FA’s with regular expressions

Week-3

Finite Automaton

Finite automata are computing devices that accept/recognize regular languages and are used to model operations of many systems we find in practice.

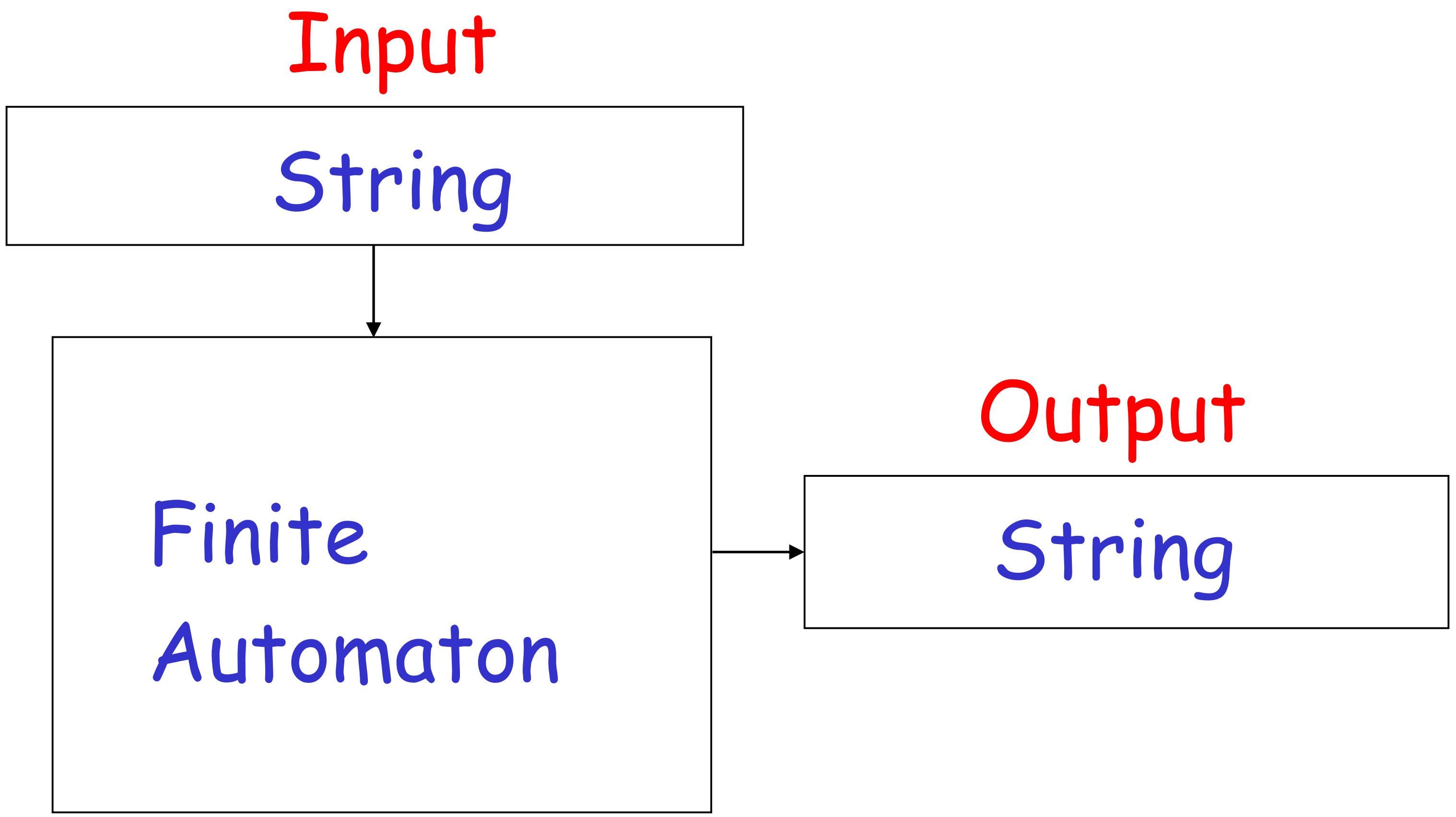
In the word 'Finite Automata', Finite means the no. of letters in the input alphabet & the no. of states are both finite and automata means automatic.

Finite Automaton
(also called acceptor/recognizer)

