INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

THEORY OF COMPUTATION

CS340

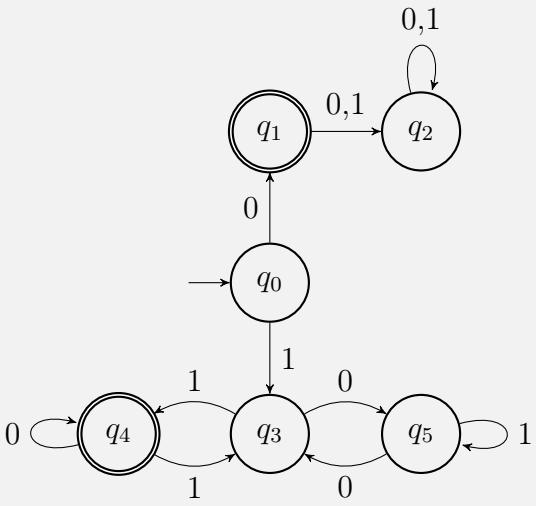
Assignment 1

Authors:
Kajal Deep (200483)
Uttam Kumar (201071)
Nishant Roshan (200643)

September 15, 2022

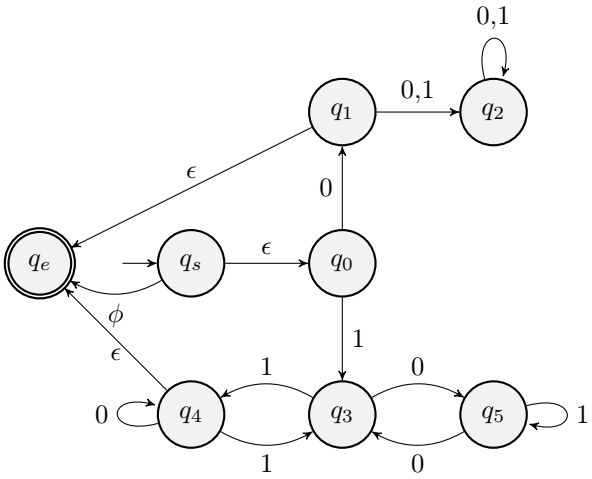
Question 1

Describe the language accepted by the following DFA :

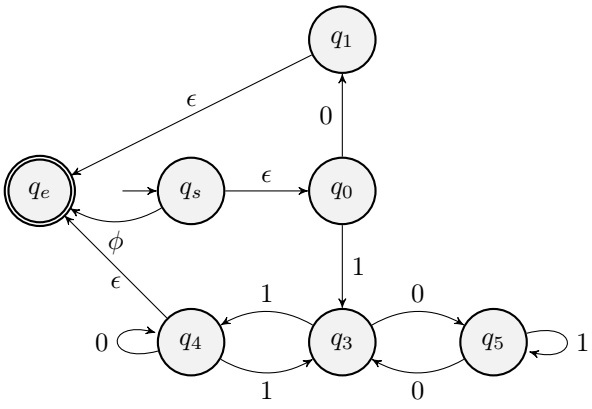


Solution:

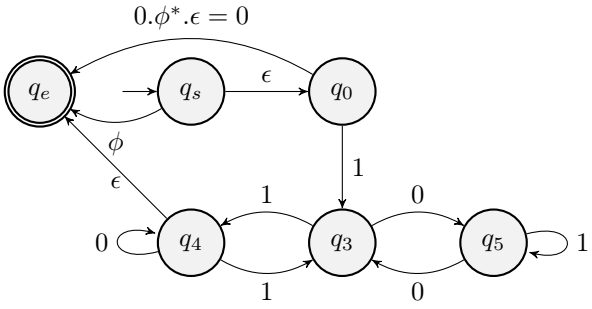
Converting DFA to RE, we first introduce two new states, i.e a start state and a final accepting state. Now introduce an ϵ transition from the new start state to old start state and also from the old accepting states to the new accepting state. Other than these and those mentioned, any other transition from any state to any other state is assumed to be ϕ transition.



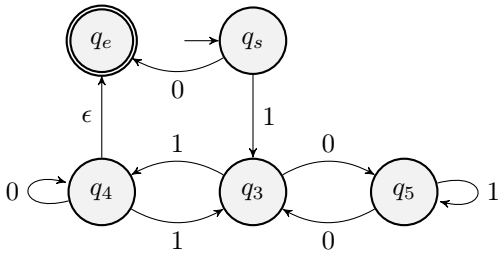
Now we start removing the states one by one.
Removing q_2,



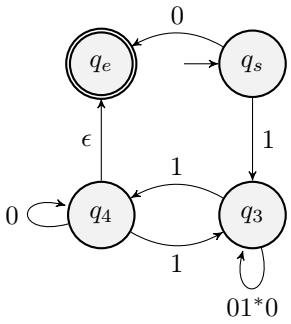
Removing q_1,



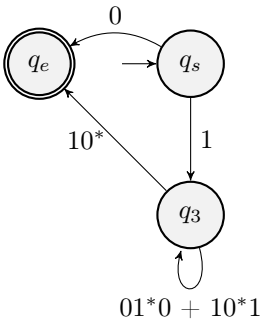
Removing q_0,



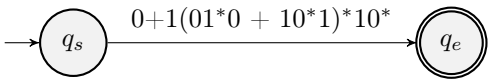
Removing q_5,



Removing q_4,



Removing q_3,



So finally we are left with the GNFA ($0 + 1(01^*0 + 10^*1)^*10^*$) ,which corresponds to the transition between the two states.

Therefore the language accepted by the DFA is ($0 + 1(01^*0 + 10^*1)^*10^*$).

Question 2

Let w be a string over some alphabet Σ. Odd(w) refers to the string obtained by deleting symbols at all even positions of w. Example, if w = $a_1a_2a_3...a_n$, then odd(w) = $a_1a_3a_5...a_m$, where m is n or n − 1 when m is odd or even respectively. For a language L $\subseteq\Sigma^*$, define oddL = {odd(w) | w ∈ L}. Prove that if L is regular, then oddL is regular too.

