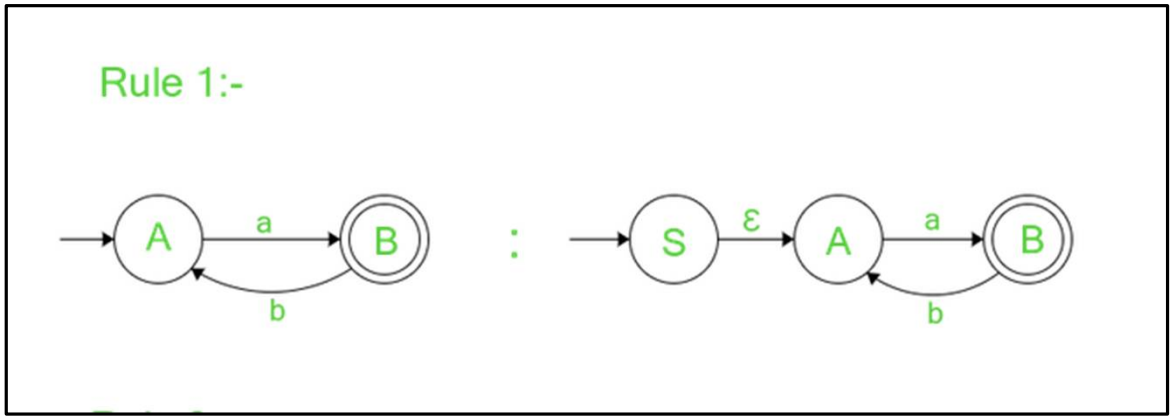
State elimination method

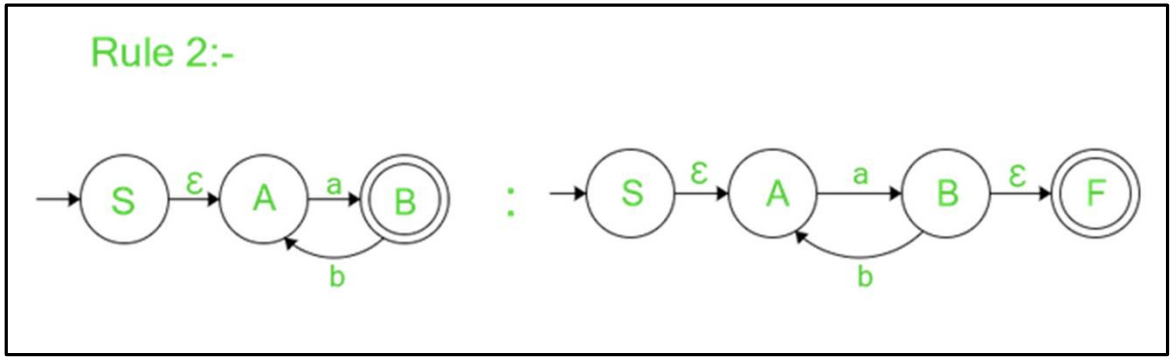
- Rules to convert a DFA/NFA//Ɛ-NFA into corresponding Regular Expression.

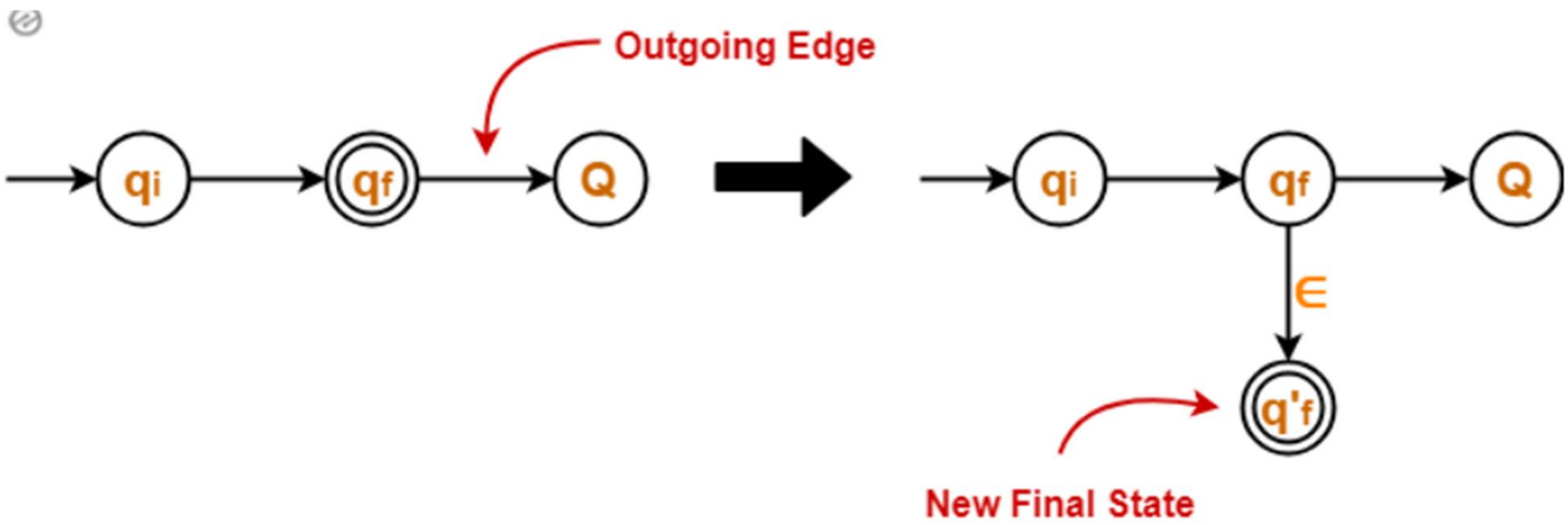
- Arden's Method is not capable of converting Ɛ-NFA. By state elimination method you can conveniently and quickly find RE without writing anything just by imagination.