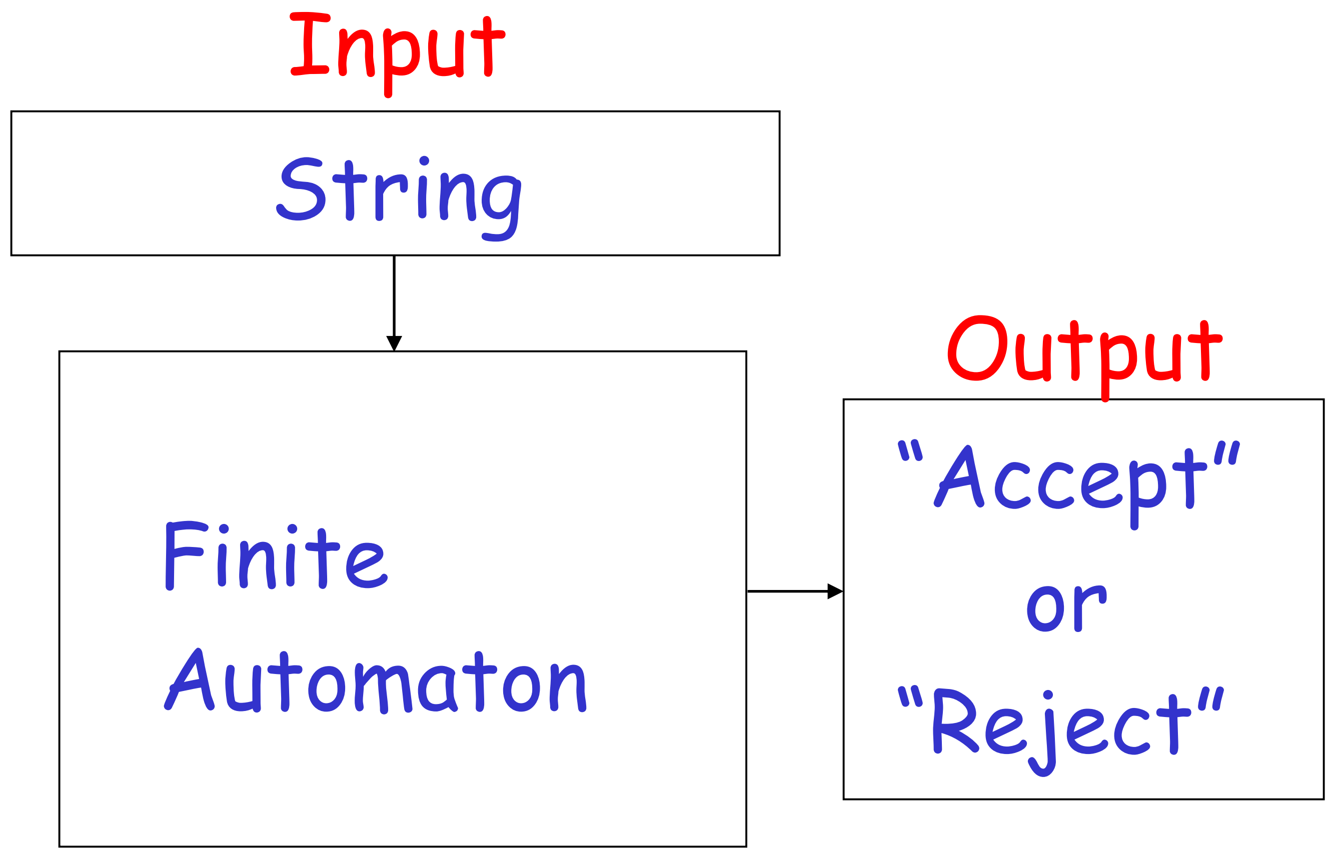
A Finite Automaton (FA) is a collection of three things

1. Finite set of input letter Σ from which input strings are formed.
2. Finite number of states, having one initial and some (maybe none) final states.
3. Finite set of transitions i.e. for each state and for each input letter there is a transition showing how to move from one state to another.

Finite Automaton



Finite Accepter



FA - Examples

Example:

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

States: x, y, z where x is an initial state and z is final state.

Transitions:

At state x reading a, go to state z

At state x reading b, go to state y

At state y reading a, b go to state y

At state z reading a, b go to state z

Example:

State Transition Table

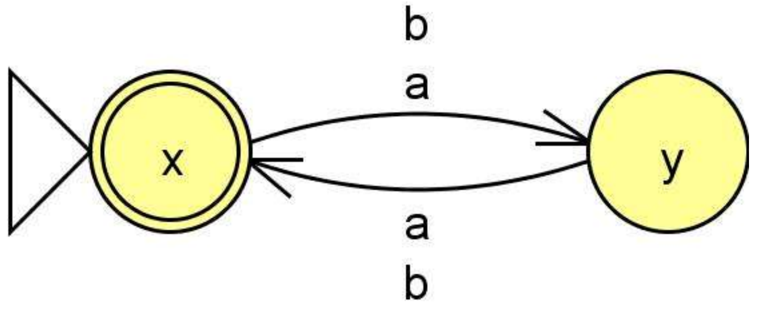
Old States	New States	
	Reading a	Reading b
x -	z	y
y	y	y
z +	z	z

The FA is (Transition Diagram)

RE : ?

a(a+b)*

FA



State Transition Table ? Over same Σ

Regular Expression?

Formalities

Deterministic Finite Accepter (DFA) OR
Deterministic Finite Automaton

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

Q : set of states

Σ : input alphabet

δ : transition function

q_0 : initial state $q_0 \in Q$

F : set of final states $F \subseteq Q$

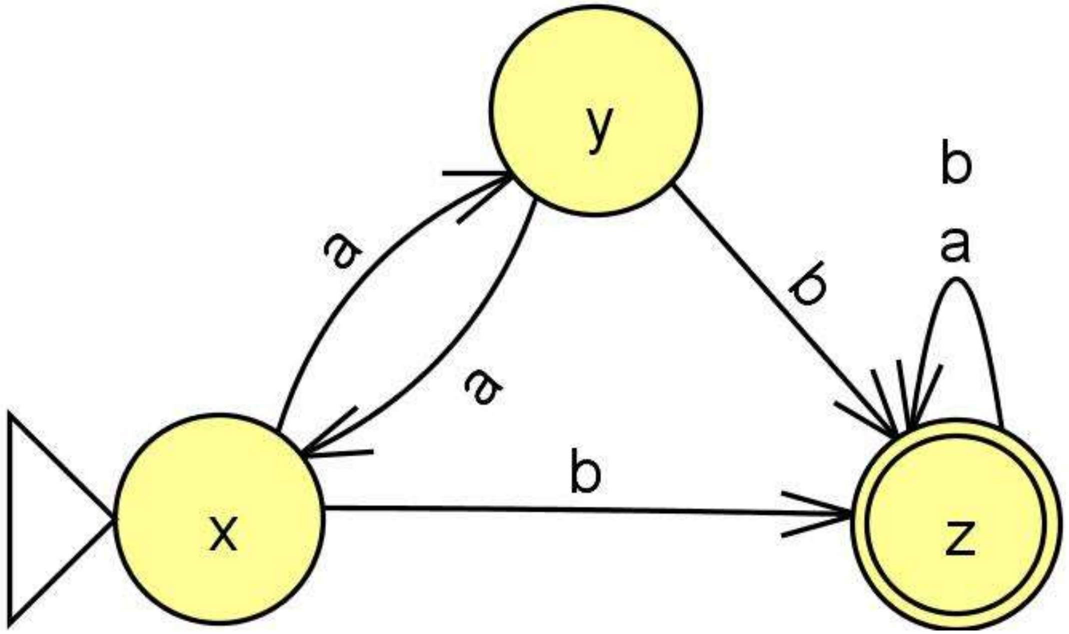
Write R.E: It accepts at least one b somewhere in the strings

RE :

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

State Transition Table

Old States	New States	
	Reading a	Reading b
x -	y	z
y	x	z
z +	z	z



Ex: The FA that accepts the language with RE = (a+b)*(aa+bb)(a+b)*

Ex: The FA that accepts the language with all the strings having at least one triple letter.

