CONSTRUCTION OF FINITE AUTOMATA EQUIVALENT TO A REGULAR EXPRESSION

(Subset Method)

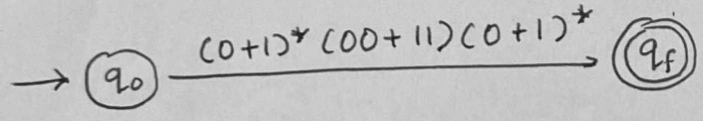
Step 1: Construct a transition graph (transition system) equivalent to the given r.e. using Λ-moves. This is done using Kleene's Theorem.

Step 2: Construct the transition table for the transition-graph obtained in step 1. ~~Using~~ Then, we can construct an equivalent DFA (NDFA ⟶ DFA). We reduce the # of states if possible.

Ques :- Construct a finite automata equivalent to r.e.

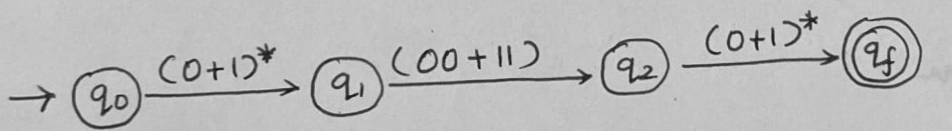
$$(0+1)^* (00+11) (0+1)^*$$

1)

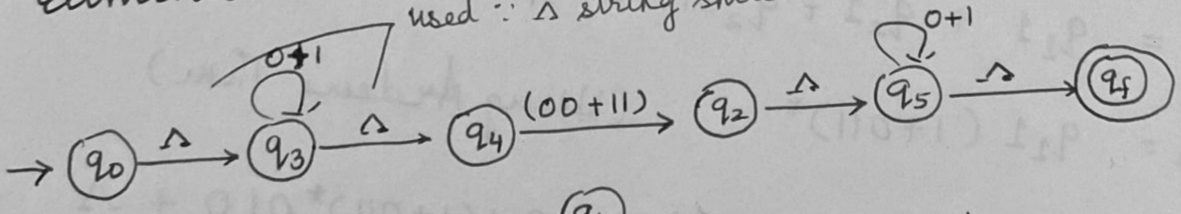


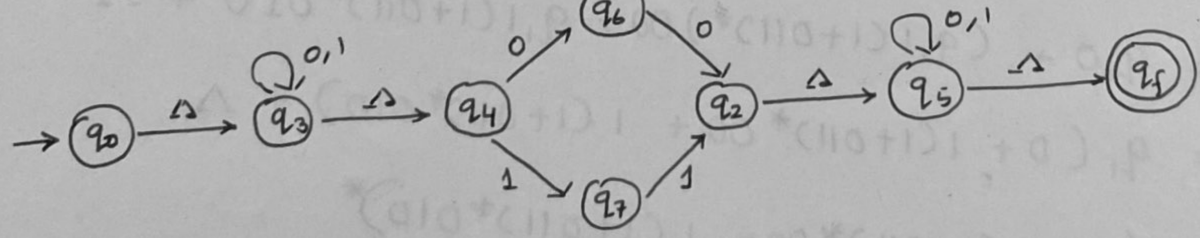
(construct a transition with Λ-moves

2) Remove concatenations.

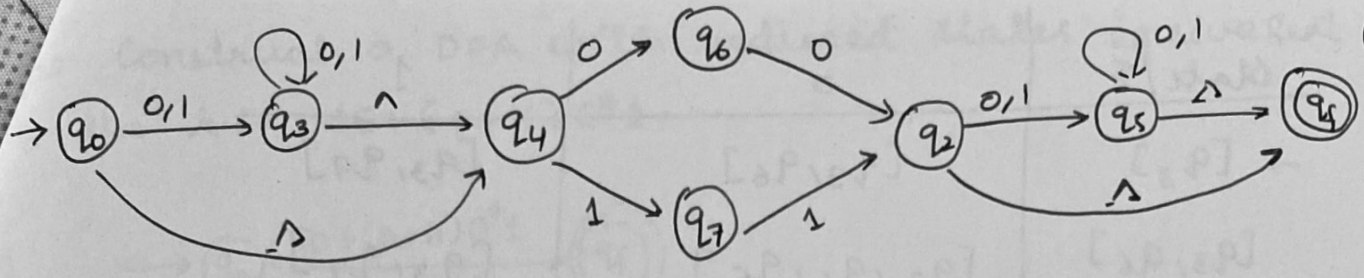


3) Eliminate * operations by introducing Λ-moves.
used ∵ Λ string should be accepted.

