CS340A: Assignment 1

Submission By: Alphabets

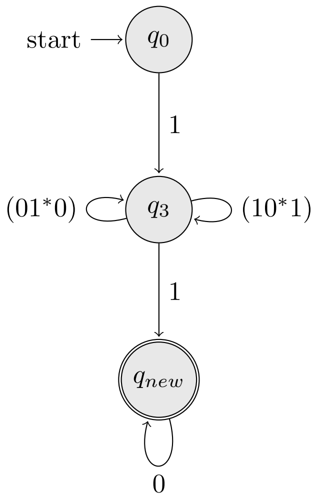
1. **The language accepted by the following DFA**

   Solution: Given, starting of DFA = q_0 and final states being q_1, q_4
   Now, the transition can be represented as,

   (a)
   $$q_0 \underset{0}{\to} q_1$$
   after which any input either 0 or 1 will lead it to dead state of q_2 and remain there for all time

   (b)
   $$q_0 \underset{1}{\to} q_3 \longrightarrow q_4$$
   upon reaching state q_3 it has two optional loops to go,
   - through q_5 upon reading 0
   - through q_4 upon reading 1

   We will remove q_5 and q_4 and add a new accepting state q_{new} which accepts same string. So, the lower half of DFA can be converted to equivalent GNFA.



   Thus, the language accepted by the DFA is-
   $$0 + 1(((01^*0)^*) + ((10^*1)^*))^*1(0^*)$$

□

2. **Prove that if L is regular, then $oddL$ is regular too**

   Solution:

   According to question, defined entities are –
   w be a string over some alphabet $\sum$ represented as -
   $$w = a_1a_2a_3...a_n$$

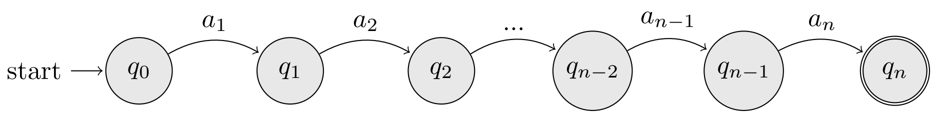
then $odd(w)$ is stated as -

$$odd(w) = a_1a_3a_5...a_m$$

where $m = n$ or $n-1$ depending upon n being odd or even Now, over language $L \subseteq \Sigma^*$ $oddL$ is defined as-

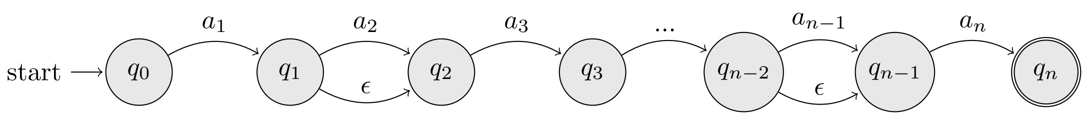
$$oddL = \{odd(w)|w \in L\}$$

If L is regular, then there must exist a DFA which can be made as -

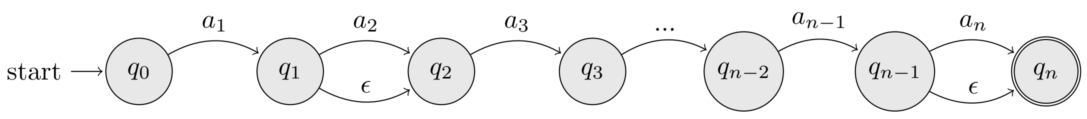


for, $oddL$ to be regular a NFA from this DFA by making ϵ transition between the odd states -

if n is odd-



if n is even-



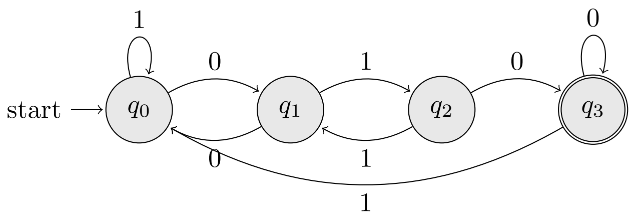
which shows $oddL$ is regular too.

□

3. **Minimum-state finite automaton corresponding to the given DFA**

Solution:

Since, the right half states can never be reached from the starting state, we dont need to bother about it and thus the reachable states can be represented as-



Drawing transition table -

state/input	0	1
q_0	q₁	q₀
q_1	q₀	q₂
q_2	q₁	q₃
q_3	q₃	q₀

We will start by making partitions and in each step we will breakdown the partitions into the smaller partitions which will eventually land up into equivalence classes.

Property:
Two states will be in same set of current partition if on reading x $\forall x \in \Sigma$ both land up to same set of previous partition. – [1]

Step 1: We will make partition on the basis of accepting and non-accepting states-

$$Q = \{\{q_0, q_1, q_2\}, \{q_3\}\}$$

Step 2: Now we will make smaller partition, using - property[1] after considering each pair-

(a) q_0, q_1: from transition table we can see that, upon reading 0 as well as 1 both land in same set.

(b) q_0, q_2: from transition table we can see that, upon reading 0 is same set where as for 1 both land in different set. Thus, q_0 and q_2 must be in different partition.