Ex: The FA that accepts the language of all the words that

(i) Have b as the 2nd letter

(ii) Have b as the 3rd letter

Note: there are FAs that accept no language. These are of two types

(i) FA's that have no final state, such as

(ii) FA's in which final states cannot be reached from the start state i.e. a disconnected graph or graph like this

Ex. The FA that accepts the language with all the strings that do not end with a.

Sol. Do not ends with a = Ends with what?

OR

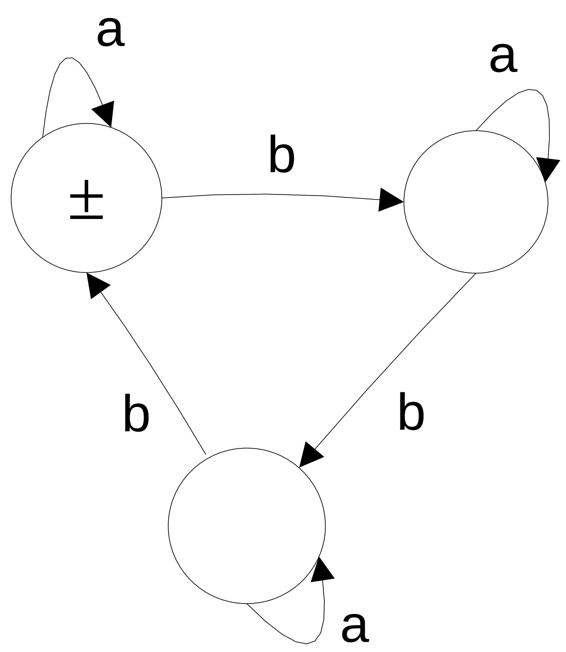
take the complement of “ends with a”.

Ex-p-68. The language of all the words that have different first and last letter.

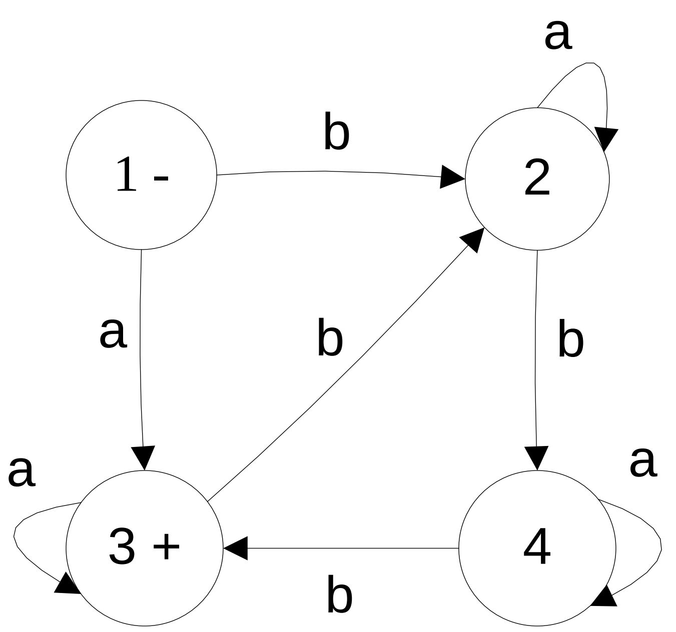
Ex-p-65. The FA that accepts only the words *baa* and *ab*.

Ex. The FA that accepts only the words *baa*, *ba* and *ab*.

Ex-p-65: What language the following FA accepts i.e. find RE.



Ex-p-65: What language the following FA accepts i.e. find RE.



Ex: FA for all strings having less than 3 b's.

Ex: FA for EVEN-EVEN language

RE = (aa+bb+(ab+ba)(aa+bb)*(ab+ba))*

Q.4. (i) Build an FA with three states that accepts all strings.

Q.6. FA that accepts only those word that do no end with ba.

Sol: Do no end with ba = Ends with what?

OR Find FA that ends with ba then take complement.

Q.7: (i) Begin or end with double letter.

(ii) Begin and end with double letter.

Sol: Find FA that begin with double letter

Find FA that end with double letter

Now combine the above two FAs. To find the solution of both the parts.

Q.8: Even number of substring ab

Q.9: (i) FA that accepts the language of all the strings that have both the letter a and the letter b in them, but not necessarily in that order.

(ii) All the strings that have onl a's or only b's in them

Important Notes

1. If λ is present in the language we can make the FA accept λ by making the start state as the final state.
2. From every state there will be as many outgoing arrows $\rightarrow$ as there are letters in the alphabet.
3. If any RE contains (a+b)* in the beginning we shall first construct the FA by discarding that (a+b)*, the initially discarded (a+b)* will be taken into account by missing arrows & there will be no dead end state.
4. When all the words of the language (finite) are accepted by the FA then all the missing edges will go to dead end state.