Solution:
To show that oddL is regular we will mimic the DFA (say A) that accepts language L to accept oddL. We will introduce new states corresponding to each state in the old DFA (A). Our new NFA will be such that it will alternate between old and new states. Here we will have an option to do ϵ transition at even steps.

Let A = (Q,Σ,δ,q_0,F) be the DFA which accepts language L.
More formally, we define a NFA accepting oddL as follows:
A' = (Q',Σ,δ',q_0',F').

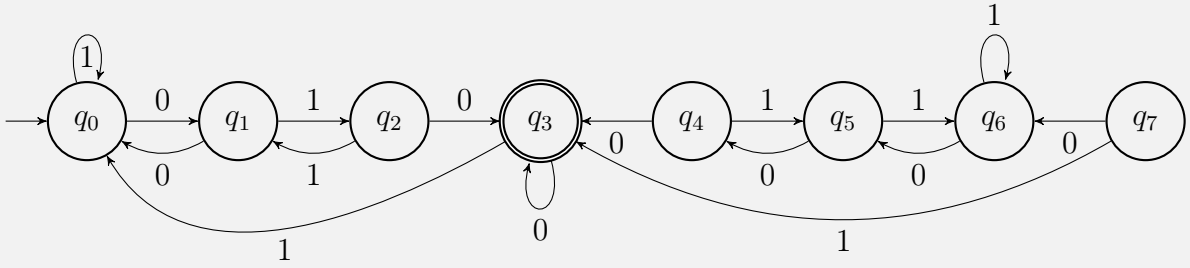
- Here Q' will contain a new states corresponding to each old state in addition to the old states. That is, if the original DFA had the set of states $\{q_0, q_1, q_2, ..., q_k\}$, the we add k new states $\{r_0, r_1, r_2, .., r_k\}$, each corresponding to a state in the original DFA.
  Q'=$\{q_0, q_1, q_2, ..., q_k, r_0, r_1, r_2, .., r_k\}$
- For F', r_i will be a final state if and only if q_i was a final state. Therefore the total set of final states in the new NFA is a union of initial final states in the DFA and the newly created final states.
- The starting state of both the automatas will be same. Therefore $q_0' = q_0$
- We define the δ' as follows:
  We modify each transition $q_i \to q_j$ on reading alphabet a in A by two transitions $q_i \to r_j$ on reading the alphabet a and an ϵ transition $r_i \to q_j$.

So after taking one symbol from oddL and doing the transition from q_i to r_j, it does ϵ transition from r_j to some q_k. This is equivalent to q_i to q_j to q_k in the original DFA which required 2 symbols to do this, but the NFA requires only 1.

Thus, this NFA A' will accept oddL. Hence oddL is regular as there is an NFA accepting it.

Question 3

Find the minimum-state finite automaton corresponding to the following DFA. Show in details all the steps of minimization.



Solution:

Property: Two states will be equivalent if they take to the same equivalent set for every transition.

The states q_4, q_5, q_6, q_7 can never be reached from the initial start state q_0. So these have no contribution in deciding which strings will be accepted by the DFA. Hence these could be easily eliminated in the minimum DFA.

Transition Table:

Initial State	input = 0	input = 1
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	q_0

Consider the 0 equivalence, separated by Final and non-Final states

$\{q_0, q_1, q_2\}$

$\{q_3\}$

Consider the first equivalence

$\{q_0, q_1\}$

$\{q_2\}$

$\{q_3\}$

Consider the second equivalence

$\{q_0\}$

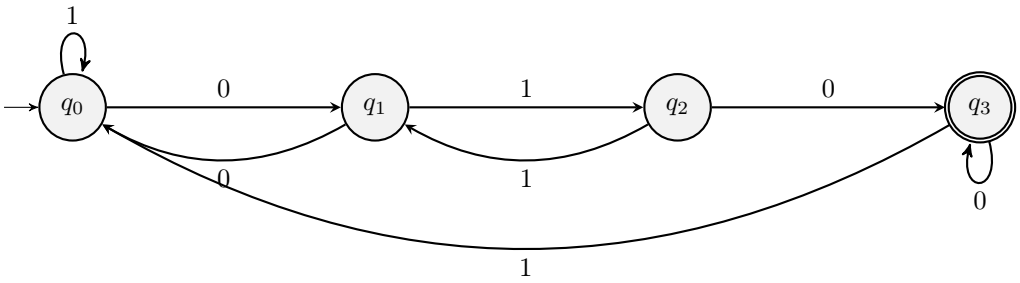
$\{q_1\}$

$\{q_2\}$

$\{q_3\}$

At this time, we realise that none of the two states can belong to the same equivalent set.

So there will not be any further minimization.



Question 4

For languages L_1 and L_2 over Σ, define,

$$Mix(L_1, L_2) = \{\omega \mid \omega = x_1y_1x_2y_2...x_ky_k,\ where\ x_1x_2..x_k \in L_1 and\ y_1y_2..y_k \in L_2,\ each\ x_i, y_i \in \Sigma^*\}$$

Show that if L_1 and L_2 are regular then Mix(L_1, L_2) is also regular.

Solution:
Let the DFA accepting language
L_1 be $A_1 = (Q_1, \sum, \delta_1, q_1, F_1)$

and the DFA accepting the language
L_2 be $A_2 = (Q_2, \sum, \delta_2, q_2, F_2)$.

Let a string $x_1x_2x_3...x_k \in L_1$. Also, let another string $y_1y_2y_3...y_k \in L_2$ (where each x_i,$y_i \in \Sigma^*$). Since these strings are formed by two regular languages, they must take the corresponding DFA A_1 and A_2 to their final states.