(c) q_1, q_2: from transition table we can see that, upon reading 0 is same set where as for 1 both land in different set. Thus, q_1 and q_2 must be in different partition.

Thus, new partition is -

$$Q = \{\{q_0, q_1\}, \{q_2\}, \{q_3\}\}$$

Step 3: Now we will make smaller partition, using - property[1] after considering each pair in non-singleton sets-

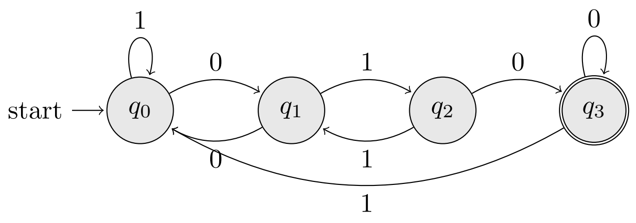
(a) q_0, q_1: from transition table we can see that, upon reading 0 is same set where as for 1 both land in different set. Thus, q_0 and q_1 must be in different partition.

Thus, final partition is -

$$Q = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_3\}\}$$

And this is also the equivalence class.

Thus, the DFA converges to minimum **4** states.



□

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

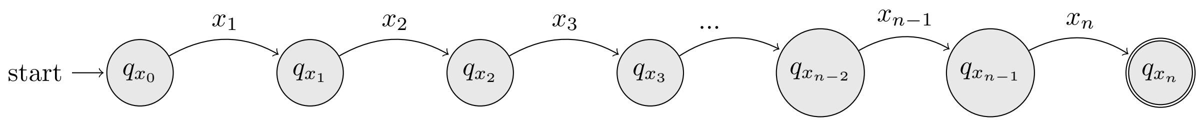
For languages L_1 and L_2 over $\sum$, we define,

$$Mix(L_1, L_2) = \{w \in \Sigma^* | w = x_1y_1x_2y_2x_3y_3...x_ny_n\}$$

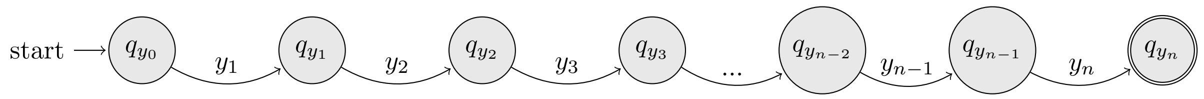
$$where \;\; x_1x_2x_3...x_n \in L_1 \;\; and \;\; y_1y_2y_3...y_n \in L_2, each \;\; x_i, y_i \in \Sigma^*$$

Solution:

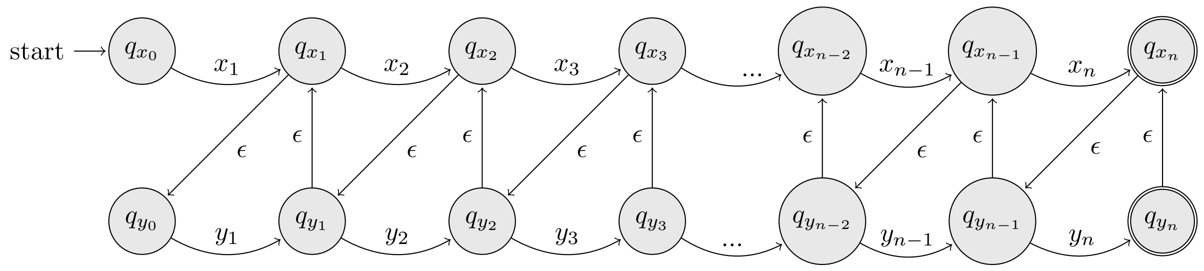
DFA for L_1-



DFA for L_2-



Merging both DFA into an NFA we get -



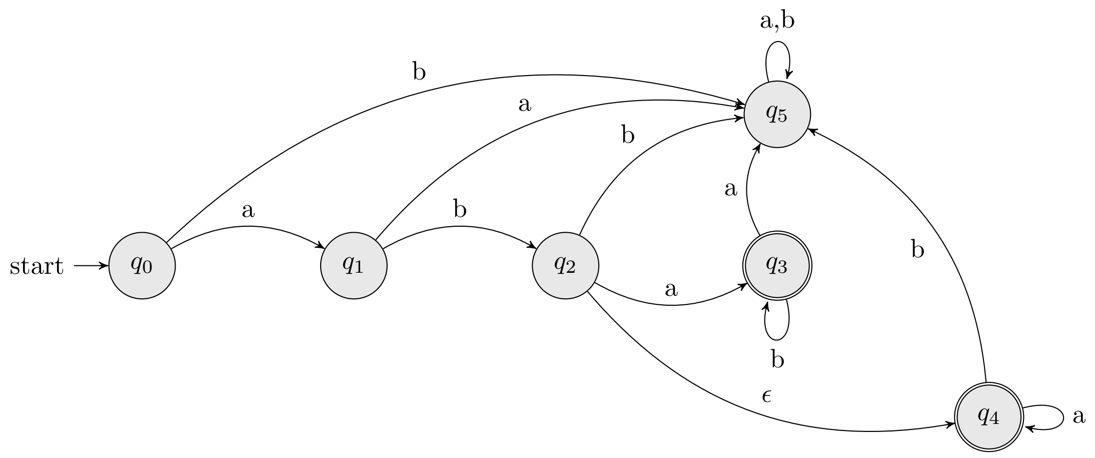
And we can clearly $Mix(L_1, L_2)$ is accepted by this NFA thus, $Mix(L_1, L_2)$ is also regular.