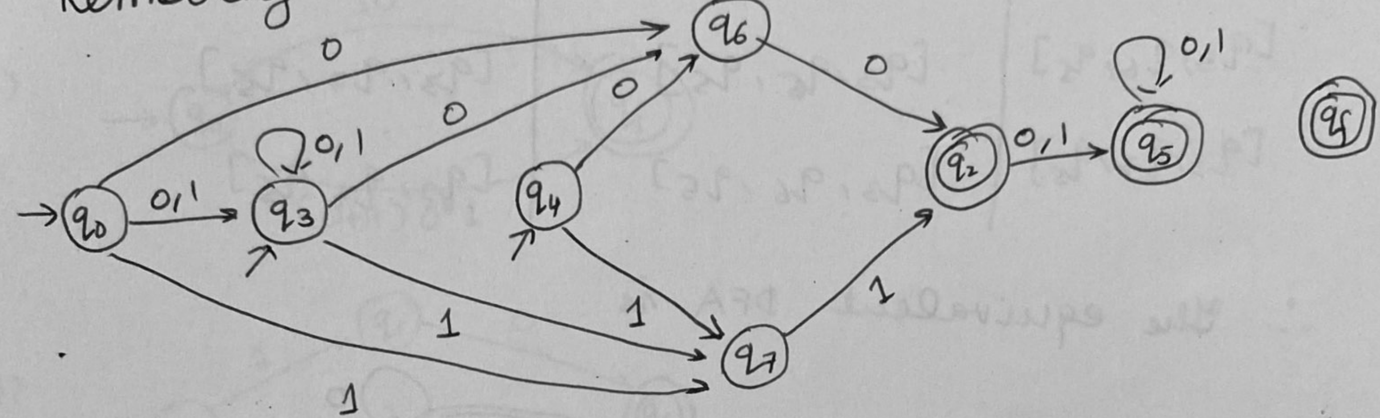




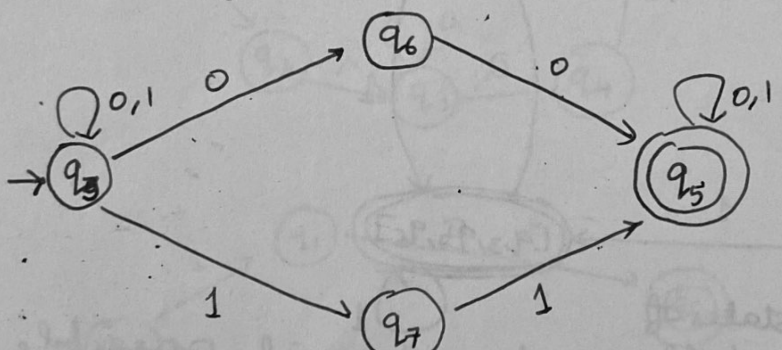
4) Next, we remove Λ-moves.



Removing Λ-transitions



Removing futile (useless) states.



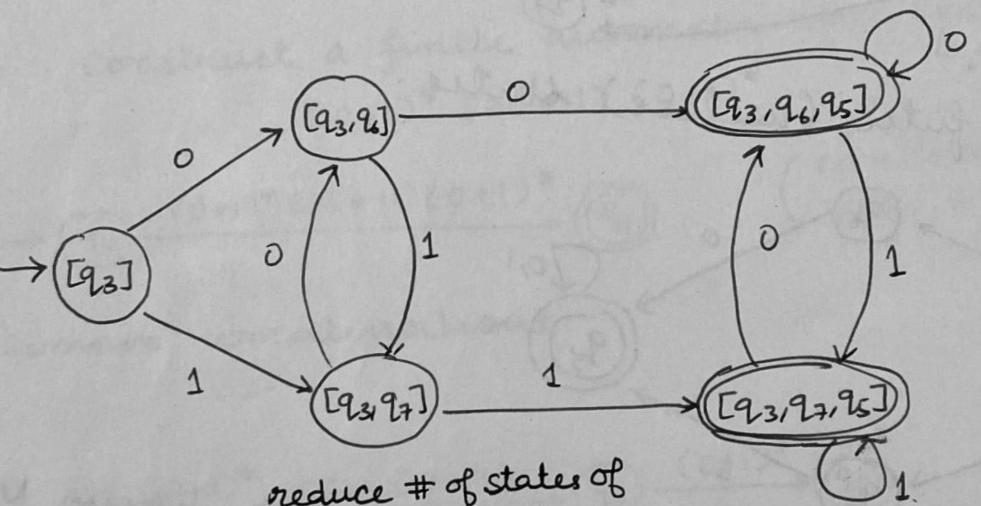
Next, we construct DFA corresponding to above NDFA represented by :-

state / Σ	0	1
→ q_3	q_3, q_6	q_3, q_7
q_6	q_5	
q_7		q_5
q_5	q_5	q_5

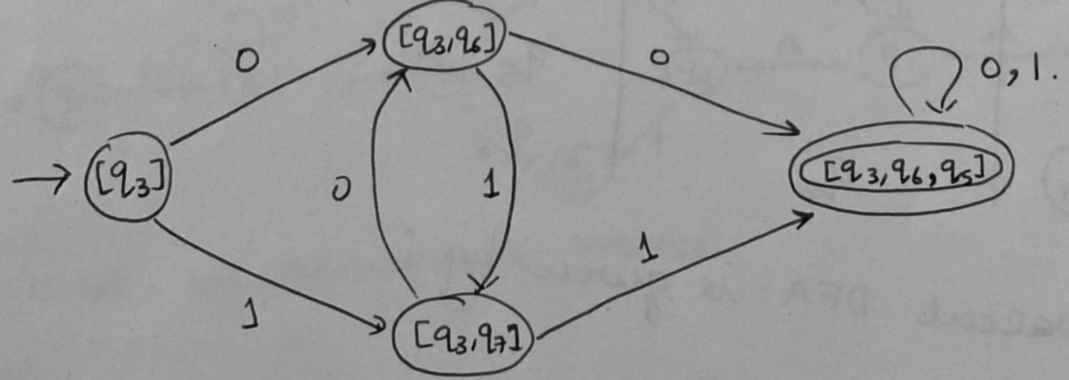
The equivalent DFA is given by :-

State/Σ	0	1
→ $[q_3]$	$[q_3, q_6]$	$[q_3, q_7]$
$[q_3, q_6]$	$[q_3, q_6, q_5]$	$[q_3, q_7]$
$[q_3, q_7]$	$[q_3, q_6]$	$[q_3, q_7, q_5]$
$[q_3, q_6, q_5]$	$[q_3, q_6, q_5]$	$[q_3, q_7, q_5]$
$[q_3, q_7, q_5]$	$[q_3, q_6, q_5]$	$[q_3, q_7, q_5]$

∴ The equivalent DFA is

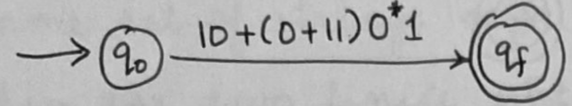


Finally, we ~~minimize~~ reduce # of states of the automata, if possible.
This is possible when 2 rows are identical in the table representing transition function of equivalent DFA.
∵ rows corr. to $[q_3, q_6, q_5]$ & $[q_3, q_7, q_5]$ are identical, delete one of the rows.