Let a NFA $A = (Q, \sum, \delta, q_0, F)$.
We will define this NFA in such a way that it accepts the language L = Mix(L_1, L_2) .
At any time, NFA needs to keep track of the current states of A_1 and A_2. At any stage, if the input string is in language L_1 or L_2, the NFA makes transition in A_1 or A_2 respectively. Finally, after the input ends, the NFA accepts if both A_1 and A_2 are in their final states. In addition, the NFA A should also be accepting empty strings. Basically, A now switches from running A_1 and running A_2 after each input string is read.

More formally we define NFA A as follows:

- Q = $(Q_1XQ_2) \cup \{q_0\}$. Where Q_1XQ_2 keeps track of states of A_1 and A_2, while q_0 is the initial state when nothing is read.
- q = q_0 is the initial state when nothing is read.
- F = $(F_1XF_2) \cup \{q_0\}$, which states that A accepts the string if both A_1 and A_2 are in accept states, or A accepts the empty string.
- Define transition function δ as:
    1. $\delta(q_0,\epsilon) = (q_1, q_2)$, which states that at the start state, A can transition in both A_1 and A_2 without reading anything.
    2. $\delta((q_{i1}, q_{j2}), a) = \{(\delta_1(q_{i1}, a), q_{j2}), (q_{i1}, \delta_2(q_{j2}, a))\}$ which means that if the current state of A_1 is q_{i1} while the current state of A_2 is q_{j2}, the on reading the input string a, there are two possibilities, either A can make transition in A_1 and let A_2 remain unchanged or vice versa.

Our newly defined NFA A will accept Mix(L_1, L_2) if we end up in one of the final states in set F starting from the initial state q_0.
Hence there exists a Finite automata that accepts Mix$(L_1, L2)$.
Therefore Mix(L_1,L_2) is regular.

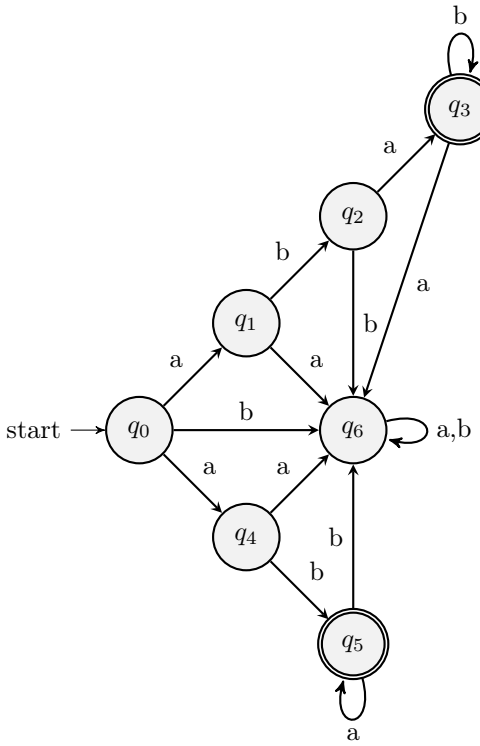
Question 5

Design an NFA to accept the following language,

$$L = \{w \mid w \text{ is } abab^n \text{ or } aba^n \text{ where } n \geq 0\}.$$

Solution:
The NFA that accepts the given language is as follows:

- Define a state q_6 as the sink state.
- If the input is of the form $abab^n$, the NFA will make transitions in the upper branch.
- If the input is of the form aba^n, the NFA will make transitions in the lower branch.
- For any other input form, the NFA ends up in the sink state.



NOTE: *Since here we have designed an NFA, it is possible for an NFA to make multiple transitions on a single input, i.e. it allows multiple states at the same time. But this is not possible in a DFA.*

Question 6

Let L be the language containing all strings over the alphabet 0, 1 that satisfy the property that in every string, the difference between the number of 0's and 1's is less than two (e.g., 00101 will be in the language, while 010001 will not). Is this a regular language? Explain your answer.

Solution:
Pumping Lemma:
For any regular language L, there exists an integer n>0, such that for all x ∈ L with |x| ≥ n, there exists u, v, w ∈ $\sum^*$, such that x = uvw, and
1. |uv| ≤ n
2. |v| ≥ 1
3. for all i ≥ 0: uv^iw ∈ L

Let us assume that L, the language containing all strings over the alphabet 0,1 that satisfy the property that in every string difference between the number of 0's and 1's is less than two is regular.

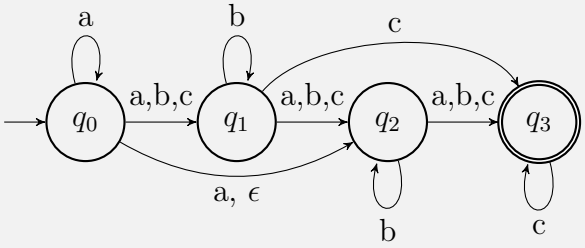
Let n be the pumping length.
We will apply the pumping lemma as a game played between the adversary and us.

- Let the adversary choose some n > 0.
- Now, we can choose any string (length ≥ n) belonging to L.
  Choose 0^n1^n ∈ L.
  0^n1^n ∈ L because the difference between the number of 0's and 1's is 0 (less than two).
- Adversary splits
  0^n1^n = uvw, with constraints on the split such that $|uv| \leq$ n and |u| ≥ n
- Now, no matter what the split is, we will always have uv = 0^{j+k}, v = 0^k
- We can choose i = 3. This would imply:
  $uv^3w = 0^j0^{3k}w$
- Difference between number of 0s and 1s in uv^iw is 2k ≥ 2 as k ≥ 1. Hence $uv^iw \notin$ A.

This contradicts our assumption that L is regular.
Hence L is not a regular Language.

Question 7

Convert below NFA to its equivalent DFA.