□

5. **Design an NFA for language L**

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

Solution:



Here, q_3&q_4 are accepting state for language L and q_5 is dead state. □

6. **Is L a regular language?**
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).

Solution:

Using contradiction by Pumping Lemma we follow the following steps-

Step 1: Lets assume L is regular.
Step 2: We adversely choose $n > 0$
Step 3: We take $x \in L$ such that $|x| = m, m > n$
where first n symbols are same, say 0 -

$$x = 0^n y$$

where y is the rest of the string

Lets, define $N(a)_t$ be the number of symbol a in string t.
Then, according to language conditions-

$$|(N(0)_y + n) - (N(1)_y)| < 2$$

$$|N(0)_x - N(1)_x| < 2$$

–[1]

Step 4: x can be split as $uvw, |uv| \leq n, |v| \geq 1$
for any $j \leq n, k \geq 1, j \geq k$, we have, $v = 0^k$ & $uv = 0^j$

Step 5: pumping v by choosing i we get string as - $x' = uv^iw$. Thus,

$$N(0)_{x'} = N(0)_u + N(0)_{v^i} + N(0)_w$$

$$N(0)_{x'} = (j - k) + ik + N(0)_w$$

$$N(1)_{x'} = N(1)_u + N(1)_{v^i} + N(1)_w$$

Using the fact that $N(1)_u = 0$ and $N(1)_v = 0$

$$N(1)_{x'} = N(1)_w$$

For regularity we must have-

$$|N(0)_{x'} - N(1)_{x'}| < 2$$

Using derived values of $N(0)_{x'}$ and $N(1)_{x'}$

$$|((j - k) + ik + N(0)_w) - N(1)_w| < 2$$

$$|(j + N(0)_w + (i - 1)k) - N(1)_w| < 2$$

–[2]

We also have $N(0)_x = j + N(0)_w$ and $N(1)_x = N(1)_w$, which gives

$$|N(0)_x + (i - 1)k - N(1)_x| < 2$$

–[3]

But from [1], we also have

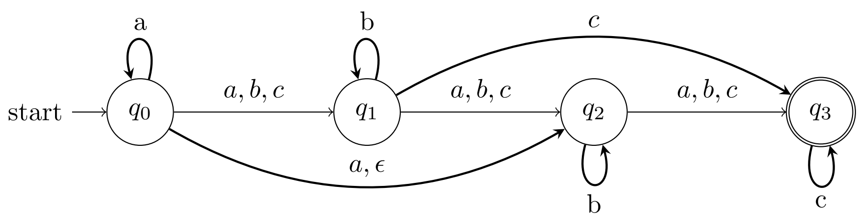
$$|N(0)_x - N(1)_x| < 2$$

we can always get the value of i such that [3] is not valid as $k > 0$ which contradicts $x' \in L$.

Therefore, L is irregular.

□

7. Convert below **NFA to its equivalent DFA**



Solution:

We will make a transition table to make our life easy

	a	b	c
q_0	q_0, q_1, q_2, q_3	q_1, q_3	q_1, q_3
q_1	q_2	q_1, q_2	q_2
q_2	q_3	q_2, q_3	q_3
q_3	-	-	q_3

We will update this transition table using some more states so that the resulting transition table can be

converted to DFA

Lets define $q_{i_1}q_{i_2}...q_{i_k}$ to be state which has all the transitions which $q_{i_1}, q_{i_2}, ...q_{i_k}$ has of their own.

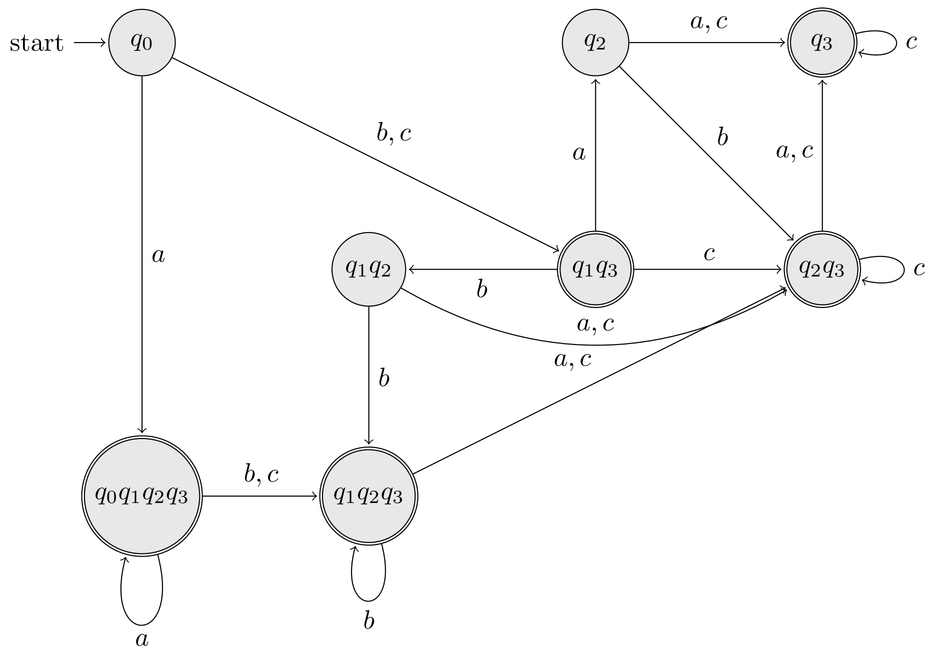
We will add some new states = $\{q_1q_2, q_1q_3, q_2q_3, q_1q_2q_3, q_0q_1q_2q_3\}$