Rule-1 : If there are no incoming edges to the start state proceed further to check other rules. But, **If there are incoming transitions to the initial state, to get rid of incoming edges make new start with no incoming edges** and an outgoing edge to the old start state with ε-transition. the initial state before is now normal state with added incoming ε-transition.

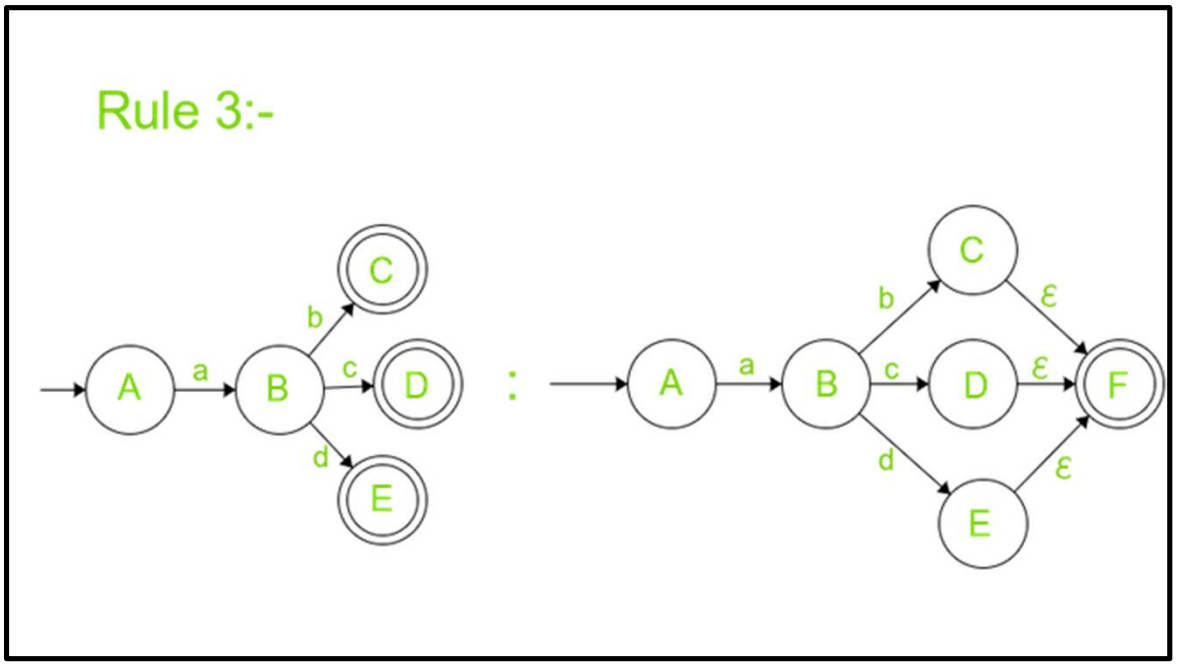


Rule-2 : If there are no outgoing edges from final state proceed further to check the last rule. **If there are outgoing transitions from final state, to get rid of outgoing edges make new final state with no outgoing edges and an incoming edge from old final state of ε-transition.** Old final state is transformed into normal state with the added transition of ε.



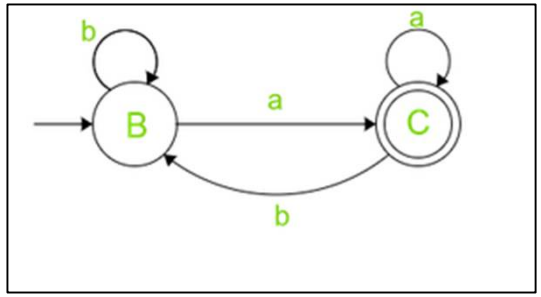

Outgoing Edge
qi
qf
Q
qi
qf
Q
∈
q'f
New Final State

Rule-3 : If there are no multiple final states **proceed to elimination** (except final and initial) of normal states. If the Automata have multiple final states it is recommended to strip their status of being final states and add outgoing ε-transition to new and only final state with no outgoing transitions.

