case because the inductive step explicitly uses the fact that a language of size one is regular when break up L into two or more smaller languages.

Also, many groups that did include the correct base case neglected to give a justification as to why a language of size 1 is regular or proved it only for the language $\{\varepsilon\}$. A valid justification could be an inductive proof like the first proof above, a direct construction of a DFA/NFA, or an appeal to regular expressions and their properties.

Problem 5 (10 points)

Prove that regular languages are closed under intersection. That is, given two regular languages L_1 and L_2, prove that $L_1 \cap L_2$ is regular.

Proof via closure under complement and union

Note that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
We previously proved (in lecture and in the textbook) that languages are closed under complement and union. Thus

$\overline{L_1}$ is regular since regular languages are closed under complement
$\overline{L_2}$ is regular since regular languages are closed under complement
$\overline{L_1} \cup \overline{L_2}$ is regular since regular languages are closed under union

$\overline{\overline{L_1} \cup \overline{L_2}}$ is regular since regular languages are closed under complement.

$\therefore$ $L_1 \cap L_2$ is regular

Proof via finite automata construction

Let M', M'', and M be the formal definition of the finite automata that recognizes L_1, L_2, and $L_1 \cap L_2$ respectively.

$M' = (Q', \Sigma, \delta', q_0', F')$
$M'' = (Q'', \Sigma'', \delta'', q_0'', F'')$

Then
$M = (Q' \times Q'', \Sigma, \delta, (q_0', q_0''), F' \times F'')$
$\delta((q', q''), c) = (\delta'(q', c), \delta''(q'', c))$ where c is the current character in the string

To prove that M recognizes $L_1 \cap L_2$, let w be a string over Σ such that $w \in L_1$ and $w \in L_2$. Therefore:

$\delta^{*}'(q_0', w) \in F'$
$\delta^{*}''(q_0'', w) \in F''$
$\delta^{*}((q_0', q_0''), w) \in F' \times F''$

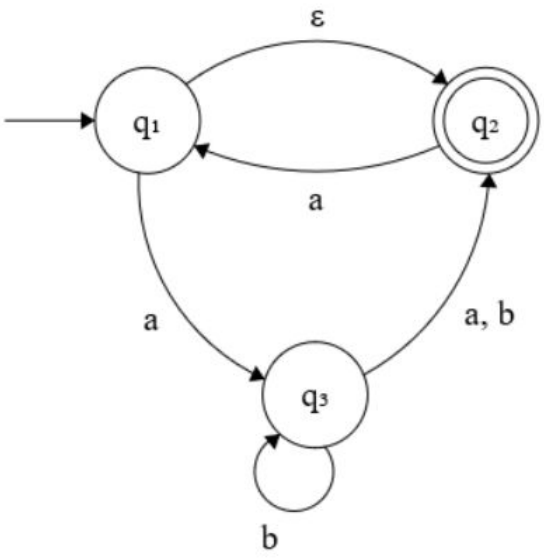
which is equivalent to the set of accept states in M.

$\therefore$ Since we can construct a finite automata for $L_1 \cap L_2$ it is regular.

Common Mistake: L1 and L2 are regular, but that does not mean that L1 and L2 are finite (all finite languages are regular but not all regular languages are finite!)

Problem 6 (10 points)

Given the following state diagram of an NFA over the alphabet Σ = {a, b}, convert it into the state diagram of its equivalent DFA. Give an informal description in English of what language these finite automata recognize. For full credit, your DFA should have no more than 8 states.



Answer:
The language recognized by these finite automata is all strings that don't begin with *b* and don't contain the substring *bab*.

We first create a table that shows the set of states we can reach after reading a single character from the specified beginning state. The ε^* column represents the set of all states we can reach by reading zero or more ε characters from the specified beginning state.

	a	b	ε^*
q_1	$\{q_3\}$	Ø	$\{q_1, q_2\}$
q_2	$\{q_1\}$	Ø	$\{q_2\}$
q_3	$\{q_2\}$	$\{q_2, q_3\}$	$\{q_3\}$

The new states Q' are the powerset of the set of states in the original NFA:
$Q' = \{\emptyset, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$.