Now the transition table will look like this:

	a	b	c
q_0	$q_0q_1q_2q_3$	q_1q_3	q_1q_3
q_1	q_2	q_1q_2	q_2
q_2	q_3	q_2q_3	q_3
q_3	-	-	q_3
q_1q_2	q_2q_3	$q_1q_2q_3$	q_2q_3
q_1q_3	q_2	q_1q_2	q_2q_3
q_2q_3	q_3	q_2q_3	q_3
$q_1q_2q_3$	q_2q_3	$q_1q_2q_3$	q_2q_3
$q_0q_1q_2q_3$	$q_0q_1q_2q_3$	$q_1q_2q_3$	$q_1q_2q_3$

Now, we can use these states to make DFA as all the transitions land up in a unique state

We can see from the transition table that state q1 is unreachable , so we can remove this state and draw our DFA,



□

8. Let L be the language $L = \{w \in \{a, b\}^* | w \text{contains an equal number of occurrences of } ab\&ba\}$

   (a) Give a **regular expression** whose language is L

   (b) Design a **DFA/NFA/ϵ-NFA** to accept L

   Solution:

$$\sum = \{a, b\}$$

   So, $\sum^*$can be divided into 4 types :

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

   Lets start analyzing Type 1,

   If there is any apperance of b in this string, it can be viewed as substring of only b's followed and preceded

   by 'a'and there can be many such substrings.

   This guarantees that there will be equal number of appearance of ab and ba possibly none.

   So Type 1 and 2 satisfies the condition asked in question.

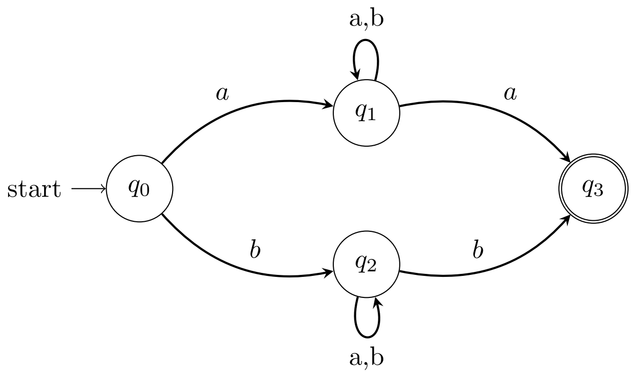
   Now lets analyze Type 3

If there is any apperance of b in this string, it can be viewed as substring of only b's followed and preceded

by 'a'except the last substring of b's which shows that there will be 1 extra ab than ba.

So, it violates our required condition and similarly Type 4 also violates it.

So, our required answer is $a(a+b)^*a + b(a+b)^*b$

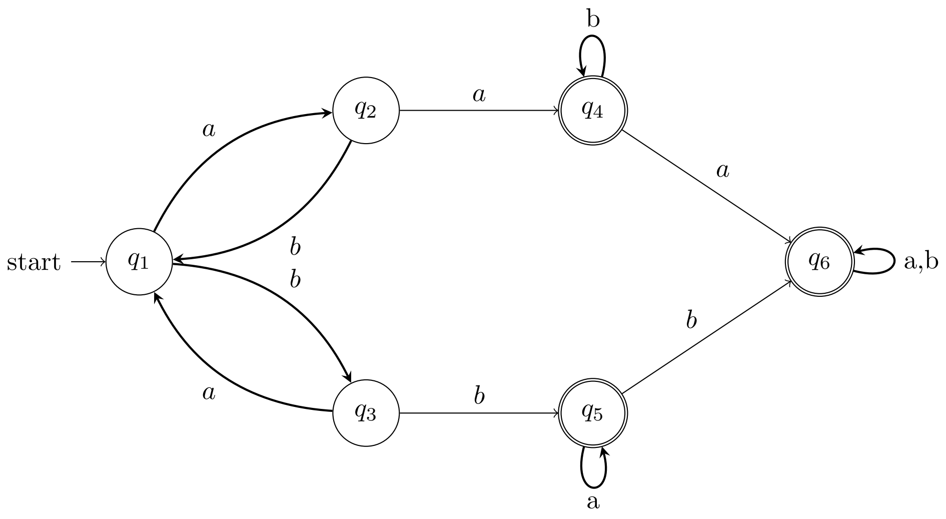


NFA to accept L

□

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

   Solution:



Drawing transition table -

state/input	a	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$_6$

We will start by making partitions and in each step we will breakdown the partitions into the smaller partitions which will eventually land up into equivalence classes.