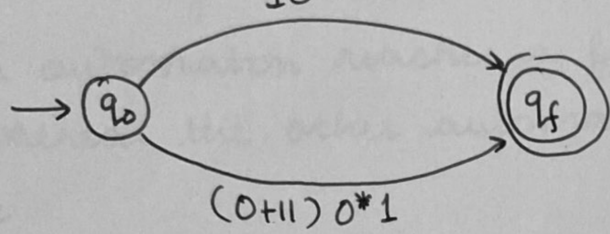


Construct a DFA with reduced states equivalent to the r.e. $10 + (0+11)0^*1$.

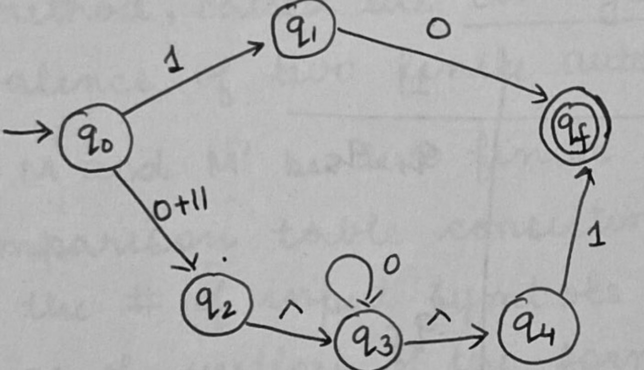
1)



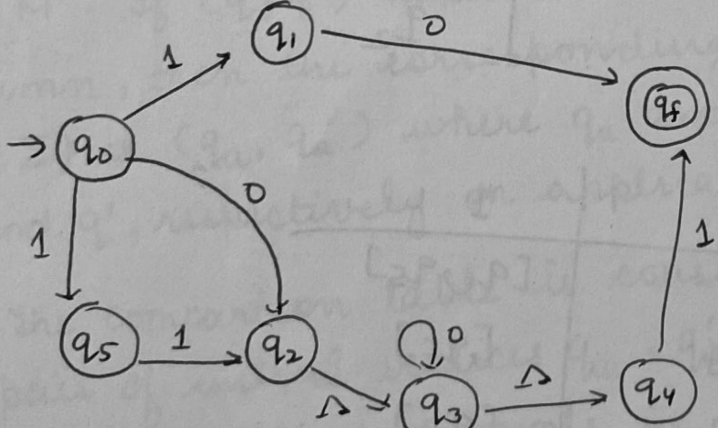
2)



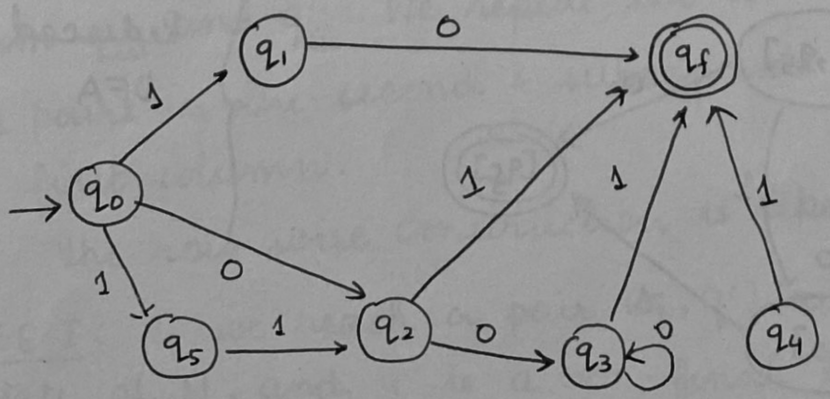
3)



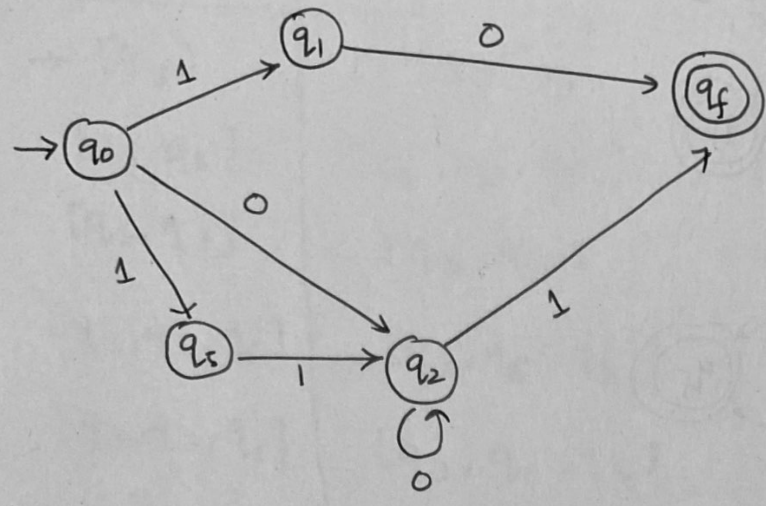
4)



Eliminating Λ-transitions.