The new start state q_0' is the set of all states that can be reached from the original start state by reading zero or more ε characters. In this case $q_0' = \{q_1, q_2\}$.

The new accept states F' are a subset of Q' such that at least one element q' of each set in F' matches one of the following criteria:

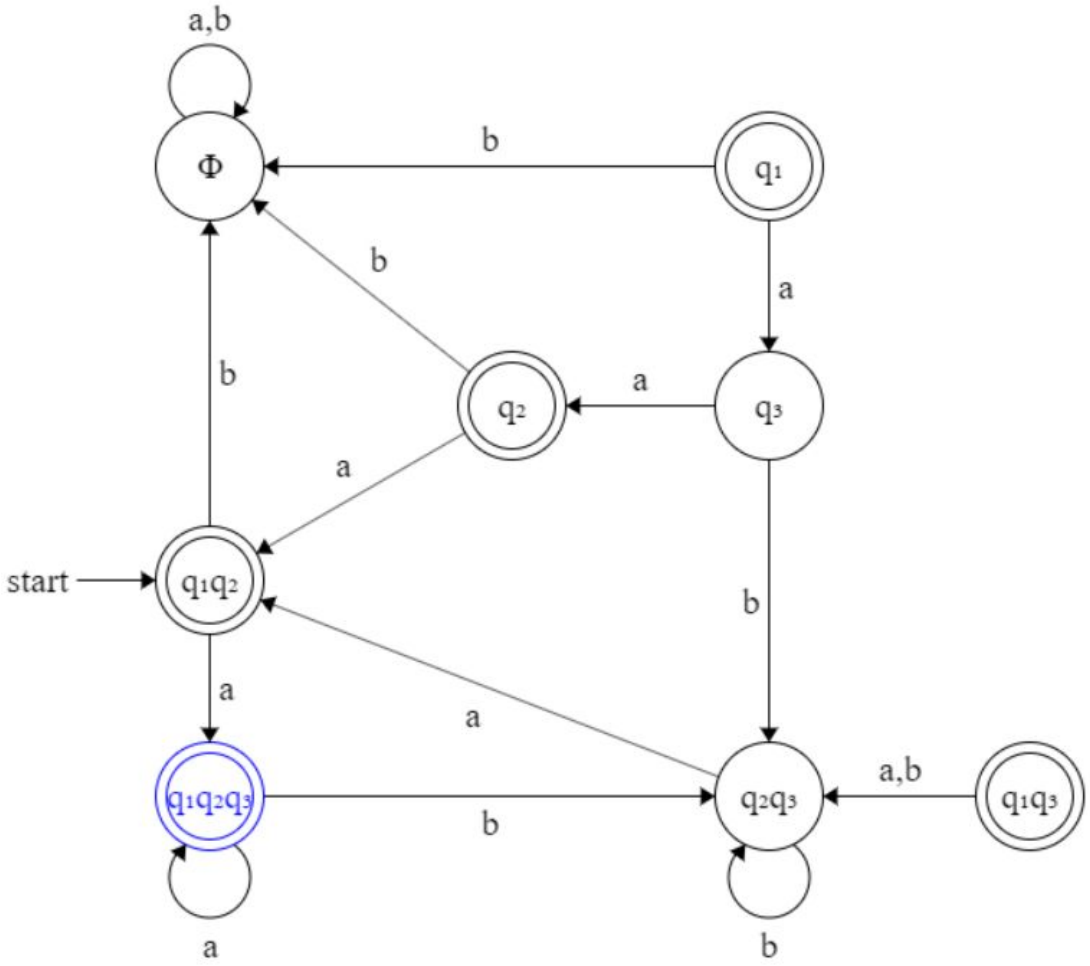
- q' is an accept state in the original NFA (i.e. $\{q_2\}$)
- q' has an ε transition to an accept state in the original NFA (i.e. $\{q_1\}$)

So the new set of accept states are: $\{\{q_1\}, \{q_2\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\})$.

