

**CISC 603-51- A-2021/SUMMER - THEORY OF COMPUTATION**

**Assignment-2**

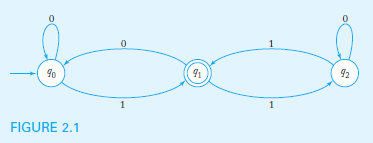
**Finite Automata & Equivalence and Reduction**

**By,**

**Satheesh Swaminathan**

**Student ID:** **266582**

2.1 #1



q0 is the input state, since has an arrow coming in showing this is input.

q1 is the output state, state where the DFA (Deterministic Finite Automaton) would exit denoted by internal circle.

**Test String 0001**

q0(initial state) 🡪 q0(input=**0**) 🡪 q0(input=**0**)🡪 q0(input=**0**)🡪 q0(input=**1**)🡪 q1(exit or final state)

This Automaton is in a final state after the last symbol is processed (i.e., 1) hence this string ‘0001’ is **accepted.**

**Test String 01101**

q0(initial state) 🡪 q0(input=**0**) 🡪 q0(input=**1**)🡪 q1(input=**1**)🡪 q2(input=**0**)🡪 q2(input=**1**) 🡪 q1(exit or final state)

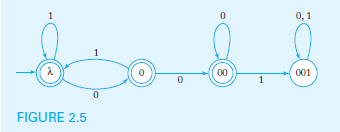
This Automaton is in a final state after the last symbol is processed (i.e., 1) hence this string **‘01101’** is **accepted**.

**Test String 00001101**

q0(initial state) 🡪 q0(input=**0**) 🡪 q0(input=**0**)🡪 q0(input=**0**) 🡪 q0(input=**0**)🡪 q0(input=**1**)🡪 q1(input=**1**)🡪 q2(input=**0**)🡪 q1(input=**1**)🡪 q2(exit or final state)

This Automaton is in a final state after the last symbol is processed (i.e., 1) hence this string **‘00001101’** is **accepted**.

2.1 #2 Delta notation (symbol: δ) translation



Symbol – δ denotes Delta.

δ -Transitions

At the first node

**δ (λ,0) 🡪 0** (Explanation: From the initial state **‘λ’** an input of **‘0’** will give a final state of **‘0’**)

**δ (λ,1) 🡪 1** (Explanation: From the initial state **‘λ’** an input of **‘1’** will give a final state of **‘1’**)

From the second node

**δ (0,0) 🡪 00** (Explanation: From this state ‘0’ an input of ‘0’ will give a final state of ‘00’)

**δ (0,1) 🡪 λ** (Explanation: From this state ‘0’ an input of ‘1’ will give a final state of ‘λ’)

From the third node

**δ (00,0) 🡪 00** (Explanation: From this state ‘00’ an input of ‘0’ will give a final state of ‘00’)

**δ (00,1) 🡪 001** (Explanation: From this state ‘00’ an input of ‘1’ will give a final state of ‘001’)

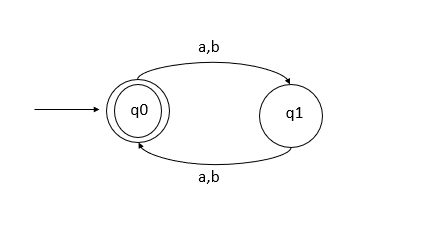
From the fourth node

**δ (001,0) 🡪 001** (Explanation: From this state ‘001’ an input of ‘0’ will give a final state of ‘001’)

**δ (001,1) 🡪 001** (Explanation: From this state ‘001’ an input of ‘1’ will give a final state of ‘001’)

2.1

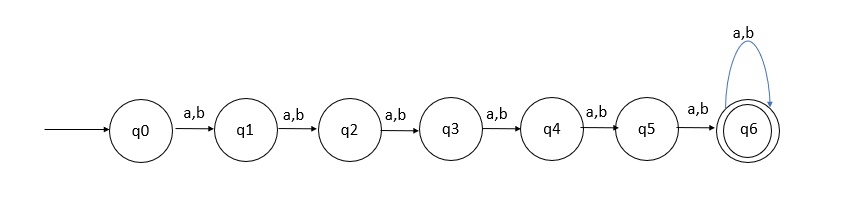
#3 a) All strings of even length.