Removing futile transitions :

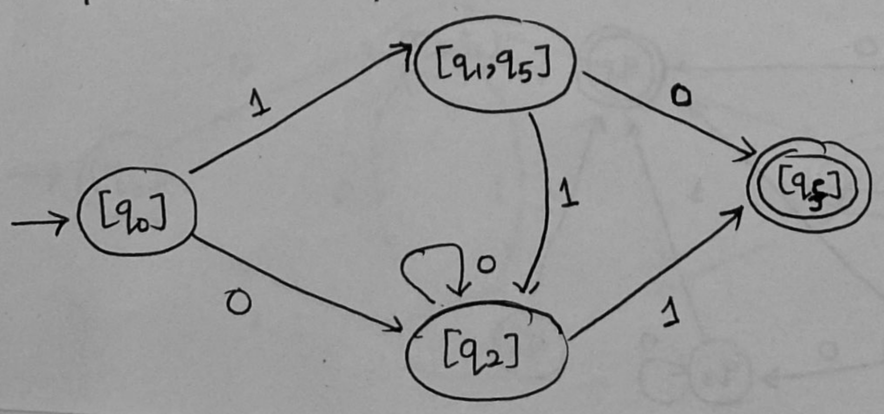


Next is construction of DFA

State/Σ	0	1
$\rightarrow q_0$	q_2	q_1, q_5
q_1	q_f	
q_2	q_2	q_f
q_f		
q_5		q_2

Equivalent DFA is

State/Σ	0	1
$\rightarrow [q_0]$	$[q_2]$	$[q_1, q_5]$
$[q_2]$	$[q_2]$	$[q_f]$
$[q_1, q_5]$	$[q_f]$	$[q_2]$
$[q_f]$	ϕ	ϕ



Reduced DFA

EQUIVALENCE OF TWO FINITE AUTOMATA

Two finite automata over Σ are equivalent if they accept the same set of strings over Σ.

When the two finite automata are not equivalent there is some string w over Σ satisfying the following:

One automaton reaches a final state on application of w, whereas the other automaton reaches a non final state.

A method, called the COMPARISON METHOD, to test the equivalence of two finite automata over Σ.

Let M and M' be two finite automata over Σ. We construct a comparison table consisting of $(n+1)$ columns, where n is the # of input symbols. The first column consists of pairs of vertices of the form (q, q') where $q \in M$ and $q' \in M'$. If (q, q') appears in some row of the first column, then the corresponding entry in the a-column $(a \in \Sigma)$ is (q_a, q_a') where q_a and q_a' are reachable from q and q', respectively on application of a (i.e. by a-paths)

The comparison table is constructed by starting with the pair of initial vertices q_{in}, q_{in}' of M and M' in the first column. The first elements in the subsequent columns are (q_a, q_a') where q_a and q_a' are reachable by a-paths from q_{in} and q_{in}'. We repeat the construction by considering the pairs in the second & subsequent columns which are not in first column.

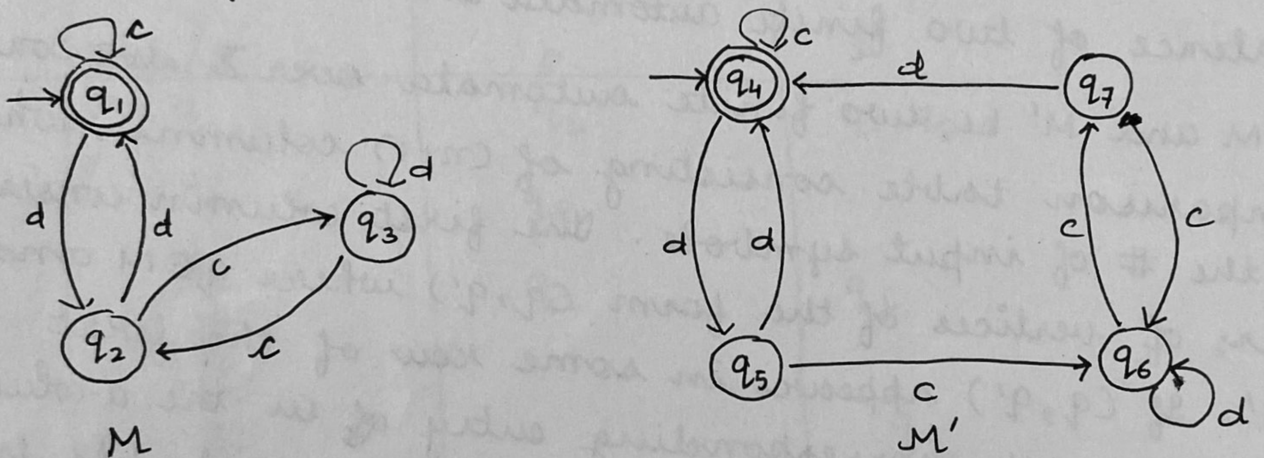
The row wise construction is repeated. There are 2 cases:-

CASE I: If we reach a pair (q, q') such that q is a final state of M, and q' is a non-final state of M' or vice-versa.

we terminate the construction and conclude that M a[nd]
M' are not equivalent.

CASE 2: Here the construction is terminated when no new element appears in the second and subsequent columns which are not in the first column (i.e. when all the elts. in the second & subsequent columns appear in the first column). In this case, we conclude that M and M' are equivalent.

Ques:- Consider the following 2 DFAs M & M' over {0,1} given in figures below. Determine whether M & M' are equivalent.

