CSci 435: Formal Languages and Automata

Instructor: Dr. M. E. Kim Name: Elena Corpus

**Home Assignment 2: 78/100 points + 10 points (optional)**

Q1. [10/10] Find all strings in L((*ab* + *b*)\* b (*a* + *ab*)\*) of length ***less than*** four.

b, ab, bb, ba, abb, bba, bbb, baa, bab

Q2. [13/20] Give a ***regular expression*** for the language

1. [7/10] L = {*anbm* | (*n*+*m*) is odd}.

(aa)\*(bb\*) - a(aa)\*b(bb\*)

EITHER the #(*a*) is odd and #(*b*)=even OR #(*a*)=even and #(*b*)=odd.

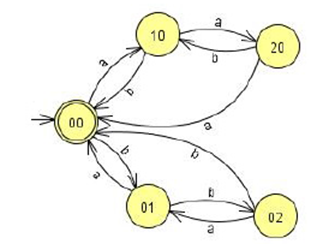
So, its REX is (*aa*)\*(*a*+*b*)(*bb*)\*

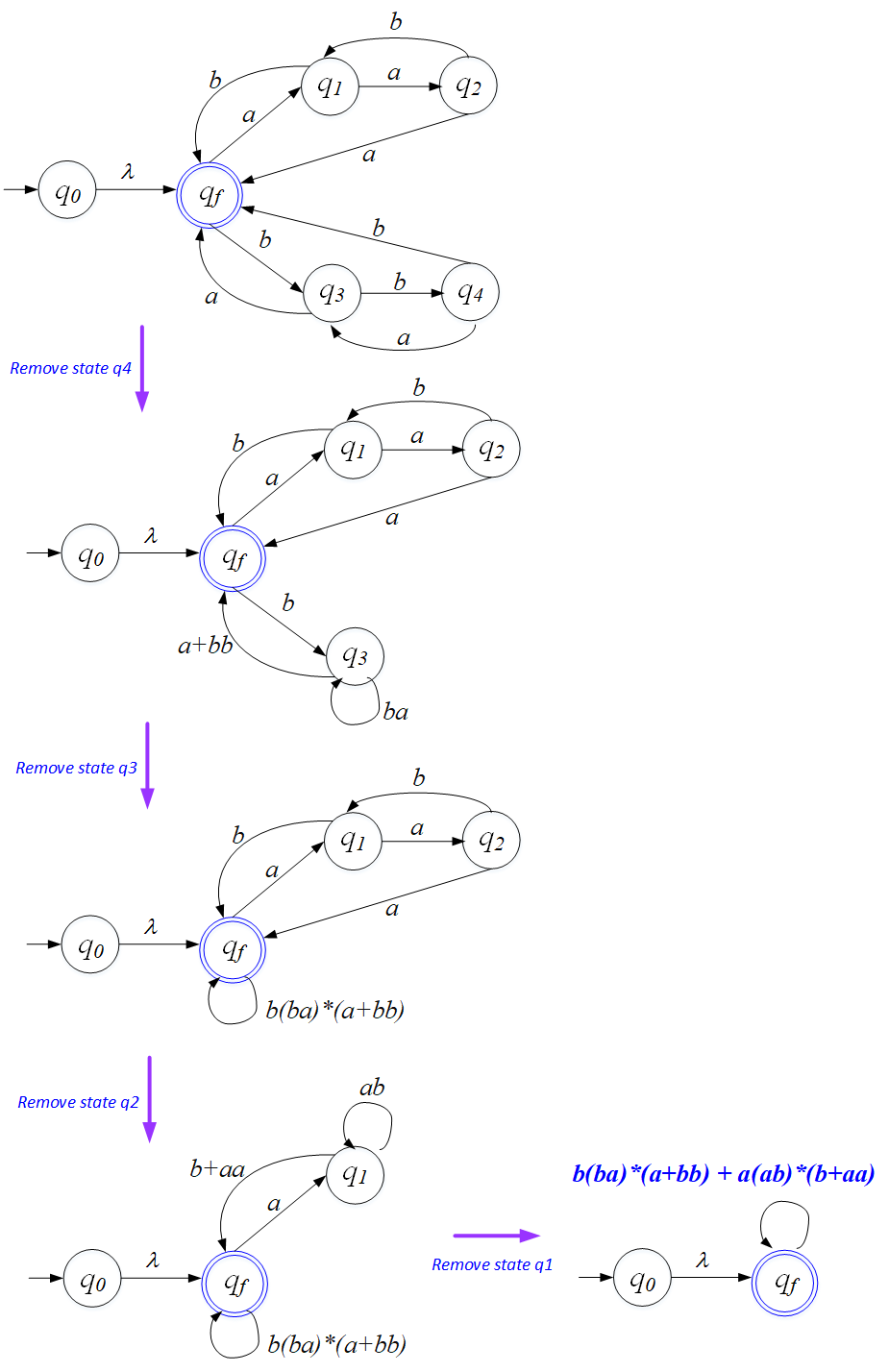
1. [6/10, optional] L = {*w* Î {*a, b*}\* | ( *na*(*w*) - *nb*(*w*) ) mod 3 = 0}. Hint: Apply Thm 3.2. .