Property:
Two states will be in same set of current partition if on reading x $\forall x \in \Sigma$ both land up to same set of previous partition. – [1]

Step 1: First, we will partition the states into two partition of accepting q_1, q_2, q_3 and non-accepting q_4, q_5, q_6 as two different sets-

$$Q = \{\{q_1, q_2, q_3\}, \{q_4, q_5, q_6\}\}$$

Step 2: Now we will make smaller partition, using - property[1] after considering each pair-

For non-accepting states-

(a) q_1, q_2: from transition table we can see that, upon reading a in different set. Thus, q_1 and q_2 must be in different partition.

(b) q_1, q_3: from transition table we can see that, upon reading a is same set where as for b both land in different set. Thus, q_1 and q_3 must be in different partition.

(c) q_2, q_3: from transition table we can see that, upon reading a in different set. Thus, q_2 and q_3 must be in different partition.

For accepting states-

(a) q_4, q_5: from transition table we can see that, upon reading a as well as b, they land up in same set. Thus, q_4 and q_5 can be same partition.

(b) q_4, q_6: from transition table we can see that, upon reading a as well as b, they land up in same set. Thus, q_4 and q_6 can be same partition.

(c) q_5, q_6: from transition table we can see that, upon reading a as well as b, they land up in same set. Thus, q_5 and q_6 can be same partition.

Thus, new partition is -

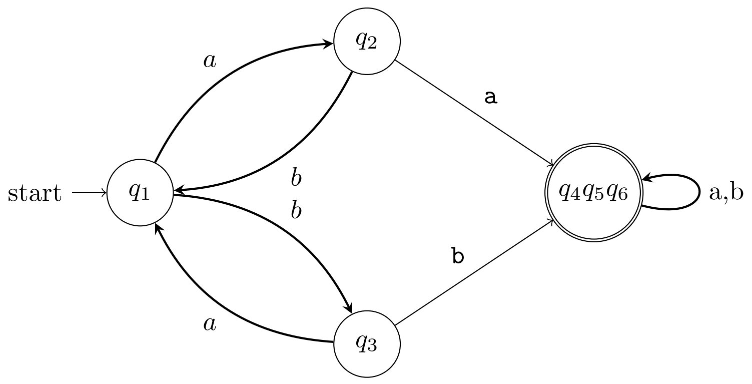
$$Q = \{\{q_1\}, \{q_2\}, \{q_3\}, \{q_4, q_5, q_6\}\}$$

Step 3: Here, we can see that, new partitions so formed are either singleton or same as earlier handled partition.

Thus, final partition is -

$$Q = \{\{q_1\}, \{q_2\}, \{q_3\}, \{q_4, q_5, q_6\}\}$$

The minimized DFA has **4** states where $\{q_4, q_5, q_6\}$ is merged into a single state $q_4q_5q_6$ -



□

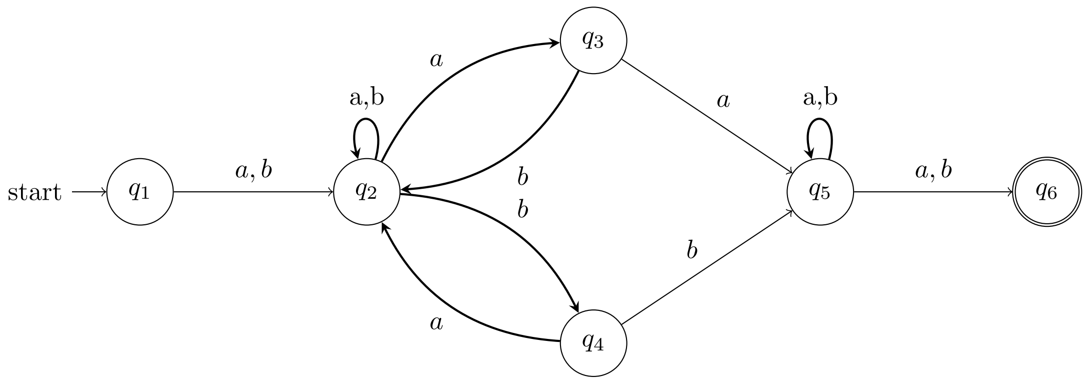
10. language $L = \{uvv^r w | (u, v, w) \in \{a, b\}^+\}$ is regular
    where, v^r is the reverse of v

    (a) a **regular expression** whose language is L

    (b) conversion of the regular expression to an **equivalent NFA**

    Solution:

    For any string v where v $\in \{a, b\}^+, v^r$ must start with the last letter of v, so vv^r must have an appearance of aa or bb, so we can represent $uvv^r w$ as $(a + b)(a + b)^*(aa + bb)(a + b)^*(a + b)$, we are taking $(a + b)$ in start and end to ensure that u and w cannot be empty string.