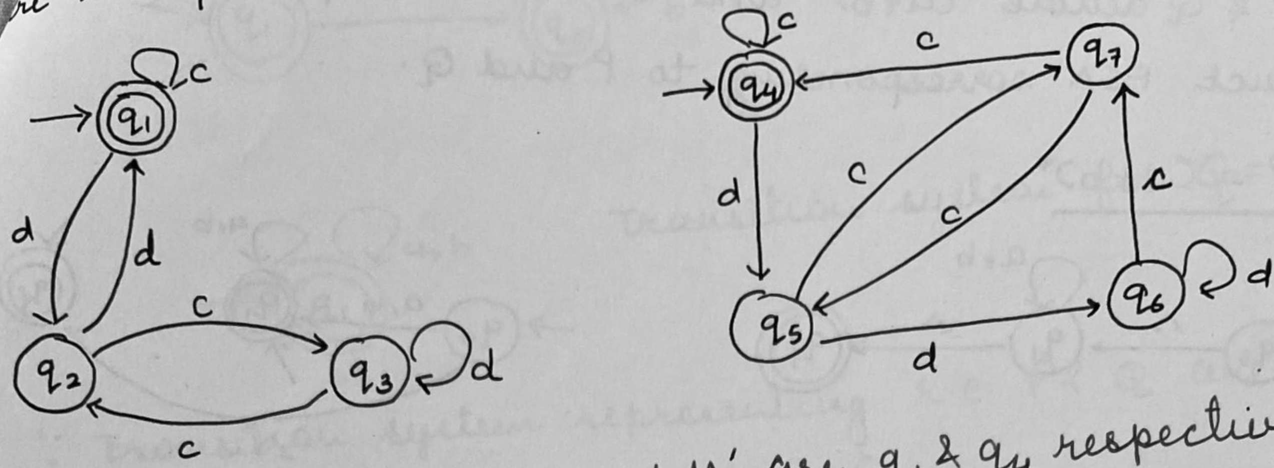


Sol^n:- The initial state of M & M' are q_1 & q_4 respectively.

(q, q')	(q_c, q_c')	(q_d, q_d')
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

All the pair (q, q') appearing in 2nd & 3rd column also appears in the first column & we don't get a pair (q, q') where q is a final state & q' is a non-final state or vice-versa. ∴ M and M' are equivalent.

Show that the automata M and M′ given below are not equivalent.



The initial states of M and M′ are q_1 & q_4 respectively.

(q, q')	(q_c, q_c')	(q_d, q_d')
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_7)	(q_1, q_6)

As q_1 & q_6 are d-reachable from q_2 and q_5 respectively where q_1 is a final state while q_6 is a non-final state.
∴ M and M′ are not equivalent.

EQUIVALENCE OF TWO REGULAR EXPRESSIONS

2 regular expressions P and Q are equivalent iff they represent the same set. Also, P and Q are equivalent iff the corresponding finite automata are equivalent.
Note that we can also prove the equivalence of P and Q using the identities.

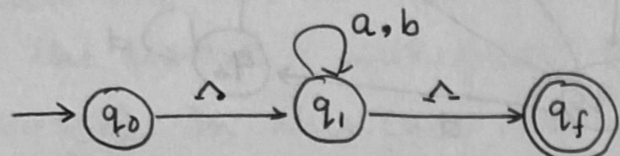
here, for non-equivalence find a string in one set which is not in the other set.

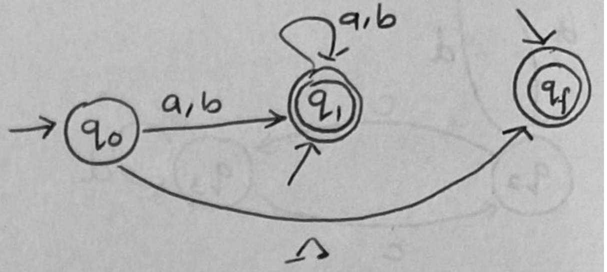
Ques:- Prove that $(a+b)^* = a^*(ba^*)^*$

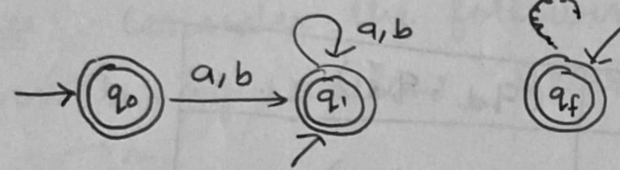
Let P & Q denote $(a+b)^*$ and $a^*(ba^*)^*$ respectively. W

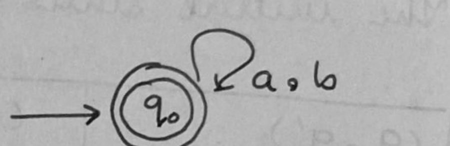
construct FSA corresponding to P and Q.