Solution:

Transition Table for the NFA:

Initial State	a	b	c
$\{q_0\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_1, q_3\}$
$\{q_1\}$	$\{q_2\}$	$\{q_1, q_2\}$	$\{q_2, q_3\}$
$\{q_2\}$	$\{q_3\}$	$\{q_2, q_3\}$	$\{q_3\}$
$\{q_3\}$	ϕ	ϕ	$\{q_3\}$

Equivalent transition Table for the DFA:

Initial State	a	b	c
$\{q_0\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_1, q_3\}$
$\{q_1, q_3\}$	$\{q_2\}$	$\{q_1, q_2\}$	$\{q_2, q_3\}$
$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$
$\{q_1, q_2\}$	$\{q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$
$\{q_2\}$	$\{q_3\}$	$\{q_2, q_3\}$	$\{q_3\}$
$\{q_2, q_3\}$	$\{q_3\}$	$\{q_2, q_3\}$	$\{q_3\}$
$\{q_3\}$	ϕ	ϕ	$\{q_3\}$

All the states containing q_3 will be the accepting state.
Hence, the corresponding state transition diagram will be

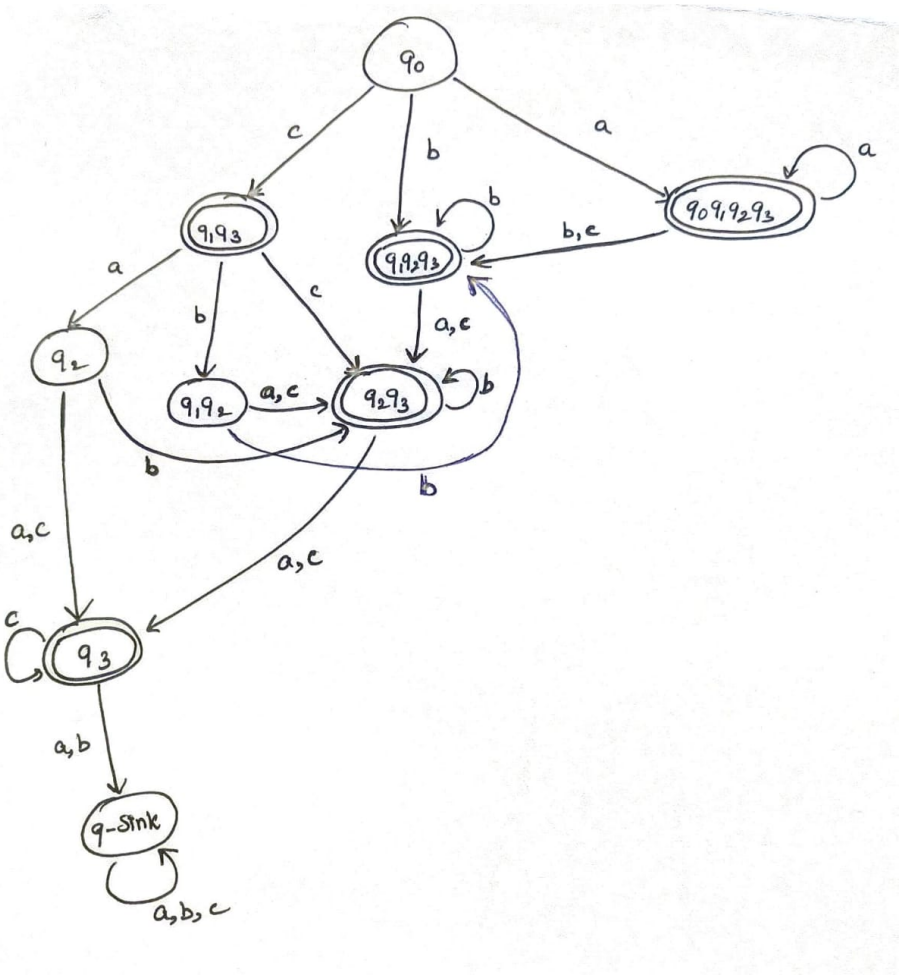

q_0
c
b
a
a
q_1q_3
b
$q_0q_1q_2q_3$
$q_1q_2q_3$
b, c
a
b
c
a, c
q_2
q_1q_2
a, c
q_2q_3
b
b
b
a, c
a, c
c
q_3
a, b
q-Sink
a, b, c

Figure 1

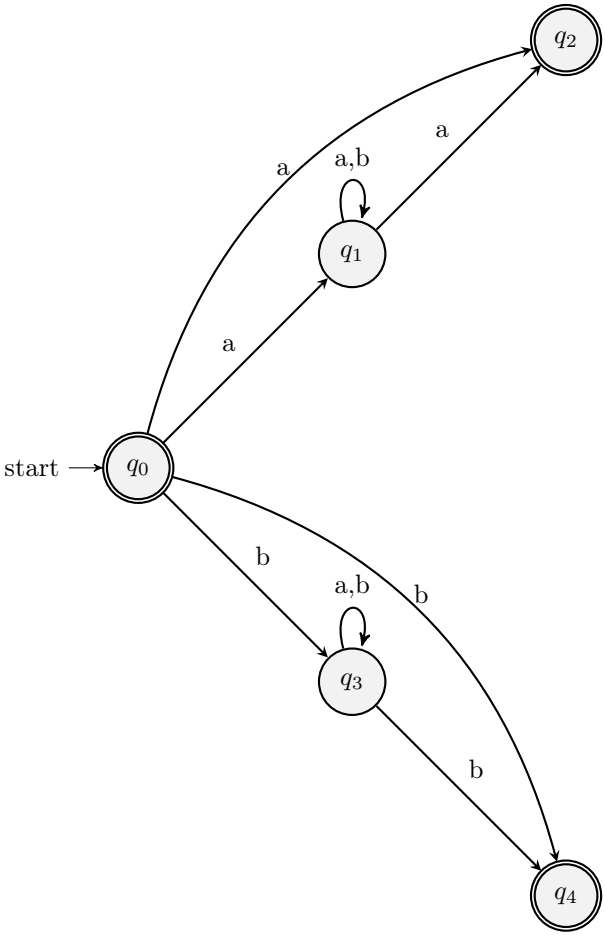
Question 8

Let L be the language $L = \{\text{w} \in \{a, b\}^* \mid \text{w contains an equal number of occurrences of ab and ba}\}$
(a) Give a regular expression whose language is L.
(b) Design a DFA/NFA/ϵ-NFA to accept L.