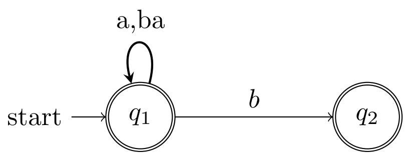
□

11. **Equivalent languages**

    Solution:

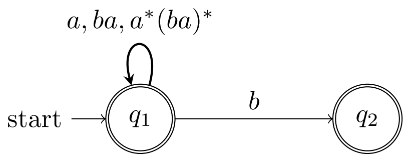
    We can make corresponding GNFAs to check if these regular expressions are equivalent or find a counter example to check if they are not.

    a



b

$$(a^*(ba)^*)^*(b+\epsilon) + a^*(b+\epsilon) + (ba)^*(b+\epsilon)$$
$$= ((a^*(ba)^*)^* + a^* + (ba)^*)(b+\epsilon)$$



Here, we can remove the $a^*(ba)^*$ transition as it can be balanced by a, ba

So, it will also be same as (a)

c String s = "b" is not accepted by this regular expression but it is accepted by (a) and (b). So, it is not equivalent to (a) and (b).

Final answer is **(a) and (b) are equivalent**.

□

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) We can keep track of last 2 places where there is 0 . Say 10010 has last two zeroes at index 0 and 2 (0 indexing) which we will use to define our states

$$\text{FA} = (Q, q_{start}, \sum, \delta, F)$$

$\Sigma = \{0, 1\}$
Q = $\{q_{start}, q_1, q_2, q_3, q_{01}, q_{01}, q_{02}, q_{03}, q_{04}, q_{12}, q_{13}, q_{14}, q_{23}, q_{24}, q_{34}, r_1, r_{11}, r_{111}, r_{1111}, q_{dead}\}$

where

q_i $i \in [0, 3]$ are states when we have only one 0 out of 5 consecutive symbols which is at index i. q_4 is not included here since, it already completes the string of length five with only one 0 thus unacceptable.
q_{ij} $i, j \in [0, 4], i < j$ are states when we have more than one 0 out of 5 consecutive symbols out of which last 2 zeroes are at index i and j.

q_{dead} is state which has strings which cannot be accepted

r_{1s} are states which have length less than 4 and are comprised of only 1s, we are taking care of it as these do not fall in q_{dead}, $111 = r_{111}$ and so on

We will take only those strings to be accepted which has atleast 2 0s in all consecutive 5 symbols if its length is greater than 4 otherwise if a string has less than 5 symbols, we will accept it if it has 2 or more 0s.

$$F = \{q_{01}, q_{01}, q_{02}, q_{03}, q_{04}, q_{12}, q_{13}, q_{14}, q_{23}, q_{24}, q_{34}\}$$

We will represent δ using this transition table

state/input	0	1
q_{start}	q_0	r_1
r_1	q_0	r_{11}
r_{11}	q_0	r_{111}
r_{111}	q_0	q_{dead}
q_0	q_{01}	q_1
q_1	q_{02}	q_2
q_2	q_{03}	q_3
q_3	q_{04}	q_{dead}
q_{01}	q_{01}	q_{12}
q_{02}	q_{01}	q_{13}
q_{03}	q_{01}	q_{14}
q_{04}	q_{01}	q_1
q_{12}	q_{02}	q_{23}
q_{13}	q_{02}	q_{24}
q_{14}	q_{02}	q_2
q_{23}	q_{03}	q_{34}
q_{24}	q_{03}	q_3
q_{34}	q_{04}	q_{dead}
q_{dead}	q_{dead}	q_{dead}

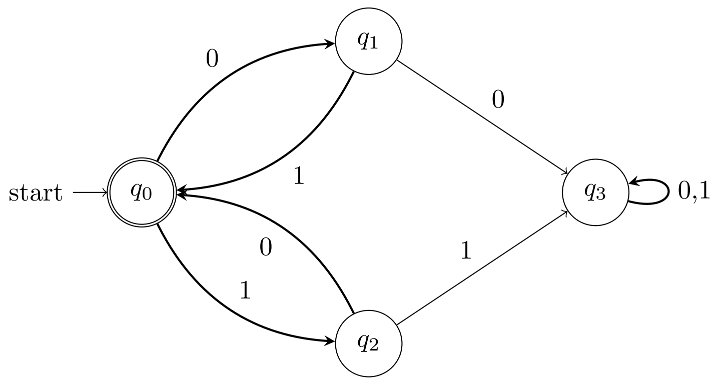
(b)

$$FA = (Q, q_0, \Sigma, \delta, F)$$

where

$$\Sigma = \{0, 1\}$$
$$Q = \{q_0, q_1, q_2, q_3\}$$

where
q_0 has equal apperances of 0 and 1 and hence an accepting state
q_1 has one more 0 than 1
q_2 has one more 1 than 0
q_3 all other strings



□

13. **Given that language L is regular, is the language $L_{1/2} = \{x|y$ s.t $|x| = |y|$, xy $\in L\}$ regular? If yes, give a formal proof.** :

    Solution:

    Given L is regular,
    Assume ∃ DFA M = $(Q, \Sigma, \delta, q_0, F)$ which accepts L.