For $P = (a+b)^*$



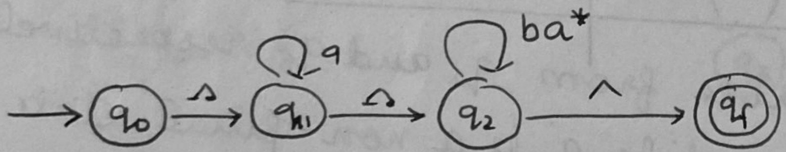


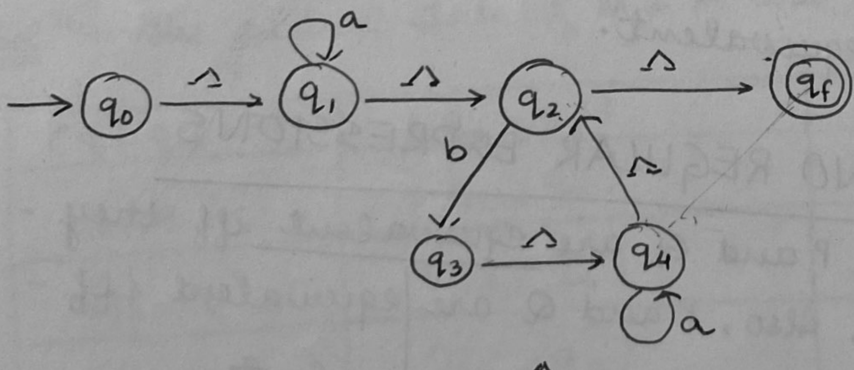


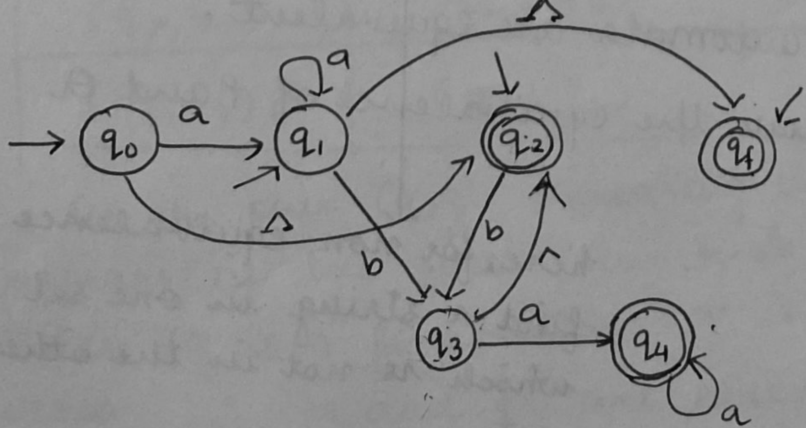


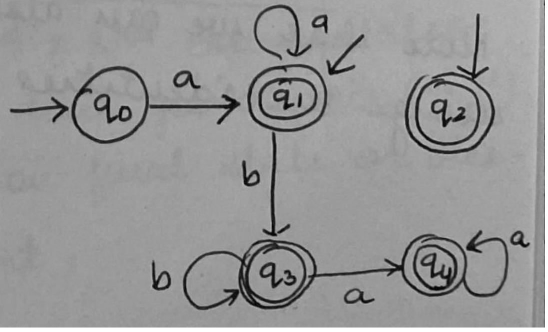
Transition system for P

For $Q = a^*(ba^*)^*$

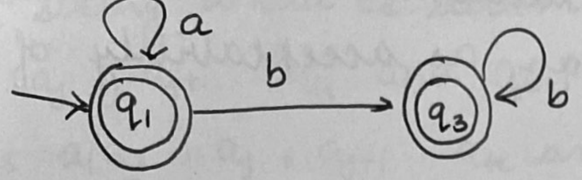


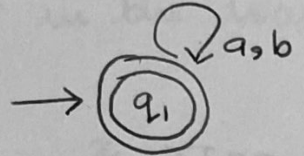






Scanned with CamScanner





Transition system for $Q = a^*(ba^*)^*$

$\because$ Transition system representing r.e. P & Q are same

$\therefore$ P and Q are equivalent or $P = Q$.

RESULTS :

1. Every r.e. is recognized by a transition system.
2. A transition system M can be converted into a finite automata accepting the same set as M.
3. Any set accepted by finite automaton is represented by an r.e.
4. A set accepted by a transition system is represented by an r.e. (from 2. & 3.)
5. To get r.e. representing a set accepted by a transition system, we can apply the algebraic method using the Arden's Theorem.
6. If P is an r.e., then to construct a finite automaton accepting the set P.
7. A subset L of Σ^* is a regular set (or represented by an r.e.) iff it is accepted by an FA. (from 1, 2 and 3).
8. A subset L of Σ^* is a regular set iff it is recognized by a transition system (from 1 & 4).

9. The capabilities of finite automata & transition systems are the same as far as acceptability of su[ch] of strings is concerned.

10. To test the equivalence of 2 DFAs, we can apply the comparison method.

PUMPING LEMMA FOR REGULAR SETS

Pumping Lemma gives a necessary condition for an input string to belong to a regular set.

It is called Pumping Lemma as it gives a method of pumping (generating) many input strings from a given string.

Pumping Lemma can be used to show certain sets are not regular (as it is a necessary condition).

THEOREM :- PUMPING LEMMA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with n-states. Let L be the regular set accepted by M. Let $\omega \in L$ and $|\omega| \geq m$. If $m \geq n$, then $\exists\ x, y, z$ s.t. $\omega = xyz$, $y \neq \Lambda$ and $xy^i z \in L$ for each $i \geq 0$.

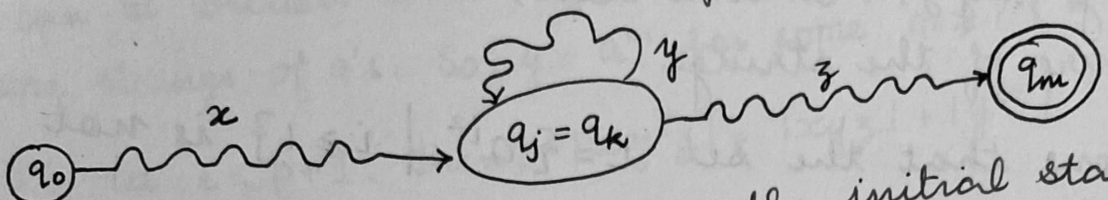
PROOF :- Let $\omega = a_1 a_2 \dots a_m$, $m \geq n$

$\delta(q_0, a_1 a_2 \dots a_i) = q_i$ for $i = 1, 2, \dots, m$; $Q_1 = \{q_0, q_1, \dots, q_m\}$

i.e. Q_1 is the ~~set of~~ sequence of states in the path with path value $\omega = a_1 a_2 \dots a_m$. As there are only n distinct states, at least 2 states in Q_1 must coincide. Among the various pairs of repeated states, we take the first pair. Let us take them as q_j and q_k ($q_j = q_k$). Then, j and k satisfy the condition $0 \leq j < k \leq n$.

Scanned with CamScanner

The string ω can be decomposed into 3 substrings - $a_1 a_2 \cdots a_j$, $a_{j+1} \cdots a_k$ and $a_{k+1} \cdots a_m$. Let x, y, z denote these strings $a_1 a_2 \cdots a_j$, $a_{j+1} \cdots a_k$ and $a_{k+1} \cdots a_m$ respectively. As $k \leq n$, $|xy| \leq n$ and $\omega = xyz$. The path with the path value w in the transition diagram of M is shown below



The automaton M starts from the initial state q_0. On applying the string x, it reaches q_j $(= q_k)$. On applying the string y, it comes back to q_j $(= q_k)$. So, after application of y^i for each $i \geq 0$, the automaton is in the same state q_j. On applying z, it reaches q_m, a final state. Hence, $xy^iz \in L$. As every state in Q_1 is obtained by applying an input symbol, $y \neq \wedge$

<u>NOTE</u> :- The decomposition is valid only for strings of length greater than or equal to the # of states.