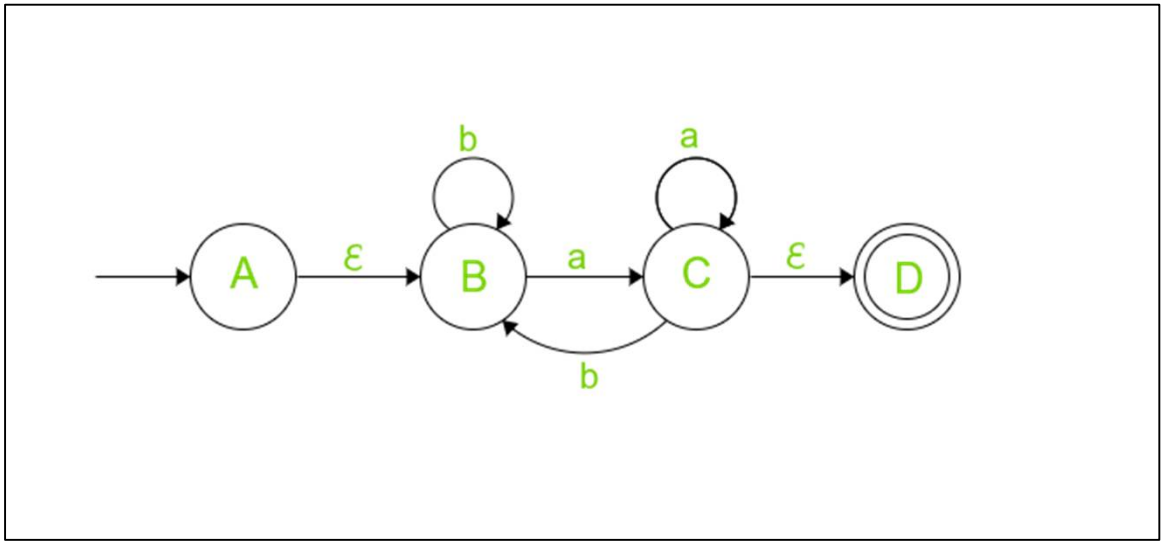


Elimination of Normal States : Let's take an example of DFA, and try to find RE by State Elimination method.