    Define M' = $(Q', \Sigma, \delta', q_0', F')$.
    Where, Q' is the set of pairs of q and S. q is the state where M would have landed upon reading a string x. S is set of all states of M which upon reading any string of length = $|x|$ would land up in one of the final states of M.

    $q_0' = [q_0, F]$ as only states in F upon reading string of length 0 i.e ϵ would land up in F.

    We want x to be accepted if ∃ y s.t $|x| = |y|$ and xy ∈ L.
    So our final states should be [q, S] where q ∈ S.

    As upon reading x we land in q in M, and we want that we can reach final state upon reading any string of length $|x|$ from q.
    $\delta'([q_1, S], \lambda) = [\delta(q_1), S']$.
    Where S' is the collection of all such states of M from where reading any symbol we can go to any one state in S.

    We can easily prove this transition will uphold our new DFA using induction:

    Base Case : After reading no symbol, we stay in q_0' which we have defined keeping in mind our DFA.

    Induction step : After reading string x , we are in state $[q_1, S]$, where S holds the DFA definition, now after reading a symbol we go to state $[\delta(q_1), S']$.

    We know S' is the collection of all such states of M from where reading any symbol we can go to any one state in S.

    So from any state in S' we can reach F using number of symbols required from any state in S + 1 which proves that our transition holds the definition of DFA

    So now we have a DFA that accepts $L_{1/2}$ which proves that **it is regular**.

□

14. **The Regular language L is closed under reverse**

    Solution:

    Since L is regular, ∃ DFA M = $(Q, \Sigma, \delta, q_0, F)$ that recognizes it.

    Using this we construct a NFA M' = $(Q \cup \{q_0'\}, \Sigma, \delta', q_0', \{q_0\})$ that recognizes L' such that,

(a) New start state can ϵ -transition to all previous final states i.e
$\delta'(q_0', \epsilon) = F$

(b) Old start state of M is the only final state of M' i.e
$\delta'(q_0', a) = \phi \; \forall \; a \in \Sigma$

(c) $\delta'(p, a) = \{q | \delta(q, a) = p\} \; \forall \; q \in Q, a \in \Sigma$

To prove that Language L' is regular, we show that if $w \in L \iff w' \in L'$

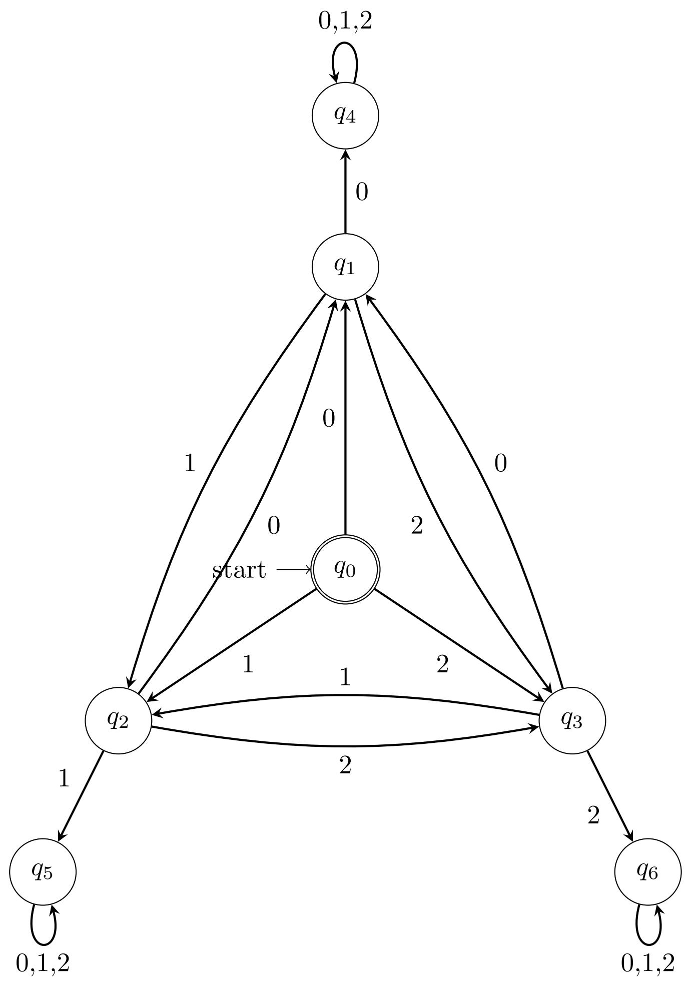
$\Rightarrow$
Since $w \in L$, and $w = w_1 w_2 .. w_n$ we have states r_i s.t $r_i = \delta(r_{i-1}, w_i)$
We can write $w' = \epsilon w_n w_{n-1} .. w_1$ with the state sequence $q_0' r_n ... r_1$.
Now we show that each state sequence is valid and thus w' is accepted.
$r_n \in \delta(q_0', \epsilon) \implies r_n \in F$ by (a)
$r_{i-1} \in \delta'(r_i, w_i) \implies r_{i-1} \in \{q | \delta(q, w_i) = r_i\}$ by (c)
Thus each transition is valid for NFA M', and w' is accepted.