RE = ((ab\*(ab+b)(ba)\*(bb+a)(ab)\*)\*

* See the attached sample answer

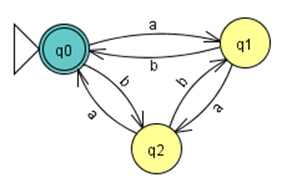
Case 1: NFA M, L(M) = L

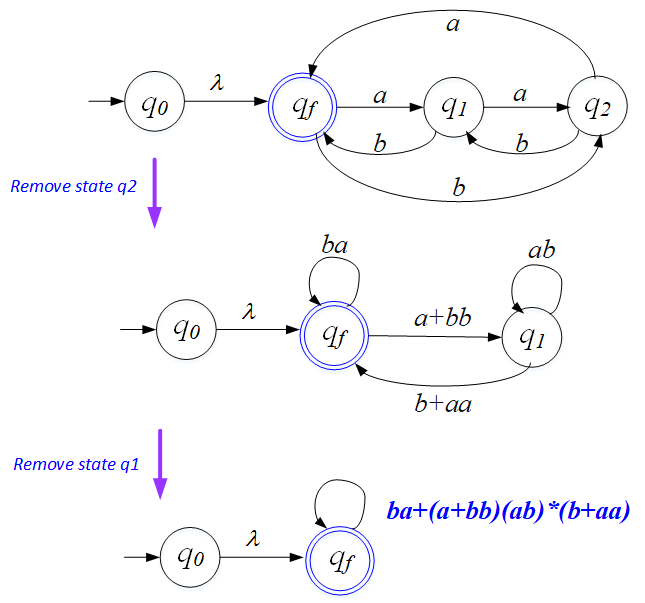




So, the REX is: *b(ba)\*(a+bb)+a(ab)\*(b+aa).*

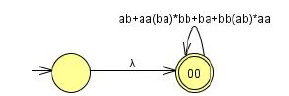
Case 2: NFA M





So, the REX is: *ba* + (*a+bb)*(*ab*)\*(*b*+*aa*).