Finite Automaton: Home tasks

Create transition table and FAs for following languages over alphabet $\Sigma = \{a,b\}$

1. Language with no words
2. Language of all words
3. Language of even / odd length words
4. Language of all words starting with a
5. Language of all words not starting with a
6. Language of all words ending with a
7. Language of all words not ending with a
8. Language of all words starting and ending with same letter

Finite Automaton: Exercises

Create transition table and FAs for following languages over alphabet $\Sigma = \{a,b\}$

1. Language of all words starting and ending with different letters
2. Language of all words containing substring abb
3. Language of all words containing substring abb or baa
4. Language of all words not containing substring abb
5. EVEN-EVEN / ODD-ODD / EVEN-ODD / ODD-EVEN
6. Language of all words with count of 'a' as multiple of three
7. Language of all words with count of 'a' as not multiple of three
8. Language of all words with a as 2nd letter

Finite Automaton: Exercises

Create transition table and FAs for following languages over alphabet $\Sigma = \{a,b\}$

1. Language of all words with 'a' as 2nd last letter
2. Language of all words where 'a' can only appear on even positions; if at all
3. Language of all words where 'a' never follows 'b'
4. Language of all words where 'a' never precedes 'b'
5. a^*b^*

- ✓Creating FA to accept a Finite Language
- ✓Creating FA for **Union** of Two Languages
- ✓Creating FA for **Intersection** of Two Languages
- ✓Creating FA for **Negation** of a Language
- ✓Creating FA for **Concatenation** of Two Languages
- ✓Creating FA for **Kleene's Star** of a Language

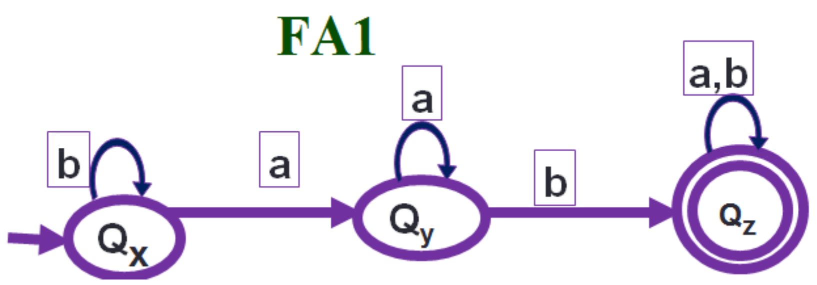
Union Theorem

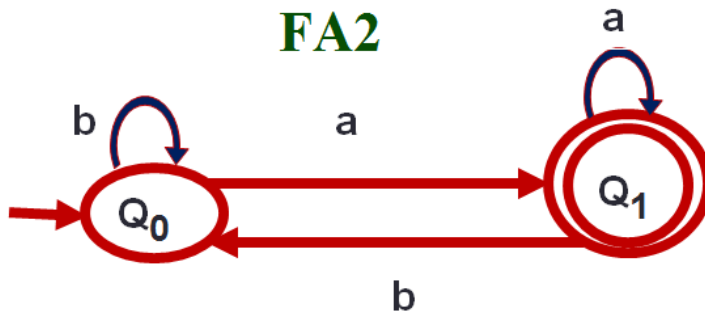
Given two languages, L_1 and L_2, define the union of L_1 and L_2 as
$L_1 \cup L_2 = \{ w \mid w \in L_1 \text{ OR } w \in L_2 \}$

Theorem: The union of two regular languages is also a regular language

Union (of languages) Theorem: (Demonstration)

Let L_1 and L_2 be two languages over such that $\Sigma = \{a,b\}$
L_1 = "Language of all words having substring ab"
L_2 = "Language of all words ending on a"
Corresponding FA's are as follows



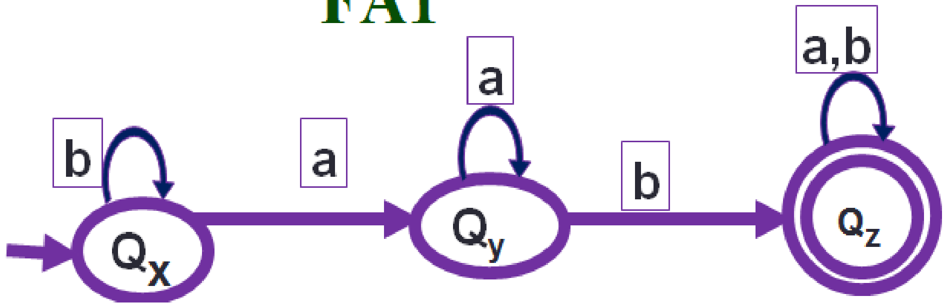


Union of above languages is **a language that contains all words that are either ending on a or having substring ab**

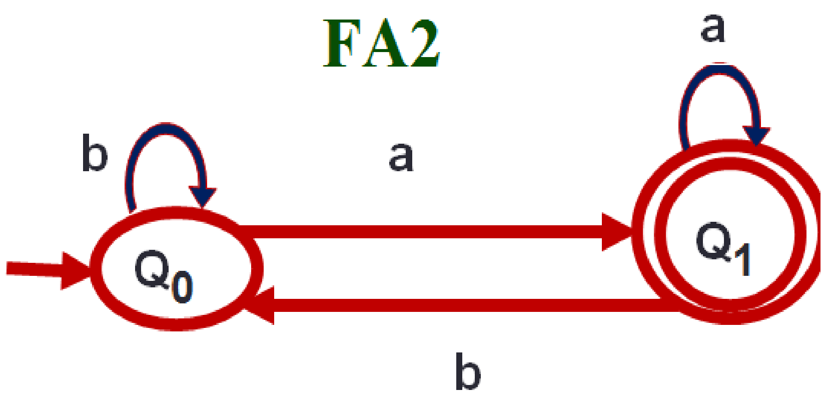
Creating FA for Union of FA_1 & FA_2

Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x OR Q_0)	Q_{z1} (Q_y OR Q_1)	Q_{z0} (Q_x OR Q_0)
Q_{z1} (Q_y OR Q_1) FINAL STATE	Q_{z1} (Q_y OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)