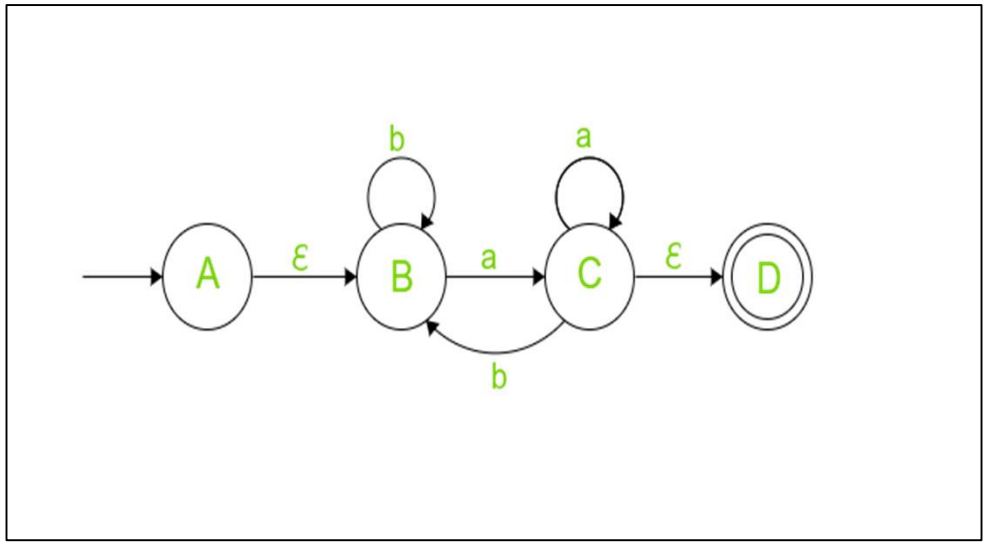
DFA Ending with a

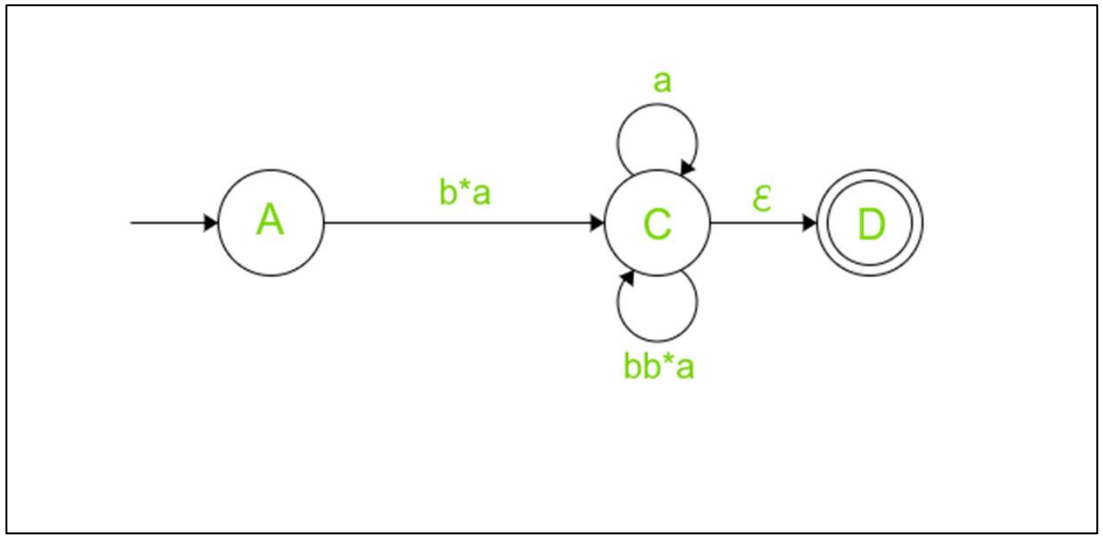


Now Rule 1, 2 is violated to satisfy them to make a new final state and new initial state with incoming ε-transition and outgoing ε-transition respectively.