Explanation:

1. This DFA has an initial state q0 with an input arrow indicated and the final exit state as q0 as well.
2. From the initial state q0 an input of either a or b will take this to the next state q1 at which point the input string will be of odd length
3. From the next state q1, any input will make the string even hence this could take us to the final state i.e., q0.
4. Hence, this DFA will accept any string of length **even**.

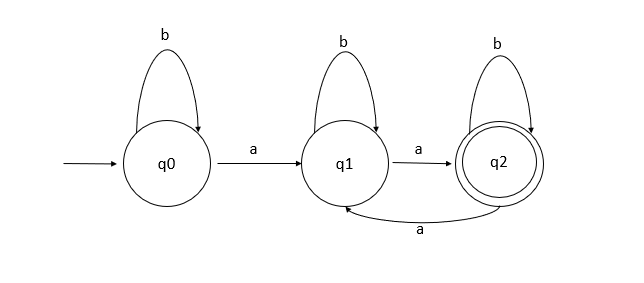
d) All strings of length greater than 5.



Explanation:

1. This DFA has an initial state q0 and the final or exit state q6.
2. From the states q0 to q5 any input either a or b will propagate through the DFA to the next state since the input string has not reached a minimum length of 5
3. When the input string reaches a minimum length of 5, we are at the stage q6, beyond which any input given to this state, the length of the test string will always be greater than 5,
4. Hence, we can conclude that the DFA has reached final or exit state.

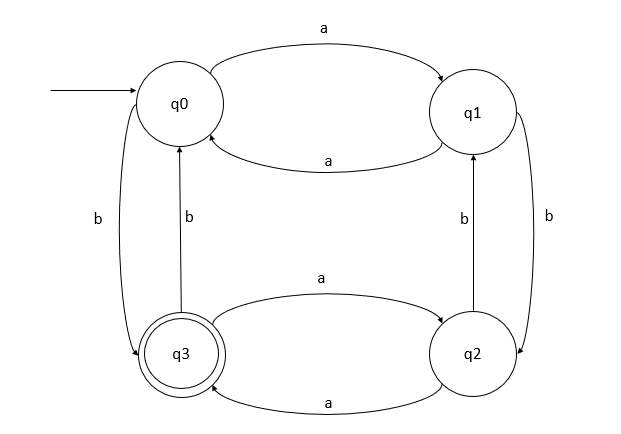
c) All strings with an even number of a’s.



Explanation:

1. This DFA has an initial state q0, an input of a will make odd number of a’s, next state is q1.
2. The state q1 input of a will make the number of a’s even, therefore the next state q2 is final or exit state.
3. The value of ‘b’ has no affect on the states of this DFA, hence any state which gets an input of ‘b’ will remain at the same state.

d) All strings with an even number of a’s and an odd number of b’s.

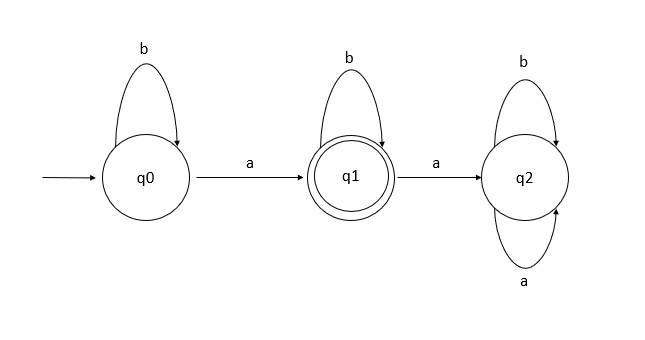


Explanation:

1. There are four states in this DFA, the initial state is denoted with an input arrow i.e., q0 (even A and even B)
2. q0(even a, a) 🡪 q1(odd a) && q0(even a, a) 🡪 q3(odd b) similar we can address what happens to the states when the input is ‘b’, which makes the total states as 4 including the initial state.
3. From the above DFA figure we can conclude that
   1. q0 (Even ‘**a’** and Even ‘**b’**) 🡨 Initial state
   2. q1(Odd ‘**a’** and Even ‘**b’**)
   3. q2(Odd ‘**a’** and Odd ‘**b’**)
   4. q3(Even ‘**a’** and Odd ‘**b’**) 🡨 Final or exit state.