FA1

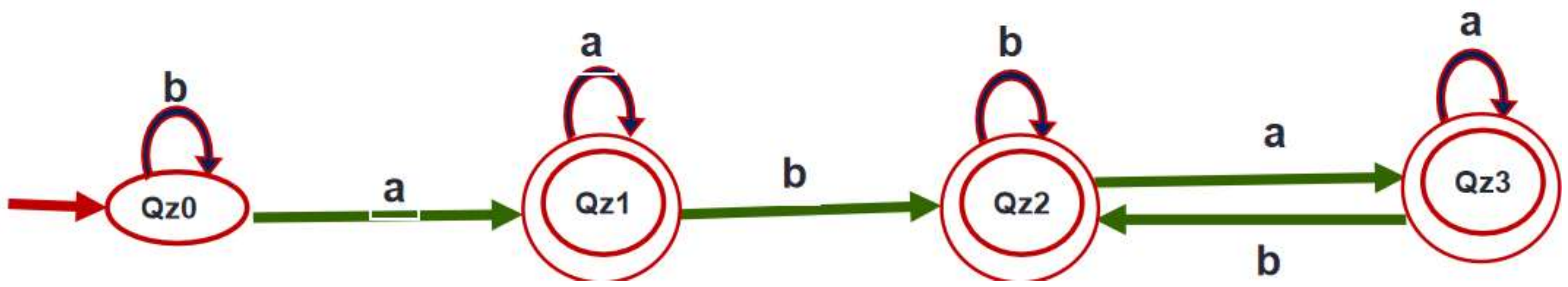


FA2



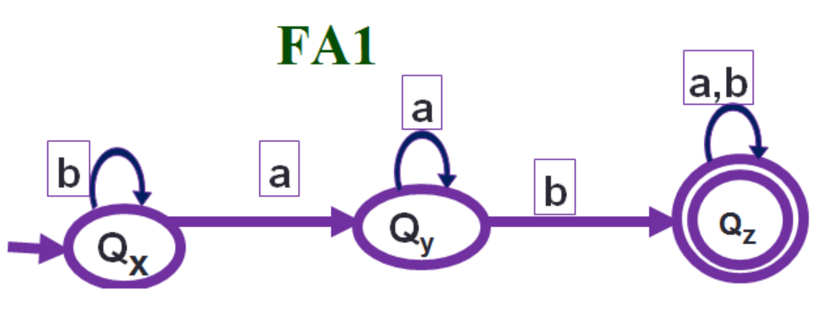
Creating FA for Union of FA_1 & FA_2 from Transition Table

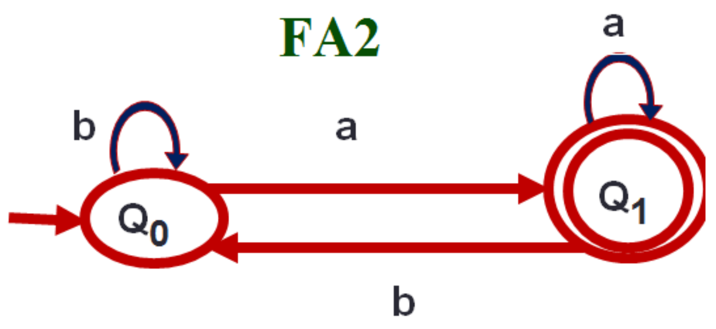
Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x OR Q_0)	Q_{z1} (Q_y OR Q_1)	Q_{z0} (Q_x OR Q_0)
Q_{z1} (Q_y OR Q_1) FINAL STATE	Q_{z1} (Q_y OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)



Intersection (of languages) Theorem: (Demonstration)

Let L_1 and L_2 be two languages over such that $\Sigma = \{a,b\}$
L_1 = "Language of all words having substring ab"
L_2 = "Language of all words ending on a"
Corresponding FA's are as follows

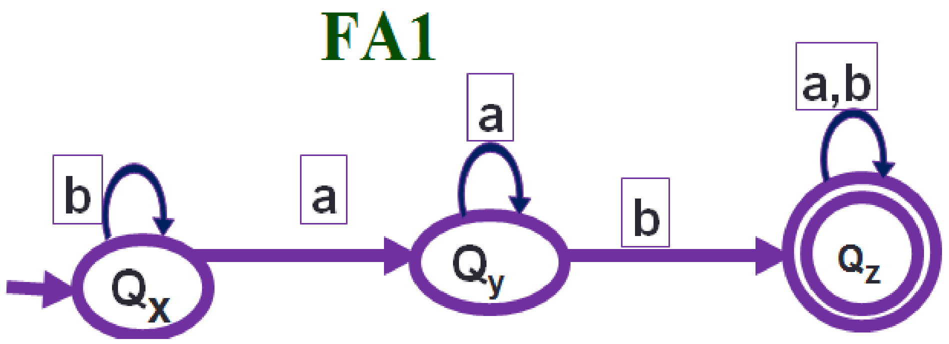


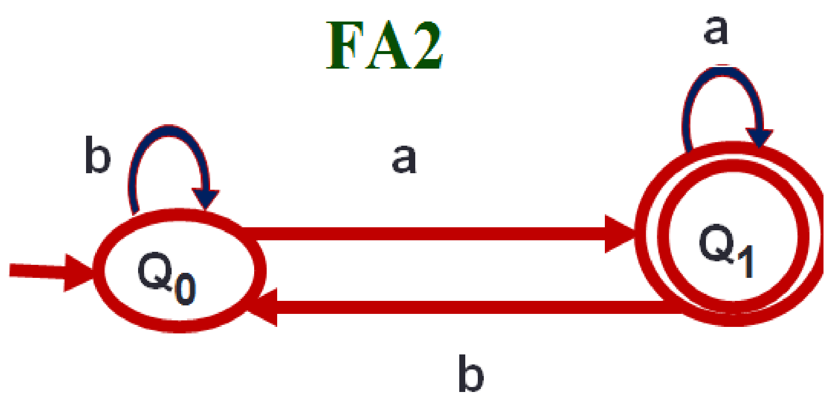


Intersection of above languages is a language that contains all words that end on a and have substring ab