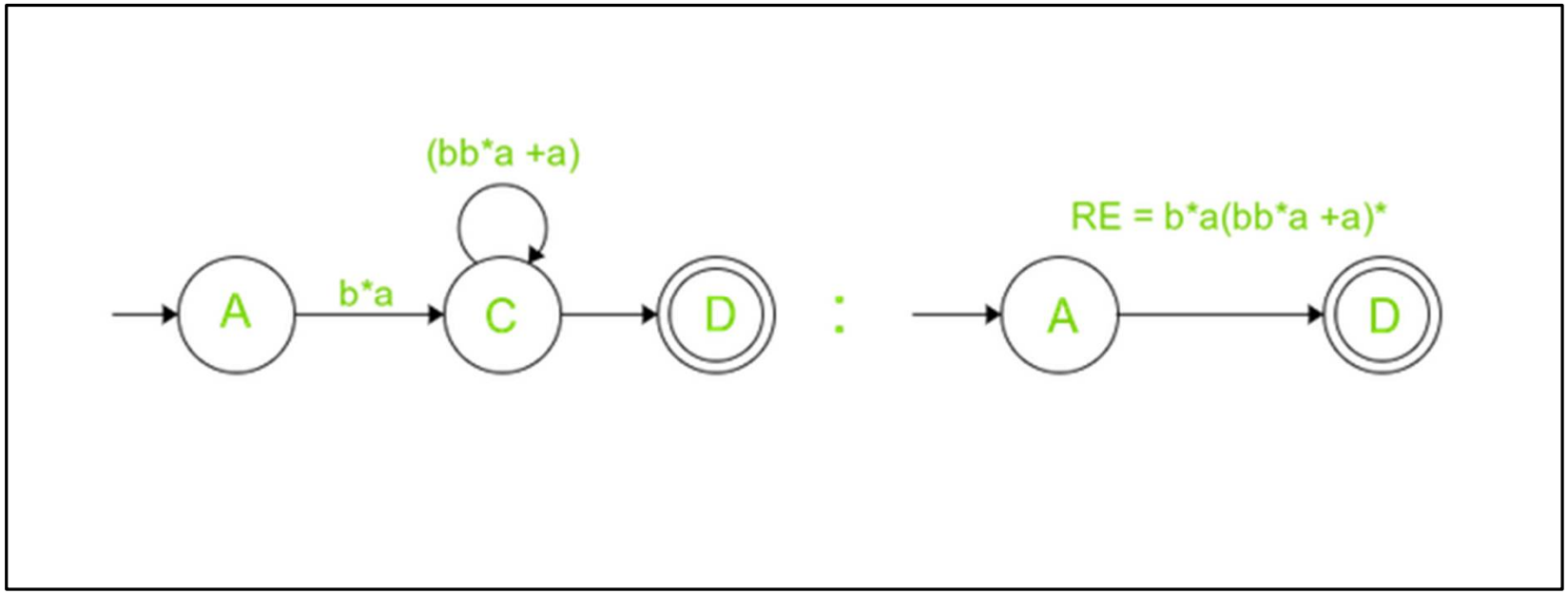


- Eliminate State B and C in any order as you wish answer will be the same in both cases, now let's eliminate B first. Eliminate B and imagine if there wouldn't have been any State B then we reach State C directly from Start state A with the input symbol of 'Ɛb*a' we can neglect Ɛ then input symbol will be 'b*a' from A to C. Outgoing edge from State C to B is now become self Loop since State B is not there, compare images above and below after removal of B state. Edge from C is going to B, taking b* and returning to C with 'a' this can happen infinite time hence self-loop of b(b)*a. We can the two self-loops 'a' and 'bb*a' as (bb*a +a) meaning we can traverse one loop at a time. Either a or bb*a as resultant (bb*a + a) is a closure on C state = (bb*a + a)*.

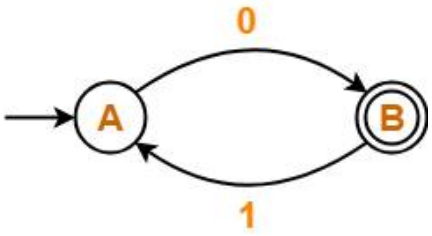


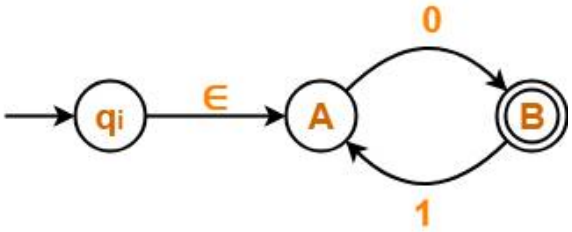


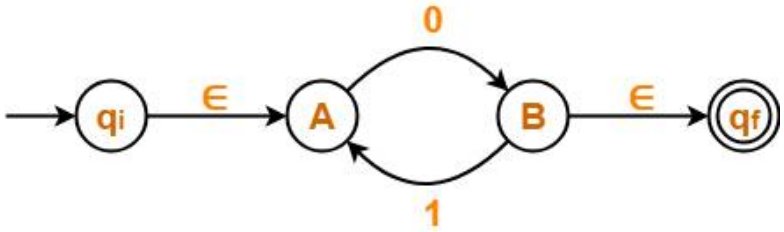
Now after eliminating C state, We would reach final State D directly from initial state A, now we were reaching State C by input of ‘b*a’ on C state we are taking infinite loops on (bb*a + a) and with Ɛ reaching final state D. hence Finally no normal states remaining we got a stream of input symbols directly from initial to final which is b*a(bb*a + a)*



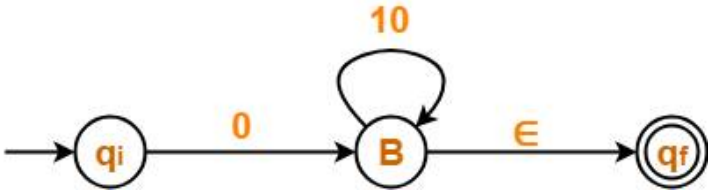
Find regular expression for the following DFA-







We start eliminating the intermediate states.

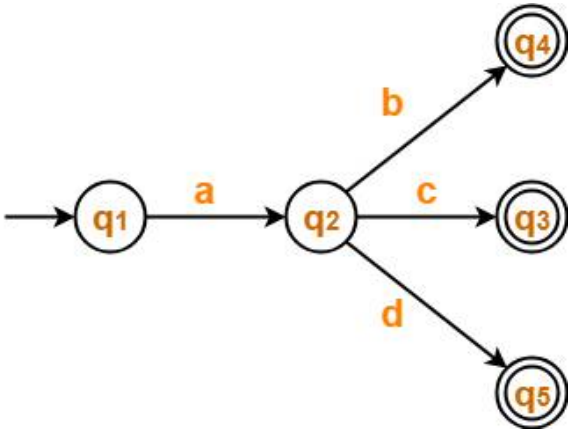


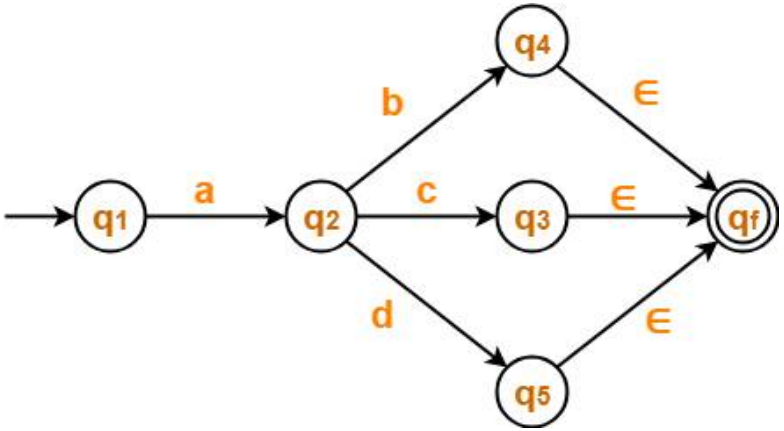
First, let us eliminate state A.