2.1

#4 a) All strings with exactly one a.

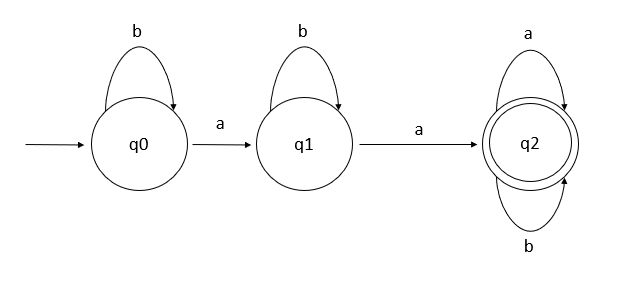


Explanation:

1. This DFA has 3 states, the initial state being q0 with input arrow, this state takes the input either ‘a’ or ‘b’. The input of ‘a’ meets the necessary condition to satisfy the DFA (exactly one ‘a’)
2. The value of ‘b’ does not have any effect on the states of this DFA, hence input of ‘b’ on any state will keep it on the same state.
3. Another input of ‘a’ at state q1 will make the number of ‘a’ more than one which is different from state q0 hence new state is marked as q2.
4. Any input on q2 doesn’t really matter.

2.1

#4 b) All strings with at least two a’s.

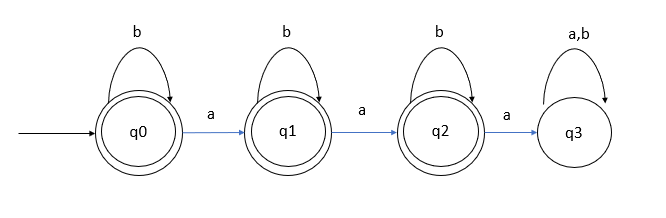


Explanation:

1. the initial state q0 takes input a, next state q1 will have only one ‘a’.
2. state q1 takes another input a which satisfies the condition (At least two ‘a’) in state q2 which is our exit or final state.
3. any input from the state q2 doesn’t really matter will loop back to the same state q2 and the value of ‘b’ has no effect on the DFA.

2.1

#4 c) All strings with no more than two a’s.

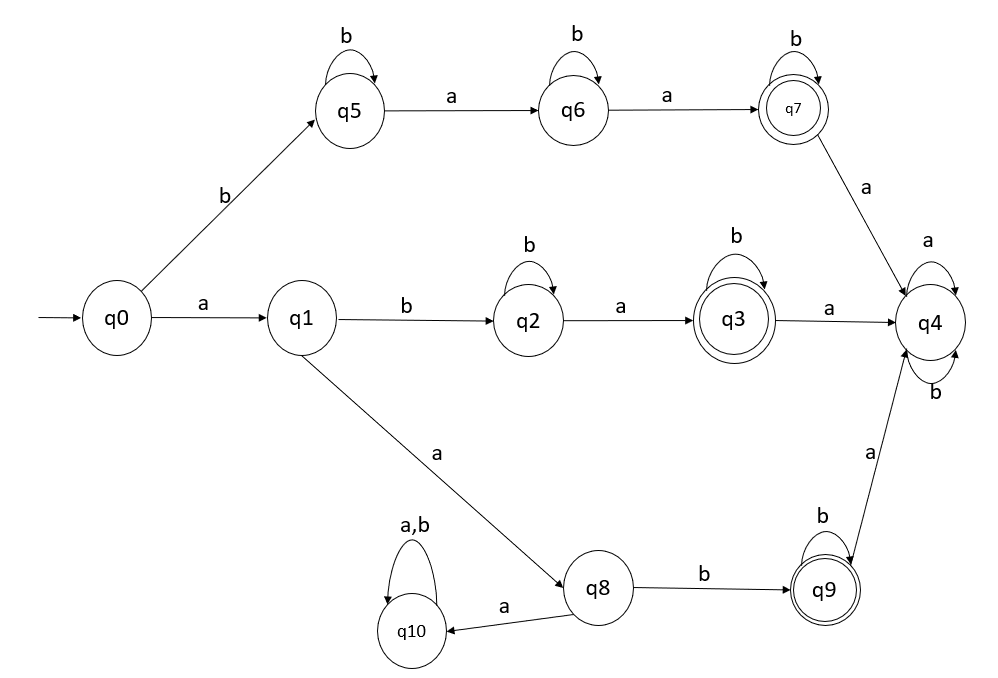


Explanation:

1. The states q0, q1, q2 and q3 all satisfies the condition of not more than two a’s.
2. At state q2 an input of ‘a’ makes it more than two a’s

2.1

#4 d) All strings with at least one b and exactly two a’s.



Explanation:

This should satisfy some of the following cases:

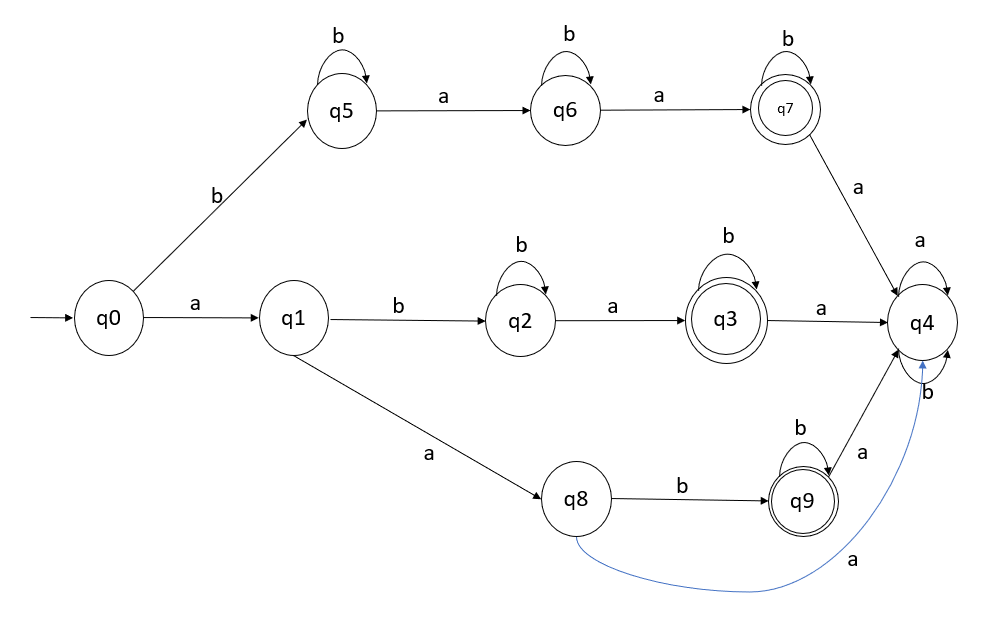
1. ‘aab’ 🡪 this will reach state q9 which is one of the final states through q0 🡪 q1🡪q8🡪 q9.
2. ‘aba’ 🡪 This will reach state q3 second final state through q0 🡪 q1🡪q2🡪 q3.
3. ‘baa’ 🡪 this will reach state q7 the third final state through q0 🡪 q5🡪q6🡪 q7.
4. ‘bbaa’ 🡪 for instance will take the route:

q0(input=b)🡪q5(input=b)🡪 q5(input=a)🡪 q6(input=a) 🡪q7(exit or final state)

1. ‘aaa’ 🡪 q0(input=a)🡪q1(input=a)🡪 q8(input=a, b)🡪 q10

2.1

#4 e) all the strings with exactly two a’s and more than three b’s.