The case when $i = 0$ corresponds to the path from q_0 to q_k and then the path from q_k to q_m (without going through the loop), we get a path ending in a final state with path value xz.

APPLICATION OF PUMPING LEMMA

This theorem is used to prove that certain sets are not regular.

<u>STEP 1</u> : Assume that L is regular. Let n = # of states in the corresponding finite automaton.

<u>STEP 2</u>: Choose a string w s.t. $|w| \geq n$. Use Pumping lemma to write $\omega = xyz$, with $|xy| \leq n$ & $|y| > 0$.

STEP 3: Find a suitable integer i s.t. $xy^iz \notin L$
This contradicts our assumption. Hence, L is not regu[lar]

Note that the crucial part of the procedure is to find i s.t. $xy^iz \notin L$. In some cases, we prove $xy^iz \notin L$ by considering $|xy^iz|$. In some cases, we may have to use the structure of the string.

Ques:- Show that the set $L = \{a^{i^2} \mid i \geq 1\}$ is not regular.

Soln:- Step 1: Suppose L is regular. Let n be the number of states in the finite automata accepting L.

Step 2: Let $\omega = a^{n^2}$. Then $|\omega| = n^2 > n$. By Pumping Lemma, we can write $\omega = xyz$ with $|xy| \leq n$ & $|y| > 0$.

Step 3: Consider xy^2z,

$$|xy^2z| = |x| + 2|y| + |z| > |x| + |y| + |z| \text{ as } |y| > 0$$

This means, $n^2 = |xyz| = |x| + |y| + |z|$
$$< |xy^2z|.$$

As $|xy| \leq n$ we have $|y| \leq n$. Therefore,

$$|xy^2z| = |x| + 2|y| + |z| \leq |x| + |y| + |z| + |y|$$
$$\leq n^2 + n$$

i.e.
$$n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$$

$$\Rightarrow \quad n^2 < |xy^2z| < (n+1)^2$$

$\Rightarrow |xy^2z|$ strictly lies b/w n^2 & $(n+1)^2$, but is not equal to any one of them. Thus, $|xy^2z|$ is not a perfect square & so $xy^2z \notin L$. By by pumping lemma, $xy^2z \in L$. This is a contraction. Hence, L is not regular.

:- Show that $L = \{a^p \mid p \text{ is a prime}\}$ is not regular.

:- Step 1: Suppose that L is regular. Let n be the # of states in the final automata accepting L.

Step 2: Let p be a prime no. $> n$. Let $\omega = a^{n^2}$. By Pumping lemma ω can be written as $\omega = xyz$ with $|xy| \le n$ and $|y| > 0$. x, y, z are strings of a's. So, $y = a^m$ for some $m \geq 1$ (& $\le n$)

Step 3: Let $i = p+1$. Then, $|xy^iz| = |xyz| + |y^{i-1}|$

$$:= p + (i-1)m = p + pm$$

By pumping lemma, $xy^iz \in L$.

But $|xy^iz| = p + pm = p(1+m)$, which is not prime.

So, $xy^iz \notin L$, which is a contradiction

Hence, L is not regular.

Ques:- Show that $L = \{0^i 1^i \mid i \geqslant 1\}$ is not regular.

Sol^n:- Step 1: Suppose L is regular. Let n = # of states in the finite automaton accepting L.

Step 2: Let $\omega = 0^n 1^n$. Then, $|\omega| = 2n > n$. By Pumping lemma we write $\omega = xyz$ with $|xy| \le n$ & $|y| \neq 0$.

Step 3: We want to find i so that $xy^iz \notin L$ for getting a contradiction. case1: The string $y = 0^k$ for some $k \geqslant 1$.

Then, taking $i = 0$ in $xy^iz = xz$

$$= 0^{n-k} 1^n, \quad k \geqslant 1$$

So, $xz = xy^0z \notin L$ (as powers of 0 & 1 are unequal)

$\therefore$ L is not regular.

NO NEED These cases can occur for other ω.

case 2: $y = 1^l, l \geqslant 1$

$xy^0z \notin L$

case 3: $y = 0^k 1^j$, $k, j \geqslant 1$

$xy^2z = 0^{n-k} 0^k 1^j 0^k 1^j 1^{n-j} = 0^n 1^j 0^k 1^{n-j}$

$j \neq k$; $xy^2z \notin L$

Ques Show that $L = \{\omega\omega \mid \omega \in \{a,b\}^*\}$ is not regular.

Step 1: Suppose L is regular & n = # of states in automata accepting L.

Step 2: Let us consider $ww = a^n b a^n b$ in L. $|ww| = 2(n+1)$
We can apply pumping lemma to write $w = xyz$ with $|y$
& $|xy| \leq n$.

Step 3: To find i s.t. $xy^iz \notin L$. The string y can be of the form :-

(a) $y = a^k$; $k \geq 1$.

In this case ; $xy^0z = a^{n-k} b a^n b$, $k \geq 1$.

$\because$ $xy^0z = xz$ is not of the form uu with $u \in \{a,b\}^*$. So, $xz \notin L$.

(b) $y = b$ ← (No need) $\because |xy| \leq n$ $\therefore$ y form is a^i

In this case, $xy^0z = a^n a^n b$; again $xy^0z \notin L$.

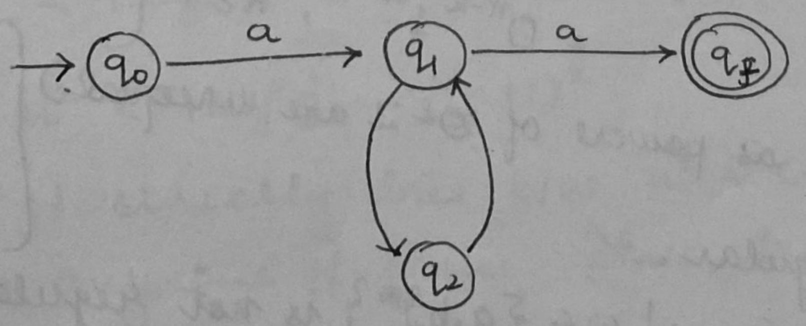
Which is a contradiction to Pumping lemma & hence L is not regular.