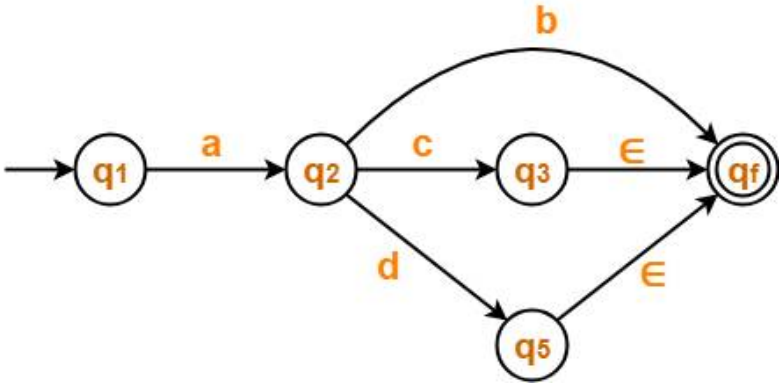
There is a path going from state qi to state B via state A.
So, after eliminating state A, we put a direct path from state qi to state B having cost ∈.0 = 0
There is a loop on state B using state A.
So, after eliminating state A, we put a direct loop on state B having cost 1.0 = 10.

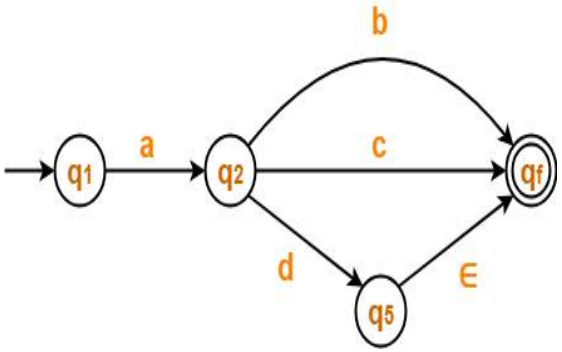
let us eliminate state B.

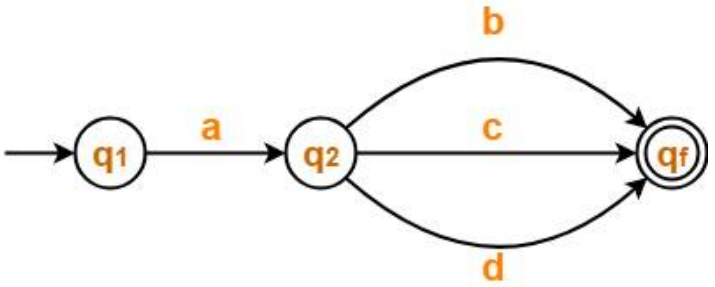
There is a path going from state qi to state qf via state B.
So, after eliminating state B, we put a direct path from state qi to state qf having cost 0.(10)*.∈ = 0(10)*