Solution:
(a) The language will contain all the strings beginning and ending with the same alphabet and also the empty string ϵ.

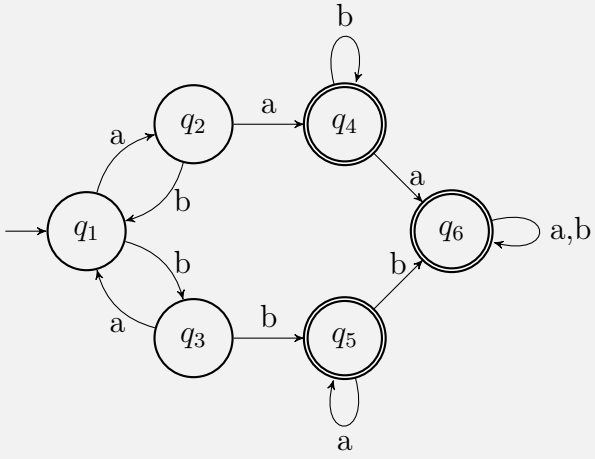
$$L = a.(a+b)^*.a + b.(a+b)^*.b + a + b + \epsilon$$

(b) We will design an NFA which accepts L :



Question 9

Use the DFA state-minimization procedure to convert this DFA to an equivalent DFA with the minimum possible number of states.

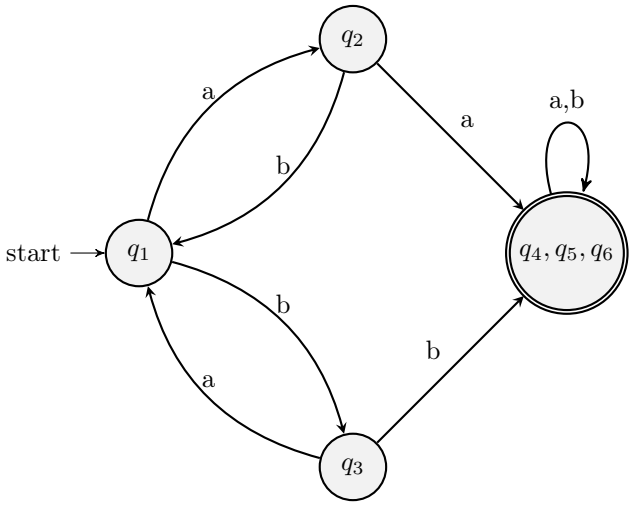


Solution:

Transition Table:

Initial State	input = a	input = b
q_1	q_2	q_3
q_2	q_4	q_1
q_3	q_1	q_5
q_4	q_6	q_4
q_5	q_5	q_6
q_6	q_6	q_6

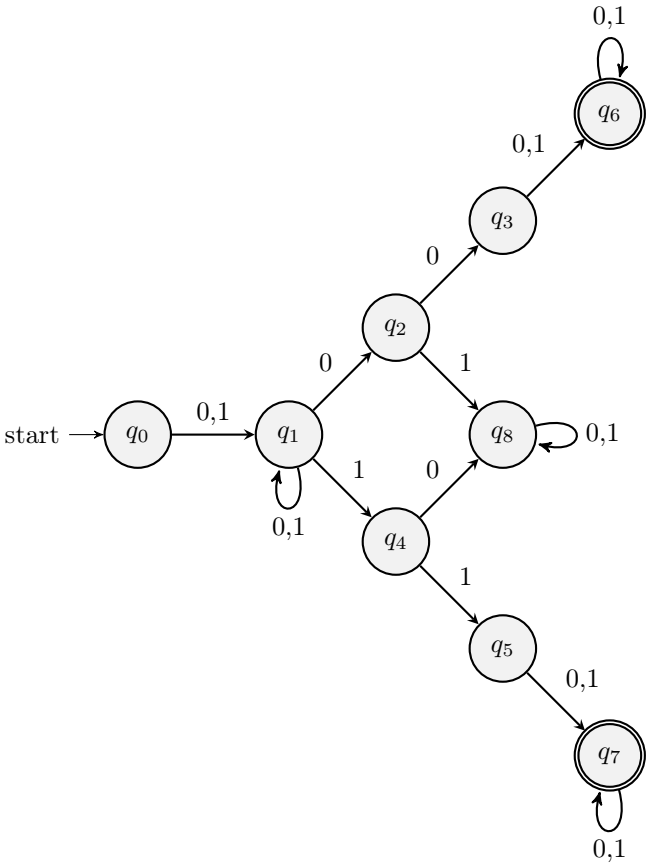
Consider the 0 equivalence, separated by Final and non-Final states
$\{q_1, q_2, q_3\}$
$\{q_4, q_5, q_6\}$
Consider the first equivalence,
(q_1,q_2) : when input = a, $q_1 \to q_2$ and $q_2 \to q_4$, as q_2 and q_4 are non-equivalent states, q_1 and q_2 are also non-equivalent.
(q_1,q_3) : when input = a, $q_1 \to q_2$ and $q_3 \to q_1$, as q_2 and q_1 are non-equivalent states, q_1 and q_3 are also non-equivalent.
(q_2,q_3) : when input = a, $q_2 \to q_4$ and $q_3 \to q_1$, as q_4 and q_1 are non-equivalent states, q_2 and q_3 are also non-equivalent.
Doing similar computations for (q_4, q_5), (q_5, q_6), and (q_4, q_6), shows that they are equivalent states.
$\{q_1\}$
$\{q_2\}$
$\{q_3\}$
$\{q_4, q_5, q_6\}$
Consider the second equivalence
$\{q_1\}$
$\{q_2\}$
$\{q_3\}$
$\{q_4, q_5, q_6\}$
Thus, q_4, q_5, q_6 are equivalent states.