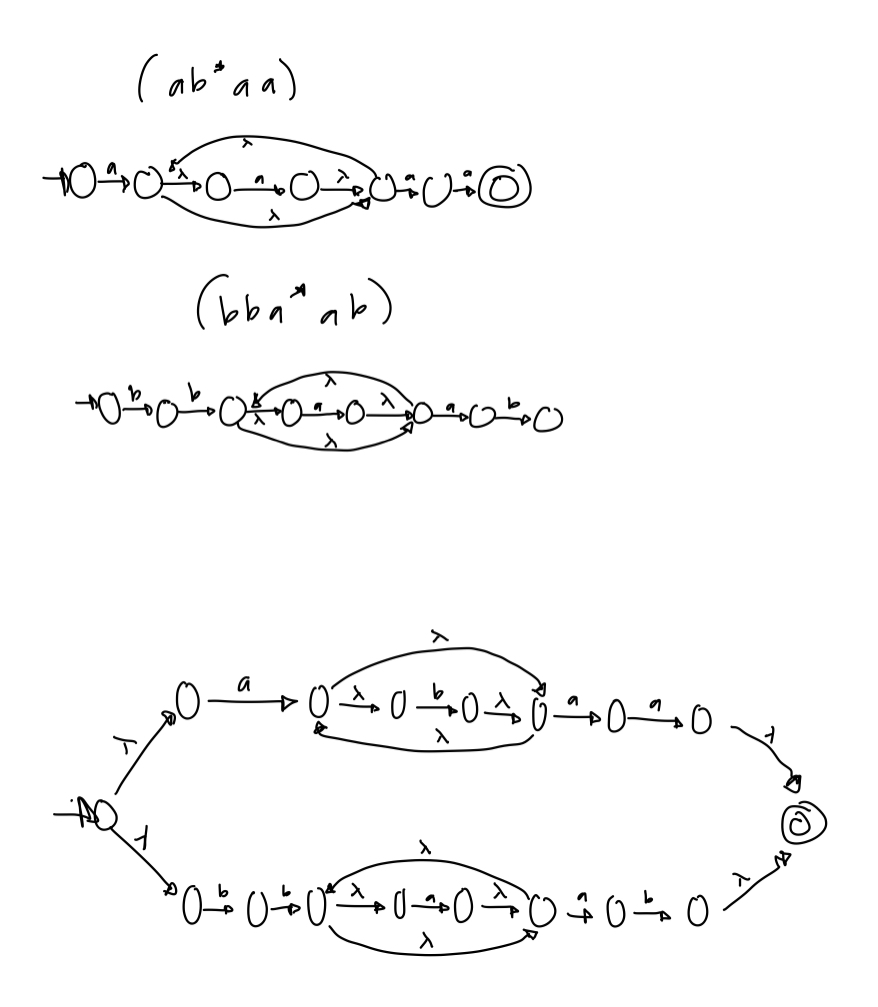
Case 3:



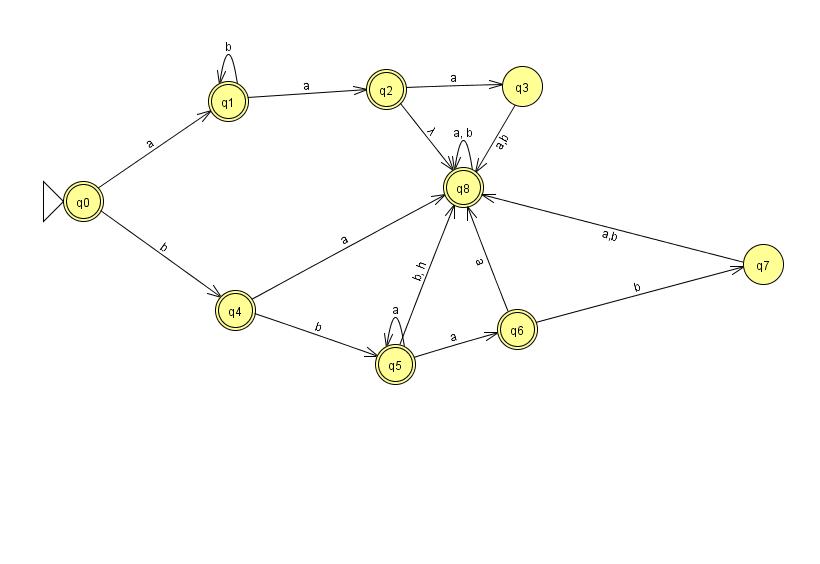
The REX is: (*ab* + *aa*(*ba*)\**bb* + *ba* + *bb*(*ab*)\**aa*)\*.

Q3. [6/10] Using the construction in Theorem 3.1, construct an NFA that accepts the complement of the

Language L(*ab*\**aa* + *bba*\**ab*).