q1
a
q2
b
c
d
q4
q3
q5


q1
a
q2
b
c
d
q4
q3
q5
∈
∈
∈
qf

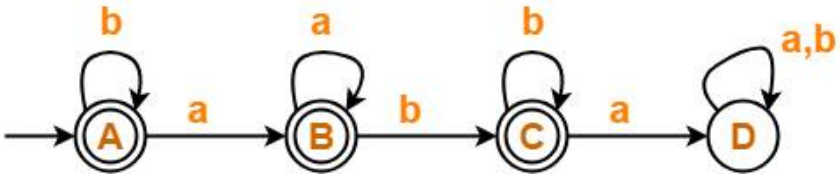

q1
a
q2
b
c
d
q3
q5
∈
∈
qf


q1
a
q2
b
c
d
q5
∈
qf

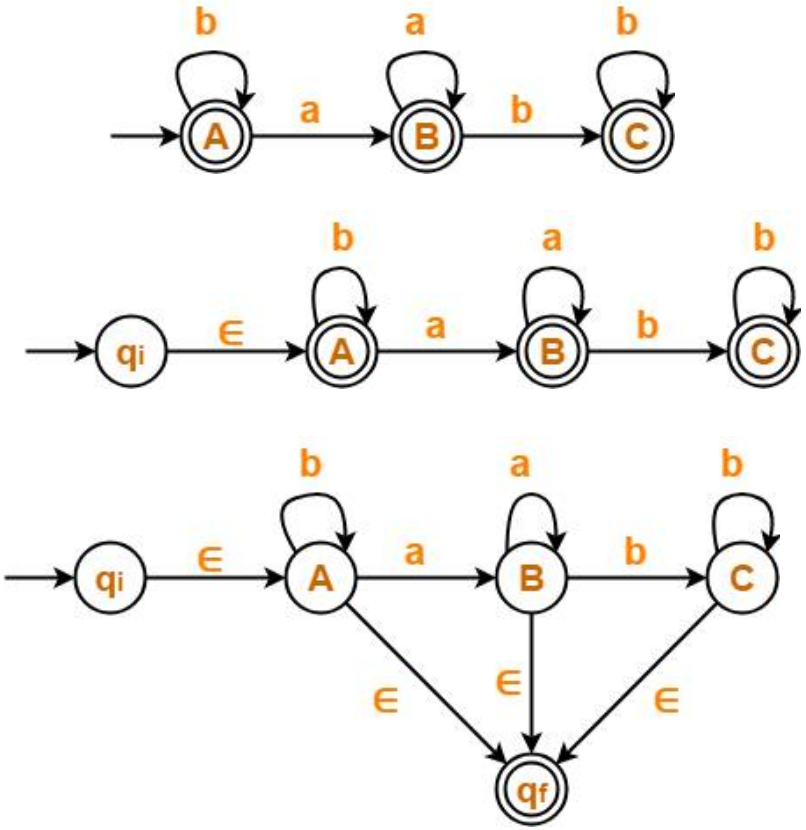

q1
a
q2
b
c
d
qf


q1
a.(b+c+d)
qf


q1
a
q2
b+c+d
qf

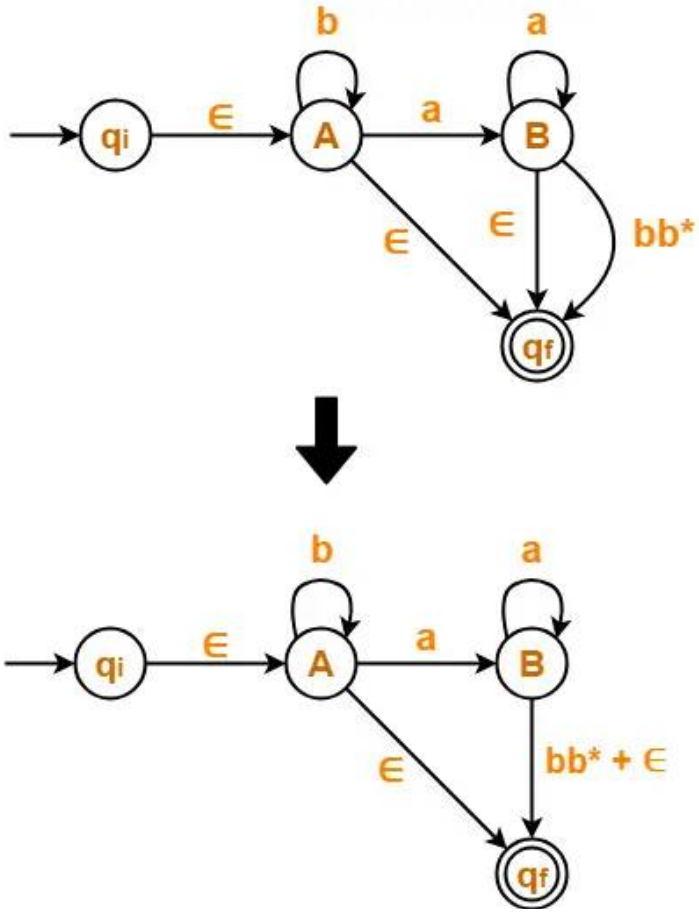


e D is a dead state as it does not reach to any final state.
So, we eliminate state D and its associated edges.



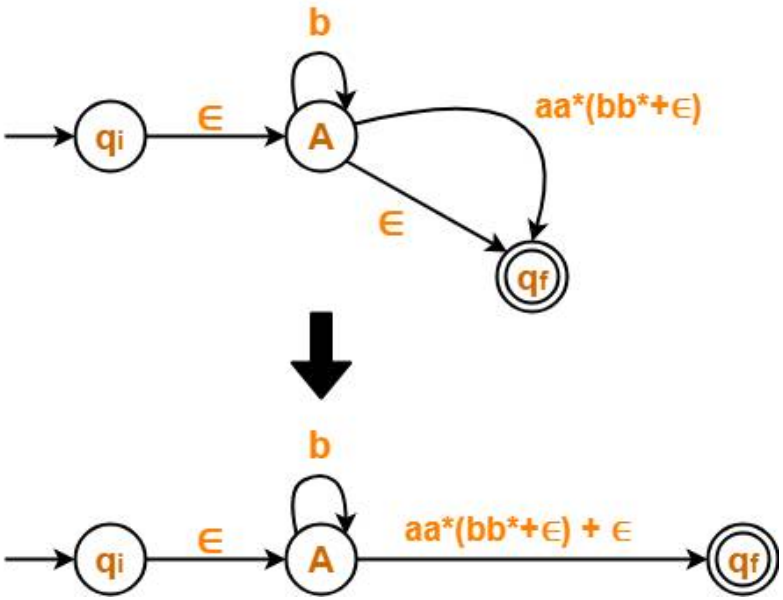
us eliminate state C.