Question 10

The language L = uvv^rw | u, v, w $\in \{a, b\}^+$ is regular. Here, v^r is the reverse of v. Design a regular expression whose language is L. Convert the regular expression to an equivalent NFA.

Solution:
vv^r is in the middle of the string means that all the accepted string will contain a palindrome of even length. If there is a palindrome of even length there must be two consecutive 0s or two consecutive 1s in the string.
L = $\{0,1\}^+00\{0,1\}^+ + \{0,1\}^+11\{0,1\}^+$


start
q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
0,1
0
1

Question 11

Find out the equivalent regular expressions from the list given and show why they are equivalent :
1. $(a + ba)^*(b + \epsilon)$
2. $(a^*(ba)^*)^*(b + \epsilon) + a^*(b + \epsilon) + (ba)^*(b + \epsilon)$
3. $(a + ba)(a + ba)^*(b + \epsilon)$

Solution:
Let exp = expression 1 = $(a + ba)^*(b + \epsilon)$
In expression 2, taking $(b + \epsilon)$ common, we get:

$$((a^*(ba)^*)^* + a^* + (ba)^*)(b + \epsilon) \qquad \rightarrow \text{Equation 1}$$

Next, apply:

$$(A^*B^*)^* = (A + B)^*$$

(This can simply be proved logically that both the sides are all possible combinations of A and B)

Applying this to Equation 1, we get:

$$((a + ba)^* + a^* + (ba)^*)(b + \epsilon) \qquad \rightarrow \text{Equation 2}$$

Now observe: $a^* \subset (a + ba)^*$ and that $(ba)^* \subset (a + ba)^*$
Therefore

$$a^* + (a + ba)^* = (a + ba)^* \text{ and } (ba)^* + (a + ba)^* = (a + ba)^*$$

Apply this in equation 2, we get:
$(a + ba)^*(b + \epsilon)$ = Expression 2 = Expression 1 = exp
Therefore expressions 1 and 2 are equivalent.
Analyse expressions 1 and 3: We see that $(b + \epsilon) \subset$ expression 1
While $(b + \epsilon) \not\subset$ expression 3

Therefore, expressions 1 and 2 are equivalent to each other but not equivalent to expression 3.

Question 12

Design DFA for the following languages over 0, 1 :
(a) The set of all strings such that every block of five consecutive symbols have at least two 0s.
(b) The set of strings with an equal number of 0s and 1s such that each prefix has at most one more 0 than 1s and at most one more 1 than 0s.

Solution:
a) The expected automata should accept strings which have at least two 0's in every block of five consecutive alphabets, for example, 100101, 0100111 etc.

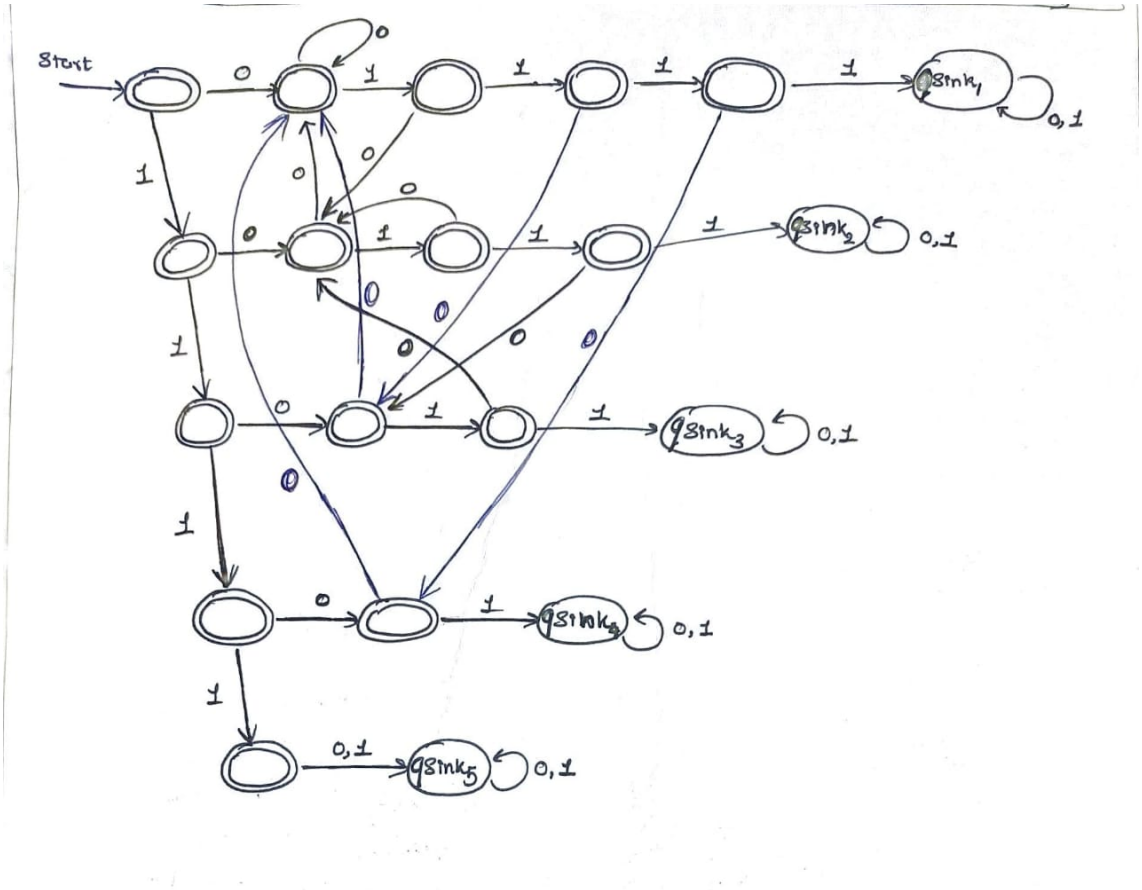
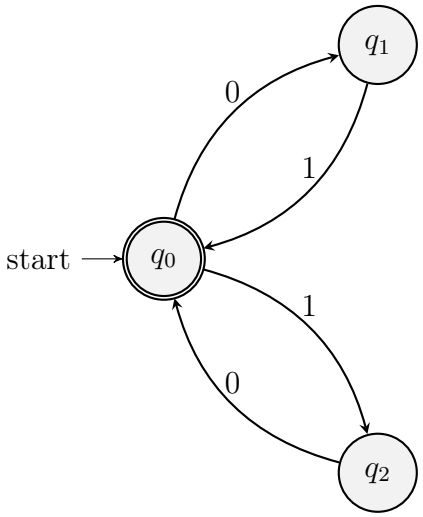