Creating FA for Intersection of FA_1 & FA_2

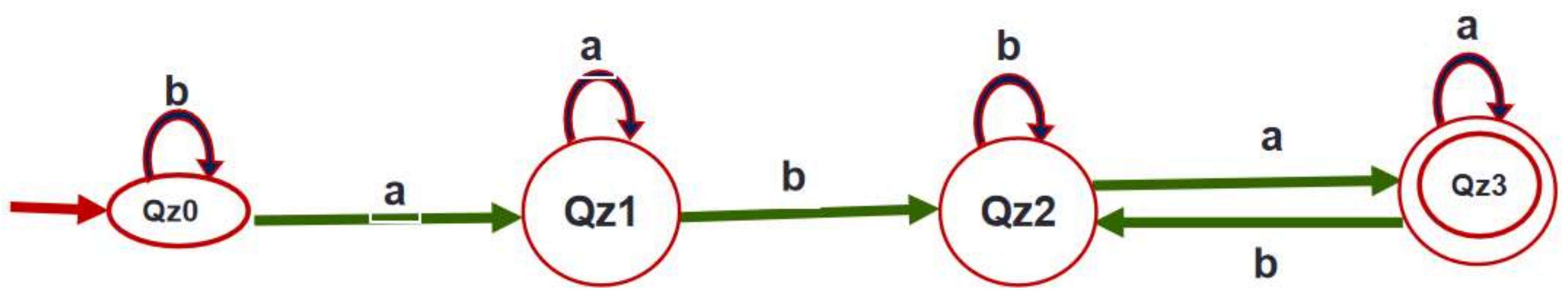
Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x OR Q_0)	Q_{z1} (Q_y OR Q_1)	Q_{z0} (Q_x OR Q_0)
Q_{z1} (Q_y OR Q_1)	Q_{z1} (Q_y OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0)	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)





Creating FA for Intersection of FA_1 & FA_2 from Transition Table

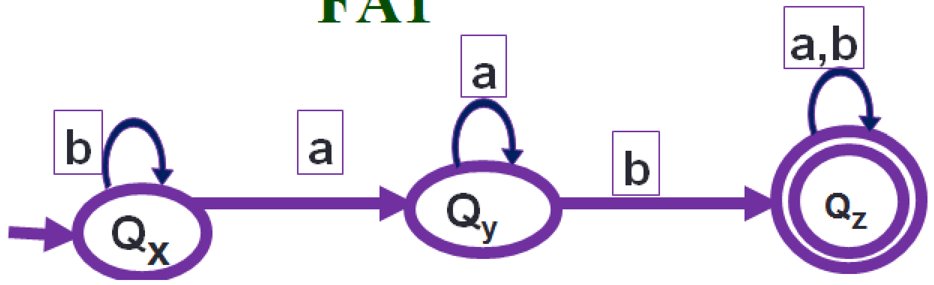
Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x OR Q_0)	Q_{z1} (Q_y OR Q_1)	Q_{z0} (Q_x OR Q_0)
Q_{z1} (Q_y OR Q_1)	Q_{z1} (Q_y OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0)	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_1)	Q_{z2} (Q_z OR Q_0)



Negation (of language) Theorem: (Demonstration)

Let L_1 be a language over such that $\Sigma = \{a,b\}$
L_1 = "Language of all words having substring ab"
Corresponding FA is as follows

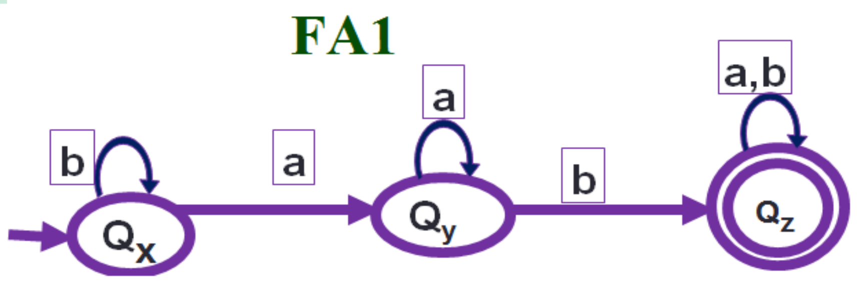
FA1

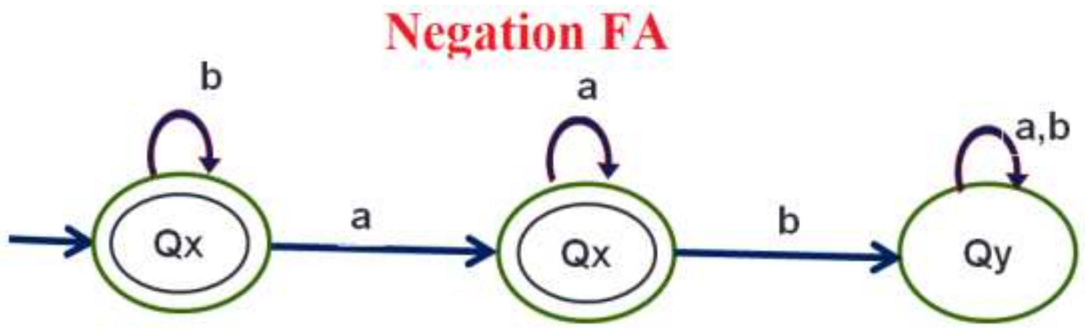


Intersection of above language is **a language that contains all words not having substring ab**

Creating FA for Negation of FA_1

Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x) FINAL STATE	Q_{z1} (Q_y)	Q_{z0} (Q_x)
Q_{z1} (Q_y) FINAL STATE	Q_{z1} (Q_y)	Q_{z2} (Q_z)
Q_{z2} (Q_z)	Q_{z2} (Q_z)	Q_{z2} (Q_z)