‘aabbbb’ 🡪 q0(a) 🡪q1(a) 🡪q8(b) 🡪q9(b) 🡪 q9(b) 🡪 q9(b)🡪 q9 (final state)

‘bbbbaa’ 🡪 q0(b) 🡪q5(b) 🡪q5(b) 🡪q5(b) 🡪q5(a) 🡪q6(a) 🡪q7(final state)

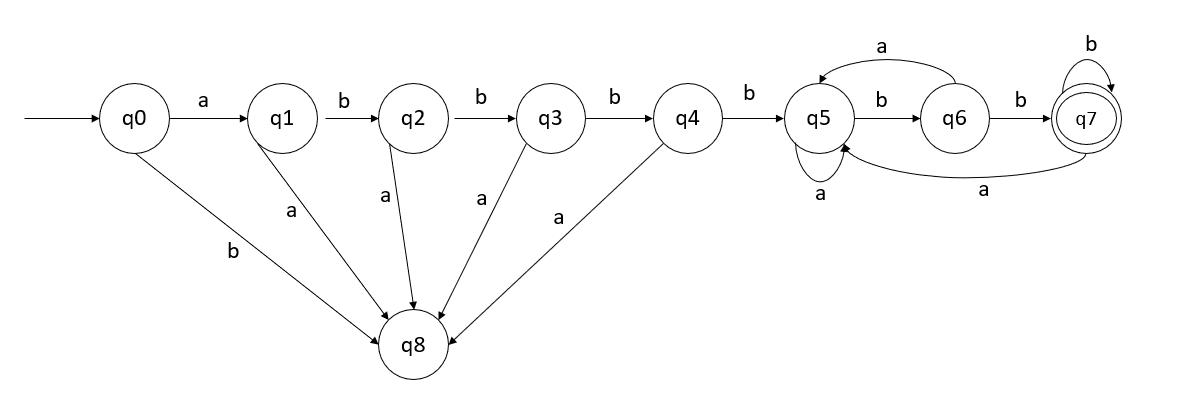
‘bababb’ 🡪 q0(b) 🡪q5(a) 🡪q6(b) 🡪 q6(a) 🡪q7(b) 🡪 q7(b)q7(final state)

‘bbabab’ 🡪 q0(b) 🡪q5(b) 🡪q5(a) 🡪q6(b) 🡪q6(a) 🡪q7(b) 🡪q7(final state)

2.1

#5 a)





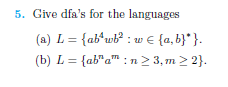
Explanation:

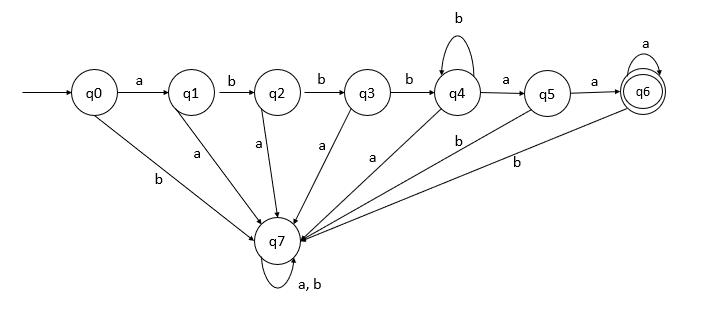
1. ab4 🡪 ‘abbbb’ this will take the route q0(a) 🡪q1(b)🡪 q2(b)🡪 q3(b) 🡪q4(b) 🡪q5.
2. any other inputs for states q0(b), q1(a), q2(a), q3(a), q4(a) will go to dead state.
3. Wb2 denotes that two ‘b’ from the state q5 will take us to the final or exit state q7 i.e.,

q5(b) 🡪q6(b) 🡪q7

1. ‘abbbbbb’ 🡪 will reach the final state q7.

#5 b)



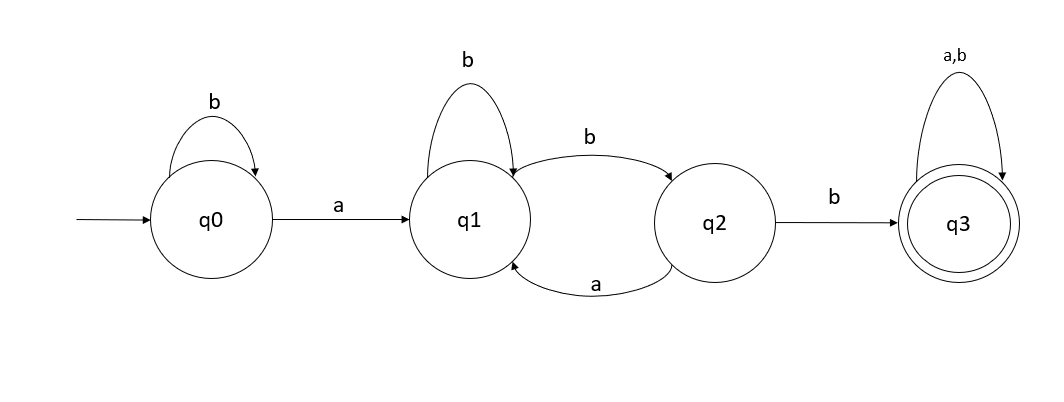


Explanation:

1. Initial state is q0.
2. The correct string that should be accepted by this DFA is of the form ‘abbbba’ because there should be ‘a’ followed by minimum of three ‘b’s and there need to be a total of at-least two ‘a’ s after the sequence of minimum of three ‘b’s to satisfy this condition.
3. ‘abbbaa’ – is the lease valid case for this DFA.
4. Its should also accept ‘abbbbaaa’
5. But not ‘aabbbaa’ because this does not follow the pattern of one ‘a’ followed by a minimum of three ‘b’.

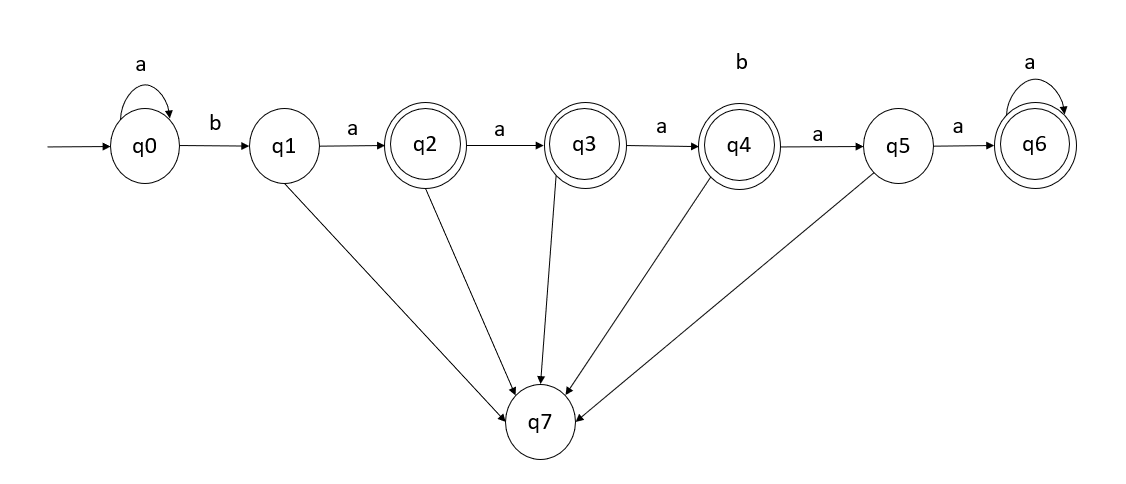
#5 c)





#5 d)





Explanation: This is a straightforward question, this DFA should only accept values like:

1. ‘ba’
2. ‘baa’
3. ‘baaa’
4. ‘baaaaa’
5. but not ‘baaaa’ (With 4 ‘a’s)
6. Any other values like ‘bab’, ‘baab’ should end up in dead state (q7)
7. Initial state 🡪 q0, final states 🡪 q2, q3, q4, q5 and dead state 🡪 q7

#1



This question is very vague, I have difficulty in understanding what “all integer numbers in C” refers to.

#2



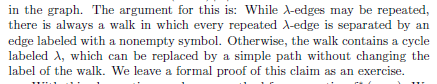
Ω is the length of input word. The length of the walk is the number of edges in the walk.

P = p1e1, p2e2…..p|w| e|w|, p|w|+1

P will always be less than or equal to lambda.

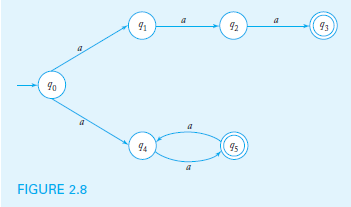
Hence the equation (1+∆)|w| + ∆ is proved.