Figure 2

(b) The set of strings with an equal number of 0s and 1s such that each prefix has at most one more 0 than 1s and at most one more 1 than 0s.
The set of strings accepted by the DFA is given by $L = (01 + 10)^*$
Hence the corresponding transition diagram for the DFA will look like-


q_1
0
1
start
q_0
1
0
q_2

Question 13

Given that language L is regular, is the language $L_{1/2}$ = { x | ∃y such that |x| = |y|, xy∈L } regular? If yes, give a formal proof.

Solution:
Since L is regular, there must be a DFA which accepts it. Let D = (Q_L,Σ,δ_L,q_L,F_L) be the DFA accepting L.

How do we make sure that a string x ∈ half(L) ?
We have to check that if there exists a string y of length same as x, which when concatenated with x is accepted by D. In other words, if D goes to state q_i on seeing input x, i.e $\delta_L(q_L,x) = q_i$, we must check if there exists a string y, |y|=|x|, such that y takes the DFA from q_i to a final state, i.e, $\delta_L(q_i,y) \in F_L$. An easier way to say this is that if S_n be the set of all states of D which lead to a final state on seeing some input of length n, then if $q_i \in S_n$ then x ∈ half(L) and vise versa.

$$x \in \text{half(L)} \iff q_i \in S_n \qquad \longrightarrow \text{Equation 1}$$

Now let N = (Q,Σ,δ,q,F) be a DFA. We will define N so that it accepts half(L).

- Q = ($Q_L \times 2^{Q_L}$). Here every state of Q is a pair consisting of a single state of Q_L (to keep track of the current state of D) and a set of states of Q_L (the set of states that can reach an accept state of D in i steps, where i is the length of string processed so far).
- q = (q_L,S_0). Here q_L is the start state of D while S_0 denotes the set of all states which reach to a final state of D on reading an input of length 0. Simply this means S_0 is F_L. Therefore q = (q_L,F_L)
- F = {(q,S) | q∈Q_L, S∈2^{Q_L}, and q∈S}. This means that a string is accepted when the current state of D is at x, and x is itself is one of those states which is i steps from some final state of D. Here i is the length of input read so far.
- To define the transition function δ, let n be the length of input that has been read so far. Let N be at state (q,S_n), where we have defined S_n as above. Transition function on taking an input symbol (a) would be:

$$\delta((q,S_n),a) = (\delta_L(q,a),\ S_{n+1})$$

Now we have q = (q_L,S_0), i.e (q_L,F). We can prove by induction that $\delta(q,x) = (\delta_L(q_L,x),\ S_n)$, where |x| = n. Now by definition of the set of final states (F), x is accepted iff δ(q,x) is a final state iff $\delta_L(q_L,x) \in S_n$. From equation 1, $\delta_L(q_L,x) \in S_n$ iff x ∈ half(L). Thus N accepts the language half(L).

Therefore we have constructed a DFA that accepts half(L). This proves that half(L) is regular

Question 14

Define L' as the language consisting of the reverse of the strings in a language L. Give a formal proof that if L is regular, then L' is regular.

Solution:
Given that L is a regular language, there must exist a DFA which accepts it.
Let us denote the reverse of L as L'
Let the DFA that accepted language L be A. We will construct a new NFA A' and design it in such a way that it accepts L'.
Let A = (Σ,Q,s,F,δ) be the original DFA
Let there be an NFA A' = We will add a new start state p which will make ϵ transition to the all the original accepting states. The starting state of A will become the final state in A'.
It is clear that any string that is accepted in A , the reverse of it will be accepted in A'.As the reversed string will make corresponding transition in opposite direction.
Let the DFA accepting L be A = (Σ,Q,s,F,δ).
We construct an NFA A$'$ = (Σ,Q$'$,s$'$,F$'$,δ')
We define it formally as:

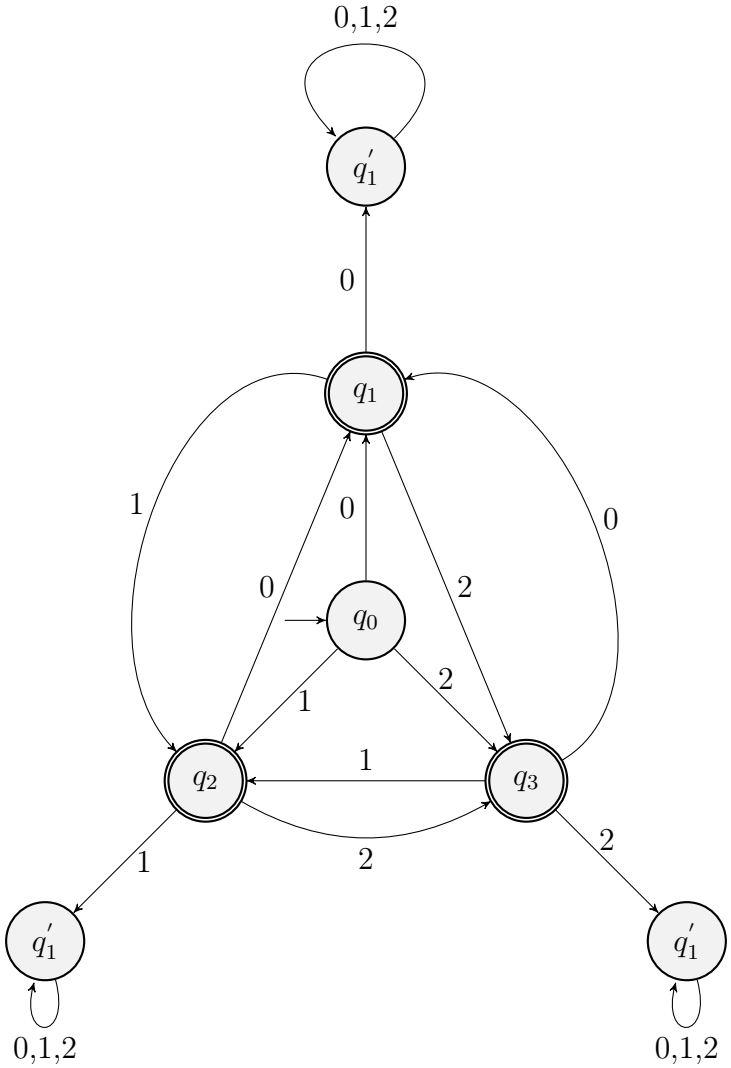
- Q$'$ = Q $\cup$ {p}, where we introduce a new state p and make ϵ transitions from p to each of the original accepting states of Q
- s$'$ = p, i.e. we define the newly introduced state as the new start state
- F$'$ = {s}, i.e. we make the start state of original DFA A as the single final accepting state of A'
- Define δ' as follows:
  1. δ'(p,ϵ) = F, i.e. there are ϵ transitions from new state p to all final accepting states of A
  2. δ'(q,a) = {q$'$ $\in$ Q | δ(q$'$,a) = q}, $\forall$ q $\in$ Q, a $\in$ Σ, i.e. we reverse all the transitions of A.

We can see that any accepting path in A will correspond to another accepting path in A'. Thus this new NFA accepts L' only. Thus L' is regular.

Question 15

Let L be the language over the alphabet {0, 1, 2} such that for any string in s ∈ L, s does not have two consecutive identical symbols, i.e. strings of L are any string in $\{0, 1, 2\}^*$ such that there is no occurrence of 00, no occurrence of 11, and no occurrence of 22. Design a DFA that accepts L.

Solution:
Given: The DFA should accept the language formed by the alphabets from the symbol {0,1,2} such that there is no consecutive occurrence of any two alphabets. Thus the corresponding DFA would be:



Question 16

Let Σ and $\triangle$ be two alphabets and let h: $\Sigma^* \to \triangle^*$. Extend h to be a function from Σ^* to $\triangle^*$ as follows:

$$h(\epsilon) = \epsilon$$
$$h(wa) = h(w)h(a) \qquad \text{where } w \in \Sigma^*,\ a \in \Sigma$$

(Such a function h is called a *homomorphism*)
Now, for $L \subseteq \Sigma^*$,

$$h(L) = \{h(w) \in \triangle^* \mid w \in L\}$$

Also, for $L \subseteq \triangle^*$,

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$

1. Prove that if $L \subseteq \Sigma^*$ is regular, then so is h(L)
2. Prove that is $L \subseteq \triangle^*$ is regular, then so is $h^{-1}(L)$

Solution:
Part 1: We need to prove that regular languages are closed under homomorphism, i.e., if L is a regular language and h is a homomorphism, then h(L) is also regular. The steps in proving this are:

1. Define homomorphism as an operation on regular expressions.
   For any regular expression R, let h(R) be the regular expression obtained by replacing each occurence of a Σ R in a, by the string h(a).
   Formally, h(R) is defined inductively as follows:
   - $h(\phi) = \phi$
   - $h(\epsilon) = \epsilon$
   - $h(a) = a$
   - $h(R_1.R_2) = h(R_1).h(R_2)$
   - $h(R_1 \cup R_2) = h(R_1) \cup h(R_2)$
   - $h(R^*) = (h(R))^*$
2. Claim : L(h(R)) = h(L(R))
   Proof of claim: By induction on the number of operations in R
   - Base Cases: For R = ϵ or ϕ, h(R) = R and h(L(R)) = L(R). For R = a, L(R) = a and h(L(R)) = h(a) = L(h(a)) = L(h(R)). So claim holds
   - Induction Step: For $R = R_1 \cup R_2$, observe that $h(R) = h(R_1) \cup h(R_2)$ and $h(L(R)) = h(L(R_1) \cup L(R_2)) = h(L(R_1)) \cup h(L(R_2))$.
     By induction hypothesis, $h(L(R_i)) = L(h(R_i))$ and so $h(L(R)) = L(h(R_1) \cup h(R_2))$
     Other cases ($R = R_1R_2$ and $R = R_1^*$) similar

   Therefore we proved that L(h(R)) = h(L(R)).
3. Let R be any regular expression of language L such that L = L(R). Let R' be another regular expression such that R' = h(R). Then using the above proved expression, we have $h(L) = L(R')$

$h(L) = L(R') \implies$ homomorphism of any regular language L is another regular language with regular expression R'

Part 2: Here we need to show that if L $\subseteq \triangle^*$ is regular then $h^{-1}(L)$ will be regular.
We will do this by modifying the DFA A for regular expression L = L(A).
Let A = (Q,$\triangle$,δ,q_0,F).
We will construct a DFA which will accept $h^{-1}(L)$ say A' = (Q,Σ,δ',q_0,F).
Definition for δ'
Let q' is the state reached from q after reading the string h(a) in w.r.t transition function δ.
Then, δ' (q,a) = q'.
That is we make the same transition in δ' on reading symbol 'a' as we would make on reading string h(a) in δ.
Our new DFA A will accept the Language represented by h^{-1}(L).
This shows that is L $\subseteq \triangle^*$ is regular, then h^{-1}(L) will be regular as we constructed DFA A' that accepts h^{-1}(L).

Thank You