$\Leftarrow$
Since $w \in L'$, and $w = w_1 w_2 .. w_n$ we have states r_i s.t $r_0 = q_0'$, $r_n \in \{q_0\}$ and $r_{i+1} = \delta'(r_i, w_{i+1})$
We know that $w - 1 = \epsilon$ and $r_1 = F$, we need to show that M accepts $w_n w_{n-1} .. w_2$ with state sequence $r_n ... r_1$
Since $w \in L'$, $r_i \in \delta'(r_{i-1}, w_i) \implies r_i \in \{q | \delta(q, w_i) = r_{i-1}\}$ and so $\delta(r_i, w_i) = r_{i-1}$
Thus each transition is valid and $w' \in L$

□

15. **Let L be the language over the alphabet $\{0, 1, 2\}$ such that for any string in s $\in$ 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:



$\square$

16. **Regularity under homomorphism**

(a) **Solution:**

Given $h : \Sigma^* \Rightarrow \Delta^*$ s.t
$h(\epsilon) = \epsilon$
$h(wa) = h(w)h(a)$

We define homomorphism h as an operation on regexp such that, for any regular expression R, h(R) be the regular expression obtained by replacing each occurence of $a \in \Sigma$ in R by h(a) i.e
$h(\epsilon) = \epsilon$,
$h(R_1R_2) = h(R_1)h(R_2)$

Clearly,
$h(R_1 \cup R_2) = h(R_1) \cup h(R_2)$
$h(R^*) = h(R)^*$

We define,

$$f : (Regular\ \ expression) \rightarrow Language$$

$f(R) = L$ –[1]
$f(h(R)) = h(L)$ –[2]

From [1], $h(L) = h(f(R))$
From [2],

$$h(f(R)) = f(h(R)) \qquad -[3]$$

We need to show that f(h(R)) = h(f(R))
Using induction on the number of operations in R.

Base Case: $R \in \{\epsilon\}$. then,

$$f(R) = \{\}$$
$$h(R) = \epsilon = R$$
$$h(f(R)) = h(R) = f(h(R))$$

Now for R=a,

$$f(R) = a$$
$$h(f(R)) = h(\{a\}) = \{h(a)\} = f(h(a)) = f(h(R))$$

Hence, base case proved

Induction Step: Assuming eq [3] holds for $R_1 \& R_2$, taking operations on $R_1 \& R_2$ we have three cases-

(a) $R = R_1 \cup R_2$ then h(R) = $h(R_1) \cup h(R_2)$

$$h(f(R)) = h(f(R_1 \cup R_2)) = h(f(R_1) \cup f(R_2))$$

From definition-

$$h(f(R_1) \cup f(R_2)) = h(f(R_1)) \cup h(f(R_2))$$

Using Induction Hypothesis,

$$h(f(R_1)) \cup h(f(R_2)) = f(h(R_1)) \cup f(h(R_2))$$

from definition of R,

$$f(h(R_1)) \cup f(h(R_2)) = f(h(R_1) \cup h(R_2)) = f(h(R_1 \cup R_2))$$

Thus,

$$h(f(R_1 \cup R_2)) = f(h(R_1 \cup R_2))$$

(b) $R = R_1R_2$, then h(R) = $h(R_1)h(R_2)$

$$h(f(R_1R_2)) = h(f(R_1)f(R_2)) = h(f(R_1))h(f(R_2))$$

By induction hypothesis, since R_1 and R_2 holds,

$$h(f(R_1))h(f(R_2)) = f(h(R_1))f(h(R_2))$$
$$f(h(R_1))f(h(R_2)) = f(h(R_1)h(R_2)) = f(h(R_1R_2))$$

Thus,

$$h(f(R_1R_2)) = f(h(R_1R_2))$$

(c) $R = R_1^*$, then h(R) = $h(R_1)^*$

$$h(f(R_1^*)) = h(f(R_1))^*) = h(f(R_1)))^*$$

By induction hypothesis,

$$h(f(R_1)))^* = f(h(R_1)))^* = f(h(R_1))^*) = f(h(R_1)^*)) = f(h(R_1^*)))$$

Thus,

$$h(f(R_1^*)) = f(h(R_1^*)))$$

Proved induction step. □

(b) **Solution:**

(a) Define M = $(Q, \Sigma, \delta, q_0, F)$ that accepts L.

(b) Define M' = $(Q, \Sigma, \delta', q_0, F)$ where the new δ takes a input and outputs the homomorphed solution i.e
$\delta'(q, a) = q'$ where $\hat{\delta_M}(q, h(a)) = \{q'\}$

(c) **Base Case**: a=ϵ, then $\delta'(q, a) = q_0, \delta_M(q_0, h(\epsilon)) = \delta_M(q_0, \epsilon) = q_0$

(d) **Induction Step**: $a = a_1a_2$, then $\hat{\delta_{M'}}(q_0, a) = \hat{\delta_{M'}}(\delta_M(q_0, a_1), a_2) = \hat{\delta_M}(\delta_M(q_0, h(a_1)), h(a_2)) = \hat{\delta_{M'}}(q_0, h(a_1a_2)) = \hat{\delta_M}(q_0, h(a))$

□