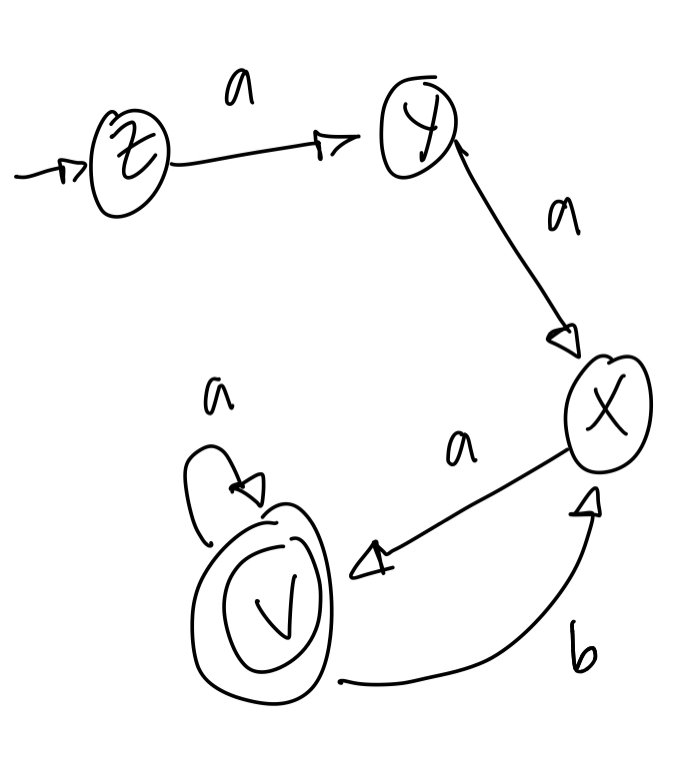


* Convert the above NFA to a DFA. Then, complement of it.
* See the attached sample answer

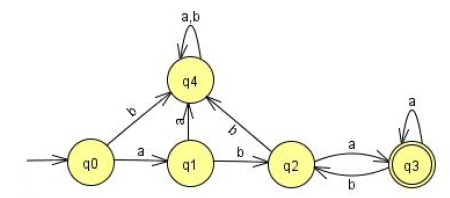


Q4. [14/20] Construct a ***minimal DFA*** that accepts the following language

1. [8/10] L(*ab*(*a*+*ab*)\*(*a*+*aa*))

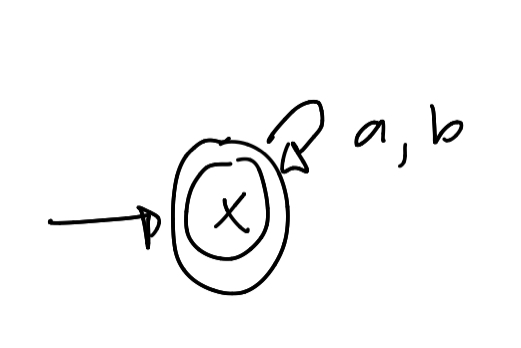


* See the attached solution



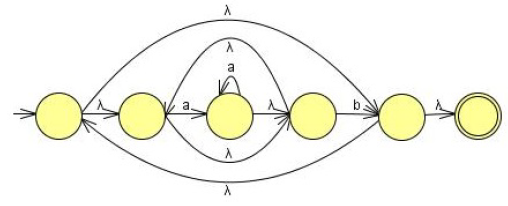
1. [6/10] L((*aa*\*)\**b*)\*)

Hint: Start with constructing an NFA (by Theorem 3.1), convert it to DFA, then get the minimal DFA by mark & reduce procedures.