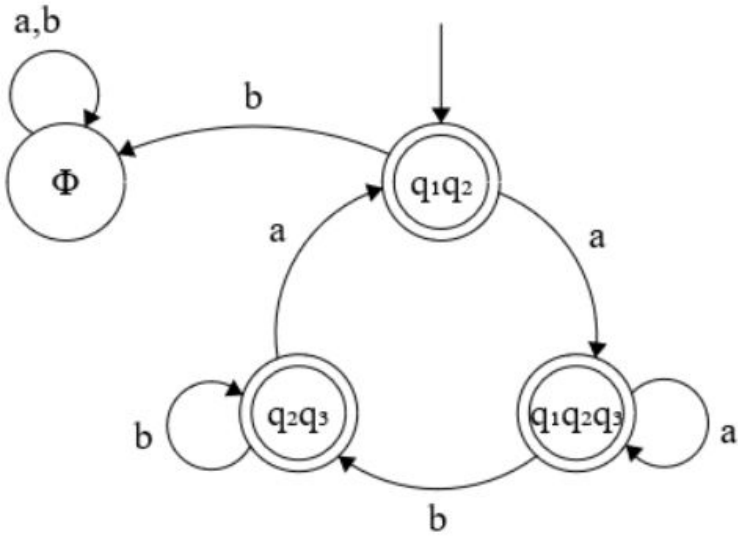
The new transitions are based on the information in the table above. For a current state q, we first create the set of states that can be reached by reading a single character in Σ (let's call this set q_Σ). Then we need to consider all transitions on ε characters for each state in q_Σ. Note that this method follows the book's pattern of only handling transitions on ε characters after reading a character from Σ.

As an example, let us consider reading an a from state $\{q_1, q_2\}$. From the table above, reading an a from q_1 or q_2 gives us $q_\Sigma = \{q_3\} \cup \{q_1\} = \{q_1, q_3\}$. We now need to consider all ε transitions from all states in q_Σ. From the table above, reading zero or more ε characters from q_1 will lead us to state $\{q_1, q_2\}$. We then need to recurse on each of these transitions. q_1 we have already handled, and reading zero or more ε characters from q_2 can only lead us to q_2, so there is nothing left to handle in this set of states. If we instead read zero or more ε characters from q_3 we can only end up at $\{q_3\}$. Finally, we need to take the union of all possible states that we have found. Thus, $\delta'(\{q_1, q_2\}, a) = \{q_1, q_2, q_3\}$.

Based on the above information, the new 8-state DFA is:

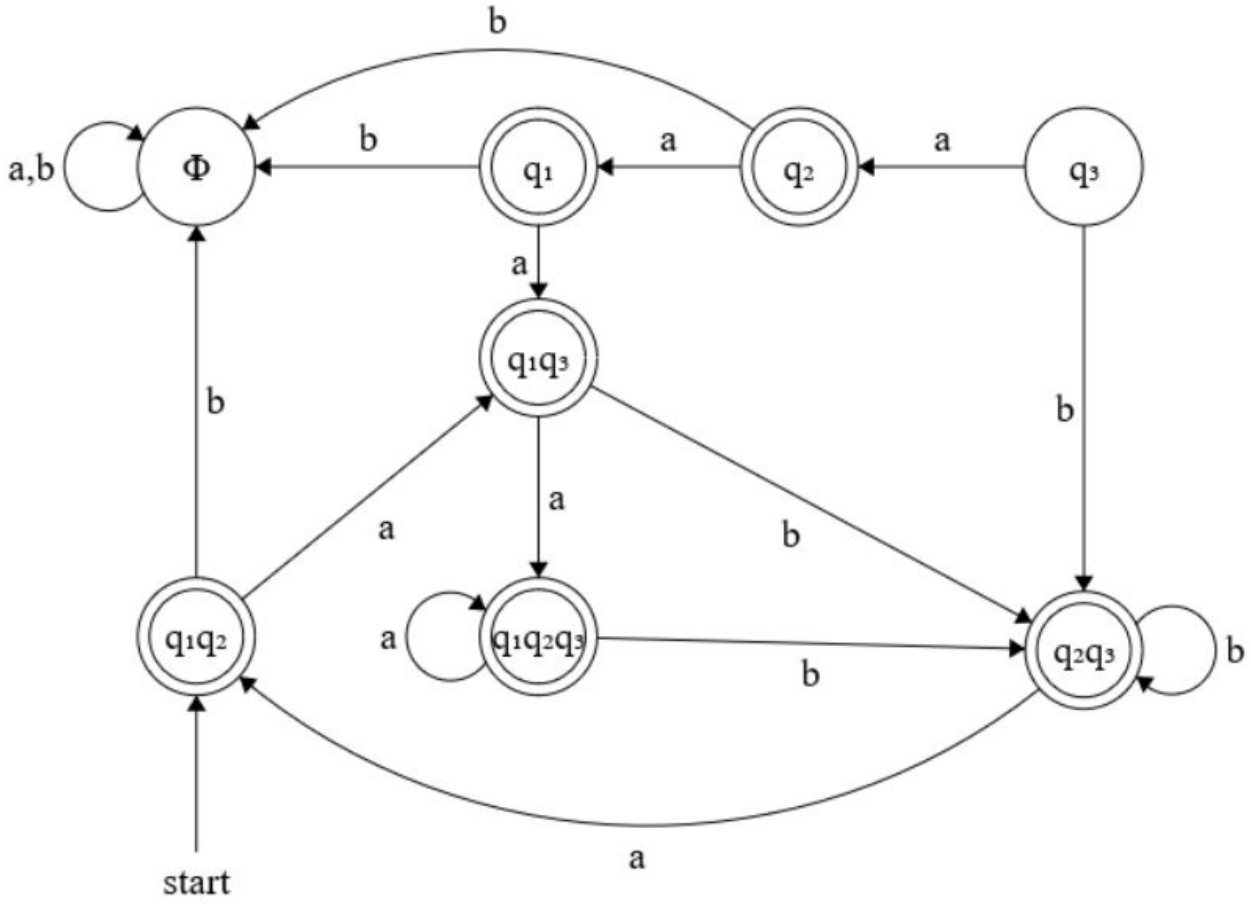


Now, note that several states in the 8-state DFA are unreachable. For example, both $\{q_1 q_3\}$ and $\{q_1\}$ have no incoming states, and so they can be removed without affecting the functionality of the DFA. By similar logic, once you have removed the above two states you can also remove $\{q_2\}$ and $\{q_3\}$. Thus, the final 4-state DFA is



Common mistakes:

1. Excluding ε from their English description of the language
2. Excluding the fact that the substring "bab" cannot appear in an accepted string
3. Not marking the start state in the diagram of the DFA
4. Not marking the accept states in the diagram of the DFA
5. Many people did not consider the idea that if you read the entire string and end at state q_1 that you can take the ε transition to $\{q_2\}$, an accept state. Thus, $\{q_1\}$ is implicitly an accept state and $\{q_1\}$ and $\{q_1, q_3\}$ should be accept states as well in the DFA.
6. Not considering ε characters when coming up with the transition function for the DFA (the book follows a methodology where it only reads ε characters are reading a single character from the input string, and this solution follows that method as well. However, as the book mentions there is an equivalent solution where you you follow ε transitions before reading a character from the input string. I have included this DFA in the following image. In this case attempting to reduce the DFA results in a 5-state DFA).


b
a,b
Φ
b
q1
a
q2
a
q3
a
q1q3
b
b
a
a
b
q1q2
a
q1q2q3
b
q2q3
b
a
start