Excerpt from the textbook Page 54



#3





There is only one input ‘a’

q0 🡪 {q1, q4}

{q1, q4} 🡪 {q2, q5}

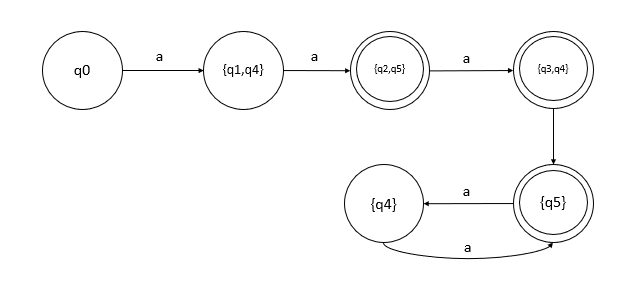
{q2, q5} 🡪 {q3, q4}

{q3, q4} 🡪 {q5}

{q5} 🡪 {q4}

{q4} 🡪 {q5}

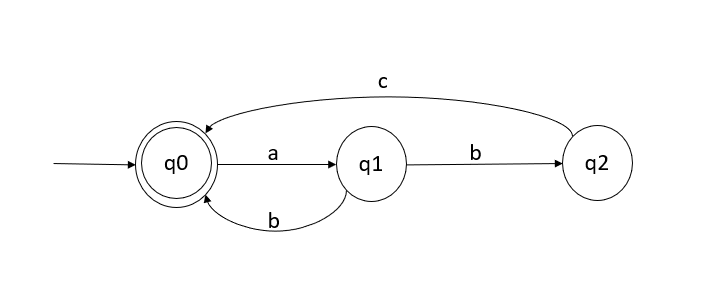
This can be re-written as:



initial state of q0 and final states {q2, q5}, {q3, q4}, {q5}

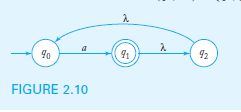
#9





#1





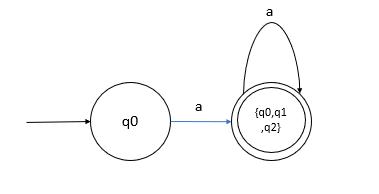
δ'({q0}, a) = ∈ - closure (δ(q0, a)) = ∈ - closure(q1) = {q0, q1, q2}

using above logic, we could say:

{q0} 🡪 {q0, q1, q2}

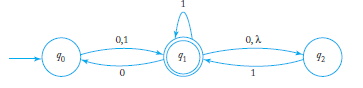
{q0, q1, q2} 🡪 {q0, q1, q2}

A simplified version of this DFA could be:



#2



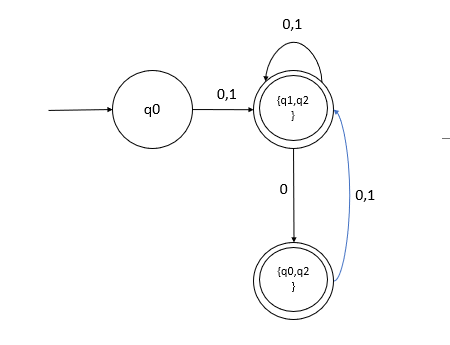


0 1

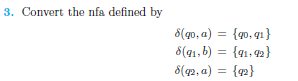
{q0} 🡪 {q1, q2} {q1, q2}

{q1, q2} 🡪 {q0, q2} {q1, q2}

{q0, q2} 🡪 {q1, q2} {q1, q2}



#3



Since the initial state is q0 and input of ‘a’ at the initial state gives us states {q0, q1} from the given statement.

δ({q0}, a) = {q0, q1}

δ({q0}, b) = dead state

δ({q0, q1}, a) = {q0, q1}

δ({q0, q1}, b) = {q1, q2}

δ({q1, q2}, a) = { q2}

δ({q1, q2}, b) = {q1, q2}

q2 is the final state and

δ({q2}, a) = {q2} 🡪 an input of a, remain at same state q2

δ({q2}, b) = dead state 🡪 invalid or dead state

#8



L(M) for DFA = {w ∈ Σ\*: δ\*(q0, w) ∈ F))

L(M) for NFA = {w ∈ Σ\* : δ\*(q0,w) ∩ F ≠ Φ)

The transition in DFA will always result in a single deterministic state hence the name Deterministic Finite Automata whereas the NFA Non-Deterministic Finite Automata, the transition will give a set of states of which at-least one state will accept the input string, hence we cannot use epsilon which would mean all the nondeterministic states should satisfy the input string or the condition which would negate the NFA statement. Hence statement should be false.