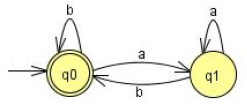


* See the attached solution

Start with

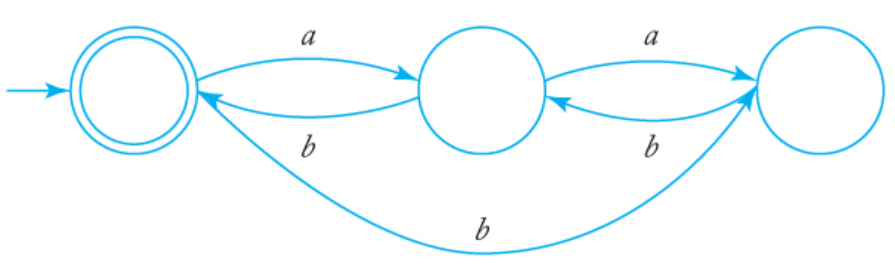


Convert NFA-to-DFA then reduce the # of states by mark-reduce in Chapter 2 to get the minimal DFA:



Q5. [14/20] Find ***regular expressions*** for the languages accepted by the following automaton.

1. [7/10]



i. q0 = q1 b + q2 b + e (e = epsilon & q0 = initial state)ii. q1 = q0 a + z3 b

iii. q2 = q1 a

Substituting eqn. (iii) in (i) and (ii)

ii. q1= q0 a + q1 ab => q1 = q0 a(ab)\* (from Arden's Theorem)

i. q0 = q1 b + q1 ab + e

Now, substituting equation (ii) into (i), we get

i. q0 = e + q0 a (ab)\* b + q0 a (ab)\* ab

=> q0 = e + q0 (a (ab)\* (b + ab))

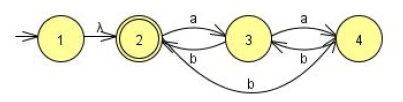
=> q0 = e (a (ab)\* (b + ab))

=> q0 = a (ab)\* (b + ab)

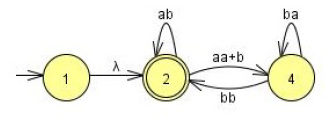
Thus, q0 **= a (ab)\* (b + ab)**

* See the attached solution

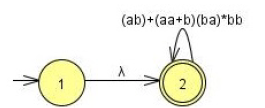
First, we have to modify the NFA so that it satisfies the conditions imposed by the construction in Theorem 3.2, one of which is q0 ∉= F.



Removing state 3, we get



Next,we remove state 4



If q0 ∈ F was allowed, NFA of a single state q2 only which is both initial and final state.

The regular expression then is: **r = (*ab* + (*aa* + *b*) (*ba*)\* *bb*)\*.**

OR

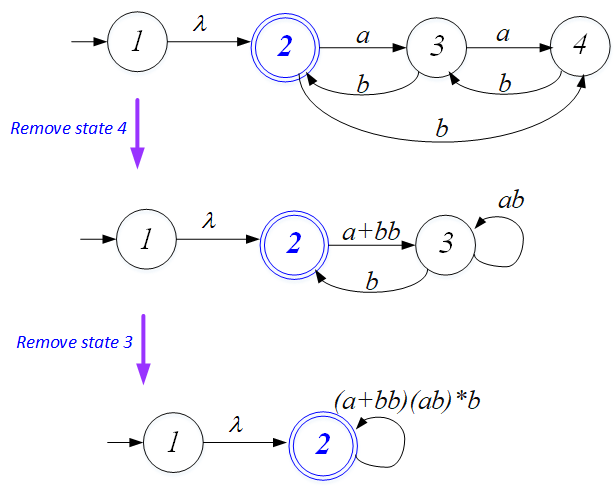
Remove 4:

2 🡪 2: 2 🡪 4 🡪 2: none

2 🡪 3: 2 🡪 4 🡪 3: *bb*. So, 2 🡪 3 = *a* + *bb*

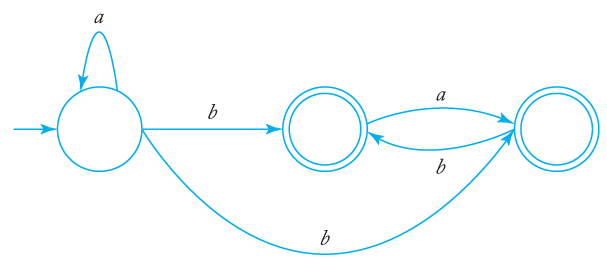
3 🡪 2 : 3 🡪 4 🡪 2 : none So, 3 🡪 2 = *b* (original)

3 🡪 3: 3 🡪 4 🡪 3: *ab*



If remove the states 4, then 3:  **r = ((*a* + *bb*)(*ab*)\**b*)\*** Both are correct.