There is a path going from state B to state qf via state C.
So, after eliminating state C, we put a direct path from state B to state qf having cost b.b*.∈ = bb*



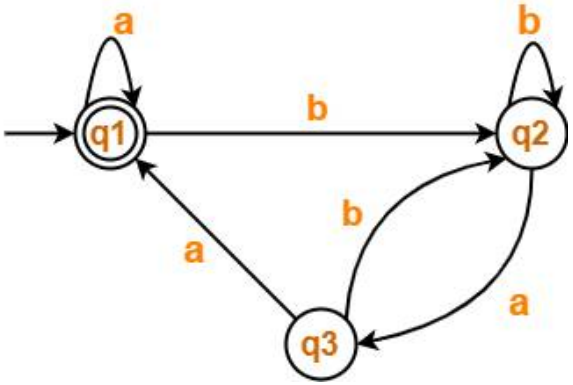
Let us eliminate state B.

There is a path going from state A to state qf via state B.
So, after eliminating state B, we put a direct path from state A to state qf having cost a.a*.(bb*+∈) = aa*(bb*+∈)



Let us eliminate state A.

There is a path going from state qi to state qf via state A.
So, after eliminating state A, we put a direct path from state qi to state qf having cost ∈.b*.(aa*(bb*+∈)+∈) = b*(aa*(bb*+∈)+∈)



Regular Expression for the given DFA = (a + b(b + ab)*aa)*

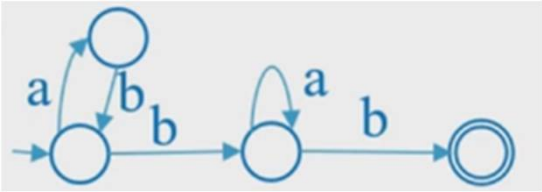
State Decomposition Method (State Creation Method)

Write RE r as an edge level for the NFA with two state.
Spilt r into symbol and create the state.
Continue the state creation process until RE is divided into symbols.

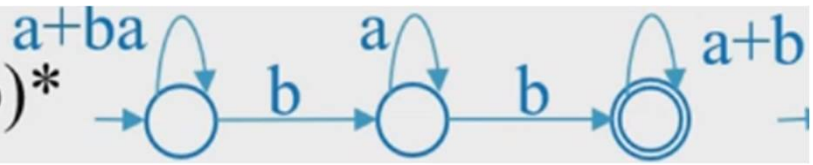
r=ab

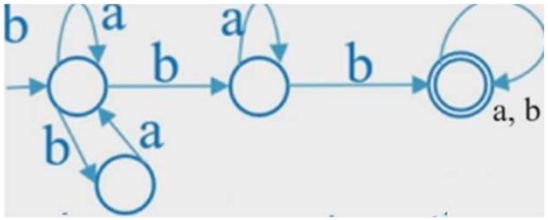
r=ab*

r=(ab)*ba*b

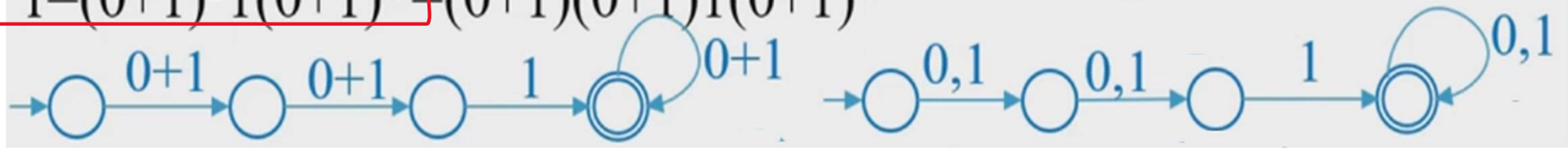


r=(a+ba)*ba*b(a+b)*

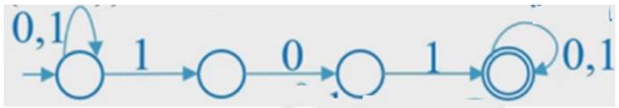




$r=(0+1)^2 1(0+1)^*=(0+1)(0+1)1(0+1)^*$



$r=((0+1)^2)^*=((0+1)(0+1))^*$



r=(a*b)*=(a*b+b)*

r=(a*b+(ba)*)*