Ques Is $L = \{a^{2n} \mid n \geq 1\}$ regular?

If regular then, we can represent L as a r.e. or create a finite automata accepting L.

r.e. for L = $a(aa)^*a$

finite automaton



Scanned with CamScanner

CLOSURE PROPERTIES OF REGULAR SETS

We discuss the closure properties of regular sets under the following operations :-

(i) set union (ii) concatenation (iii) closure (iteration)
(iv) transpose (Reverse) (v) intersection (vi) complementation

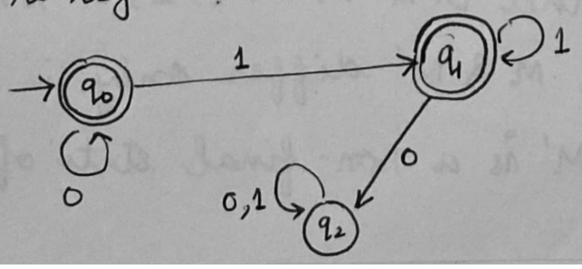
Theorem : If L is regular then L^T is also regular.

Proof :- As L is a regular language, we can construct a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $T(M) = L$.

We construct a transition system M' by starting with the state diagram of M & reversing the directions of the directed edges. The set of initial states of M' is defined as the final state set F, and q_0 is defined as the (only) final state of M', i.e. $M' = (Q, \Sigma, \delta', F, \{q_0\})$.

If $w \in T(M)$, we have a path from q_0, to some final state in F with path value w. By 'reversing the edges' we get a path in M' from some final state in F to q_0. Its path value is w^T. So $w^T \in T(M')$. In a similar way, we can see that if $w_1 \in T(M')$, then $w_1^T \in T(M)$. Thus, from the state diagram it is easy to see that $T(M') = T(M)^T$. We can prove rigorously that $w \in T(M)$ iff $w^T \in T(M')$ by induction on $|w|$. So, $T(M') = T(M)^T$. Hence, $T(M')$ is regular, i.e. $T(M)^T$ is regular.

Ques :- Consider the FA M given below. What is $T(M)$? Show that $T(M)^T$ is regular.



Scanned with CamScanner

We can construct T(M) by inspection.

As arrows do not come into q_0, the paths from q_0 to itself self-loops repeated any no. of times. The corresponding path-values are 0^i, $i \geq 1$. As no arrow comes from q_2 to q_0 or q_1 (final states), the paths from q_0 to q_1 are of the form:

$$q_0 \ldots \rightarrow q_0 \ldots q_1 \ldots \rightarrow q_1$$

The corresponding path values are $0^i 1^j$ where $i \geq 0$, $j \geq 1$.

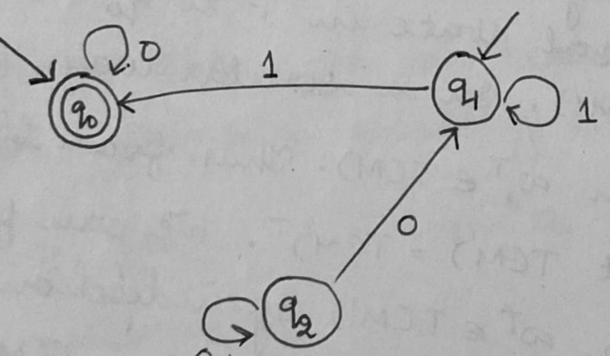
As initial state q_0 is also a final state $\therefore \Lambda \in T(M)$. Thus,

$$T(M) = \{ 0^i 1^j \mid i, j \geq 0 \}$$

Hence

$$T(M)^T = \{ 1^j 0^i \mid i, j \geq 0 \}$$

The transition system M' is $(Q, \Sigma, \delta', \{q_0, q_1\}, \{q_0\})$ where $Q = \{q_0, q_1, q_2\}$ & the δ is given by transition-diagram below :-



$\therefore \exists$ a finite automata corr. to $T(M)^T$ $\therefore$ it is regular.

<u>Theorem</u>: If L is regular set over Σ, then $\Sigma^* - L$ (complement) is also regular over Σ.

<u>Proof</u>:- As L is regular, we can construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L, i.e. $L = T(M)$.

We can construct another DFA $M' = (Q, \Sigma, \delta, q_0, F')$ by defining $F' = Q \setminus F$, i.e., M & M′ differ only in their final states. A final state of M′ is a non-final state of M & vice versa.

$w \in T(M')$ iff $\delta(q_0, w) \in F' = Q \setminus F$

i.e. iff $w \notin L$.

This proves $T(M') = \Sigma^* - L$

Theorem:- If X & Y are regular sets over Σ, then $X \cap Y$ is also regular over Σ.

Proof:- By De Morgan's Law for sets, $X \cap Y = \Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$.

$(X \cap Y = (X^c \cup Y^c)^c)$

By previous theorem, $\Sigma^* - X$ and $\Sigma^* - Y$ are regular. So, $(\Sigma^* - X) \cup (\Sigma^* - Y)$ is also regular & once again its complement $\Sigma^* - ((\Sigma^* - X) \cup (\Sigma^* - Y))$ is regular i.e. $X \cap Y$ is regular.

corresponding to regular languages create M & M' s.t.

$T(M) = X$ & $T(M') = Y$

We know $\exists$ a finite automata $T(M) \cup M'$ s.t.

$T(M \cup M') = T(M) \cup T(M')$

$= X \cup Y$

$\Rightarrow X \cup Y$ is Regular