1. [7/10]



i. q0 = q0 b + e (q0 = starting point )

ii. q1 = q0 b + q2 b

iii. q2 = q0 b + q1 a

Solving equation i) using Arden's Theorem we get,

i. q0 = e (b)\* => q0 = (b)\*

Substituting value of q0 in equation (ii) and (iii), we get:

ii. q1 = (b)\* b + q2 b

iii. q2 = (b)\* b + q1 a

Substituting equation (ii) in (iii), we get,

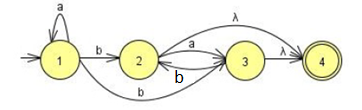
iii. q2 = (b)\* b + ( (b)\* b + q2 b ) a

=> q2 = (b)\*b + (b)\* ba + z3 ba=> q2 = ((b)\*b + (b)\* ba) (ba)\* (using Arden's Theorem)

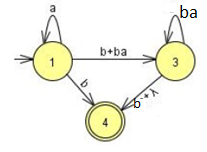
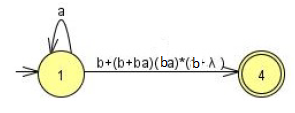
q1 = (b)\* b + (((b)\*b + (b)\* ba) (ba)\*) b

* See the attached solution

**is equivalent to**

**with a single final state.**

Remove q2: Remove q3:

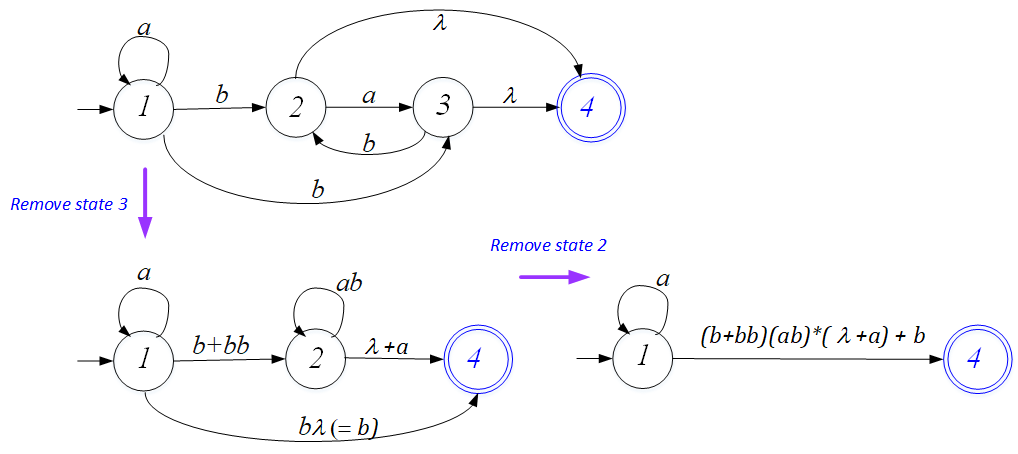
 

The regular expression then is: **r = *a*\*(b + (b + b*a*)(b*a*)\*(b + λ))**

**= *a*\*(b + b( λ + *a*)(*ba*)\*(b + λ )**

**= a\*b(λ + ( λ + *a*)(*ba*)\*(b + λ )**

OR



The regular expression then is: **r = *a*\*(b + (*b* + *bb*)(*ab*)\*( λ+*a*))**

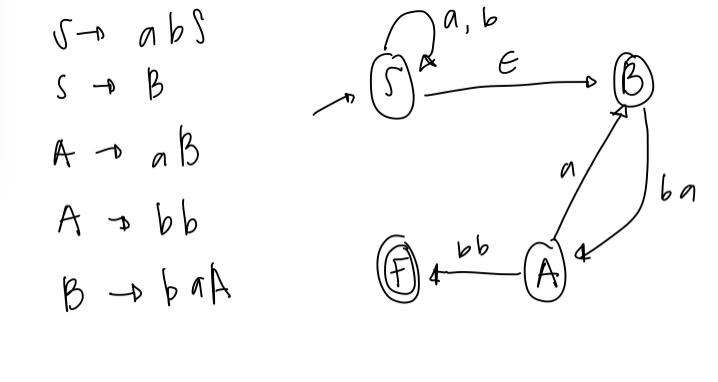
**= *a*\*(b + (*λ* + *b*)b(*ab*)\*( λ+*a*))**

**= *a*\*(b + (*λ* + *b*)(b*a)\*b*( λ+*a*))**

**Any of them are correct.**

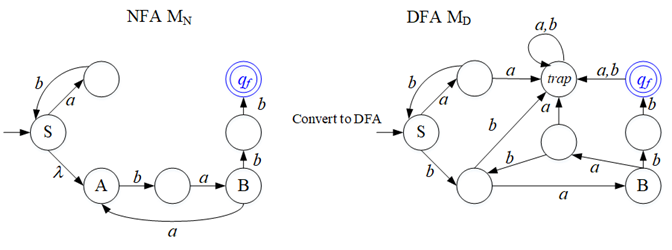
Q6. [5/10] Construct a ***DFA*** that accepts the language generated by the *grammar*

S ® *ab*S | B, A ® *a*B | *bb,* B ® *ba*A.



* See the attached solution

It is straightforward to construct a NFA for the given grammar with a λ-transition from state S to A.

Then Convert NFA to DFA.

*L*((*ab*)\**ba*(*aba*)\**bb*)

Q7. [16/20] Find a ***regular grammar*** that generates the language on S={a, b}

1. [10/10] *L*(*aa*\*(*ab*+*a*)\*)

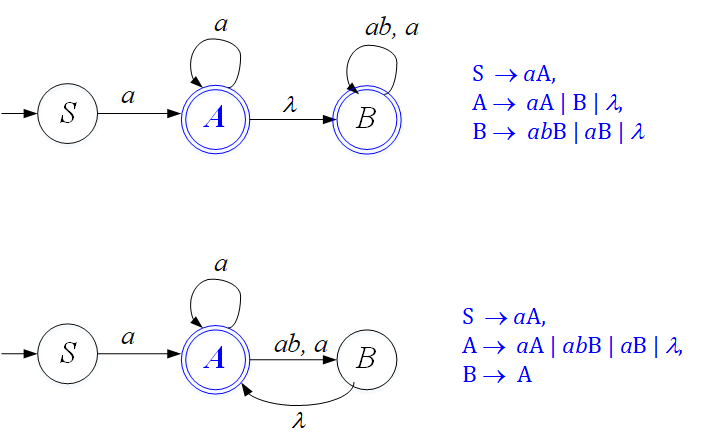
S -> aA

A -> Aa|B

B -> abB|aB\(lambda)

aA, aaA, aaaA, aaaabB, aaaababB, aaaababaB, aaaababa

* See the attached sample solution



{ S → *a*A, A → *a*A | B | λ, B → *ab*B | *a*B | λ }

or { S → *a*A, A → *aA* | *ab*B | *a*B | λ, B → A }, etc.

1. [6/10] the language consisting of all strings with no more than two *a*’s.

b\*a{0,1} b\*a{0,1} b\*

babab

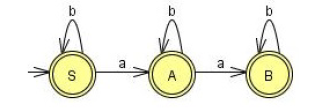
aa

aba

bbb

* See the attached sample solution
* Regular grammar is expected

A NFA that accepts strings with no a's, one a, or two a's is given below.



The construction of Theorem 3.4 then gives a right-linear grammar.

S → *b*S | *a*A| λ, A → *b*A | *a*B | λ, B → *bB* | λ