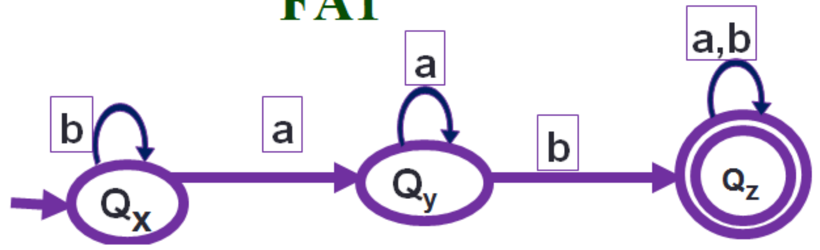




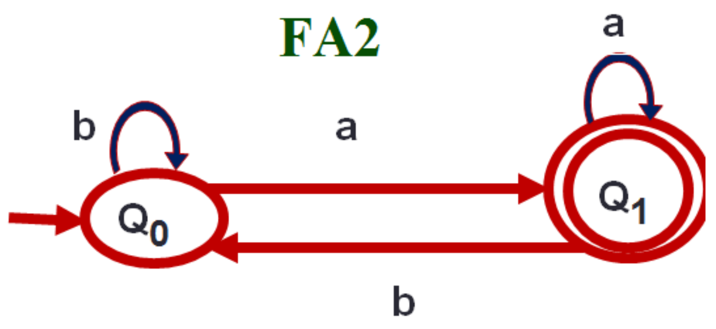
Concatenation(of languages) Theorem: (Demonstration)

Let L_1 and L_2 be two languages over such that $\Sigma = \{a,b\}$
L_1 = “Language of all words having substring ab”
L_2 = “Language of all words ending on a”
Corresponding FA’s are as follows

FA1



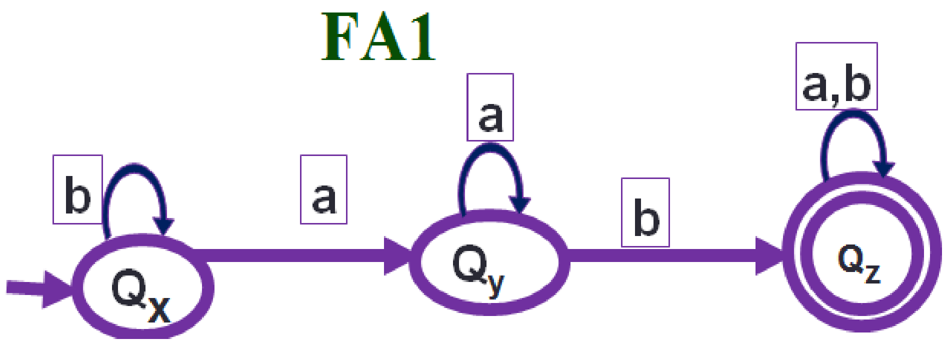
FA2

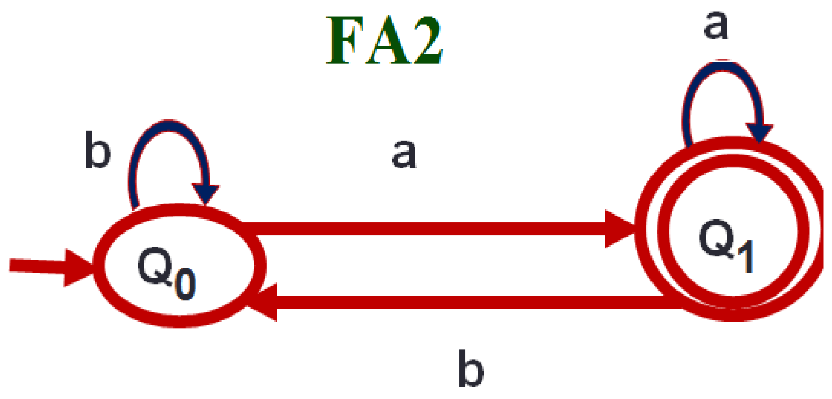


Concatenation of above languages is a language that contains all words contain substring ab and also end on a

Creating FA for Concatenation of FA_1 & FA_2

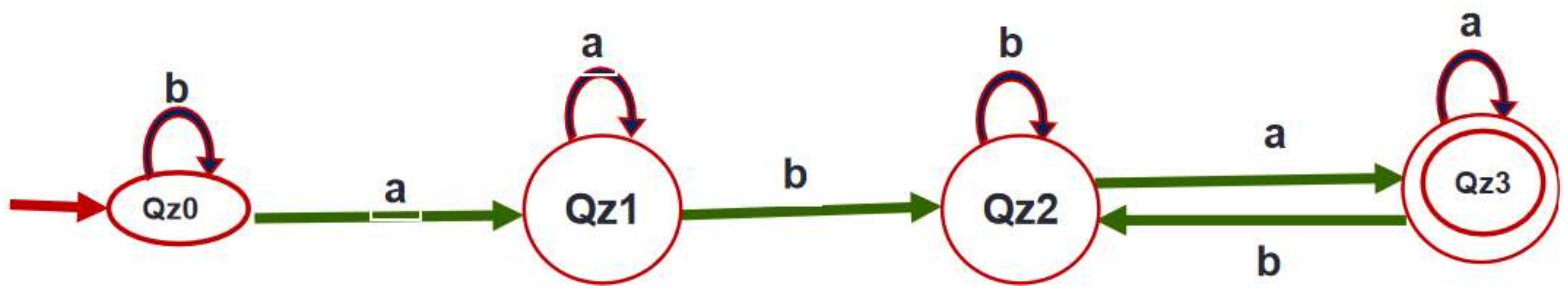
Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x)	Q_{z1} (Q_y)	Q_{z0} (Q_x)
Q_{z1} (Q_y)	Q_{z1} (Q_y)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0)	Q_{z3} (Q_z OR Q_0 OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_0 OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_0 OR Q_1)	Q_{z2} (Q_z OR Q_0)