#9



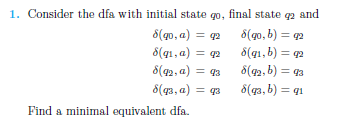
Every NFA state can be converted to equivalent NFA with only one final state, create a new final state and convert all the existing NFA states by adding epsilon transactions.

No, epsilon transitions cannot be made in DFA.

Ref; <https://www.seas.upenn.edu/~cit596/notes/dave/regexp-nfa4.html>

2.4

#1



The initial state is q0 and the final state is q2

a b lambda

{qo} {q0, q1}

{q1} {q1, q2} {q1, q2}

{q2} {q2}

**Conversion**

a b

{q0} {q0, q1}

{q0, q1} {q0, q1} {q1, q2}

{q1, q2} {q2} {q1, q2}

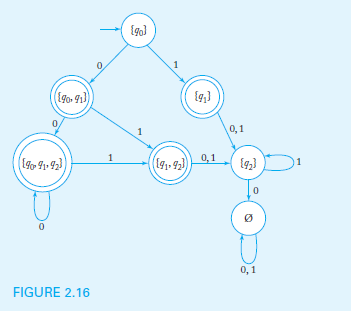
{q2} {q}

Therefore, we can say:

{q0, a} 🡪 {q0, q1}

{(q0, q1), a} 🡪 {q0, q1} 🡪 from the above conversion table





Initial state q0

Final states {q0, q1}, {q1}, {q0, q1, q2}, {q1, q2}

Grouping states together final and non-final states

Assume {q0, q1} 🡪 {q3}

q0, q1, q2 🡪 {q4}

q1, q2 🡪 {q5}

Final states (G1)🡪 {q0, q2, ∅}

Non final states (G2) 🡪 {q1, q3, q4, q5}

**δ(q0, 0)** 🡺 **{q0,q1} 🡪 {q3} = G2**

**δ(q0, 1)** 🡺**{q1} = G2**

δ(q2, 0) 🡺 {∅} = G1

δ(q2, 1) 🡺 {q2} = G1

δ(∅, 0) 🡺 {∅} = G1

δ(∅,1) 🡺 {∅} = G1

check the figure above.

q0 state on input 0 and 1 transitions to a state in different group G2

Similarly,

**δ(q1, 0) = G1**

**δ(q1, 1) = G1**

δ(q3, 0) = G2

δ(q3, 1) = G2

δ(q4, 0) = G2

δ(q4, 1) = G2

**δ(q5, 0) = G1**

**δ(q5, 1) = G1**

q1, q5 belongs to same group.

q3, q4 belongs to same group.

G1 🡪 q0

G2 🡪 q2, ∅

G3 🡪 {q1, q5}

G4 🡪 {q3, q4}

Transitions for 4 groups

G1

δ(q0, 0) = G4

δ(q0, 1) = G4

G2

δ(q2, 0) = G2

δ(q2, 1) = G2

δ(∅, 0) = G2

δ(∅, 1) = G2

G3

δ(q1, 0) = G2

δ(q1, 1) = G2

δ(q5, 0) = G2

δ(q5, 1) = G2

G3

δ(q3, 0) = G4

δ(q3, 1) = G3

δ(q4, 0) = G4

δ(q4, 1) = G3

The groups which are equal are highlighted in the same color.

So, eliminate ∅, q5 q4(newly created states)

Remaining states q0, q1, q2, q3

δ(q0, 0) = G4 = q3

δ(q0, 1) = G3 = q1

δ(q1, 0) = G4 = q2

δ(q1, 1) = G3 = q2

δ(q2, 0) = G4 = q2

δ(q2, 1) = G3 = q2

δ(q3, 0) = G4 = q3

δ(q3, 1) = G3 = q1