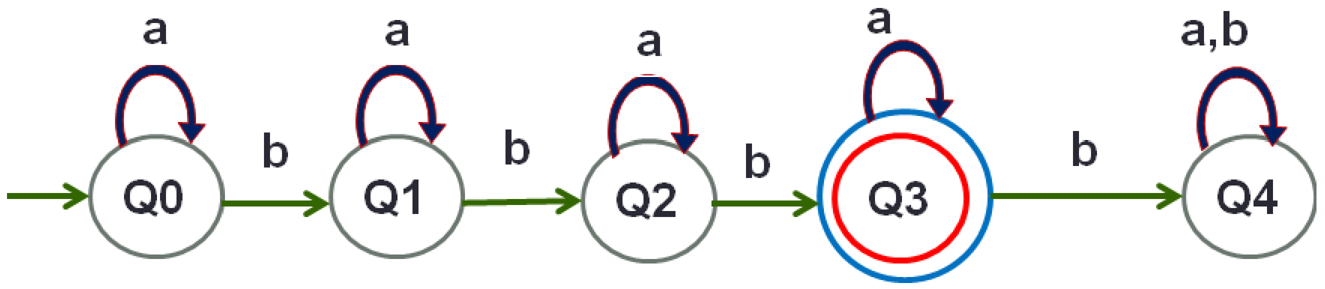
Creating FA for Concatenation of FA_1 & FA_2 from Transition Table

Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x)	Q_{z1} (Q_y)	Q_{z0} (Q_x)
Q_{z1} (Q_y)	Q_{z1} (Q_y)	Q_{z2} (Q_z OR Q_0)
Q_{z2} (Q_z OR Q_0)	Q_{z3} (Q_z OR Q_0 OR Q_1)	Q_{z2} (Q_z OR Q_0)
Q_{z3} (Q_z OR Q_0 OR Q_1) FINAL STATE	Q_{z3} (Q_z OR Q_0 OR Q_1)	Q_{z2} (Q_z OR Q_0)



Kleene's Star Theorem: (Demonstration)

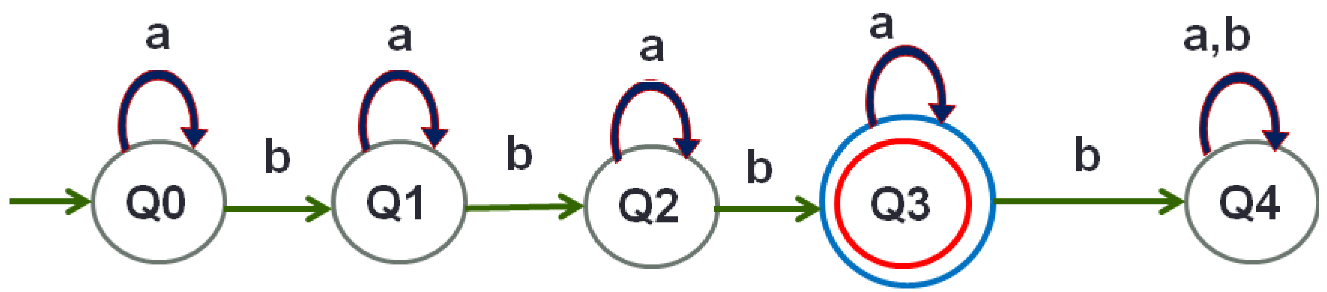
Let L_1 be a language over such that $\Sigma = \{a,b\}$
L_1 = "Language of all words exactly three b"
Corresponding FA is as follows



Kleene Star of above language is **a language that contains all words containing any multiple of three b i.e. 0 b, 3 b, 6 b, 9 b etc**

Creating FA for Concatenation of FA_1 & FA_2

Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x) FINAL STATE	Q_{z0} (Q_0)	Q_{z1} (Q_1)
Q_{z1} (Q_1)	Q_{z1} (Q_1)	Q_{z2} (Q_2)
Q_{z2} (Q_2)	Q_{z2} (Q_2)	Q_{z2} (Q_3 OR Q_0)
Q_{z3} (Q_3 OR Q_0) FINAL STATE	Q_{z3} (Q_3 OR Q_0)	Q_{z4} (Q_4 OR Q_1)
Q_{z4} (Q_4 OR Q_1)	Q_{z4} (Q_4 OR Q_1)	Q_{z5} (Q_4 OR Q_2)
Q_{z5} (Q_4 OR Q_2)	Q_{z5} (Q_4 OR Q_2)	Q_{z6} (Q_4 OR Q_3 OR Q_0)
Q_{z6} (Q_4 OR Q_3 OR Q_0) FINAL STATE	Q_{z6} (Q_4 OR Q_3 OR Q_0)	Q_{z4} (Q_4 OR Q_1)



Creating FA for Kleene Starof FA$_1$ from Transition Table

Current State	New State on Input = a	New State on Input = b
Q_{z0} (Q_x) FINAL STATE	Q_{z0} (Q_0)	Q_{z1} (Q_1)
Q_{z1} (Q_1)	Q_{z1} (Q_1)	Q_{z2} (Q_2)
Q_{z2} (Q_2)	Q_{z2} (Q_2)	Q_{z2} (Q_3 OR Q_0)
Q_{z3} (Q_3 OR Q_0) FINAL STATE	Q_{z3} (Q_3 OR Q_0)	Q_{z4} (Q_4 OR Q_1)
Q_{z4} (Q_4 OR Q_1)	Q_{z4} (Q_4 OR Q_1)	Q_{z5} (Q_4 OR Q_2)
Q_{z5} (Q_4 OR Q_2)	Q_{z5} (Q_4 OR Q_2)	Q_{z6} (Q_4 OR Q_3 OR Q_0)
Q_{z6} (Q_4 OR Q_3 OR Q_0) FINAL STATE	Q_{z6} (Q_4 OR Q_3 OR Q_0)	Q_{z4} (Q_4 OR Q_1)

Please create FA by using above Transition Table