CSci 435: Formal Languages and Automata

Instructor: Dr. M. E. Kim Name: \_\_Sam Dressler\_\_\_\_\_\_\_\_\_

**Home Assignment 2: 76/100 points + 10 points (optional)**

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

L = {b, bb, ba, abb, bab, bbb, bba, baa}

\*Q2. [10/10] Give a ***regular expression*** for the language

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

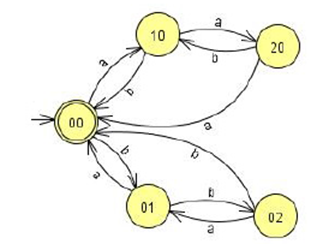
L((aa)\*(bb)\*a +(aa)\*(bb)\*b ) ={a, b, aaa, abb, baa, bbb, …}

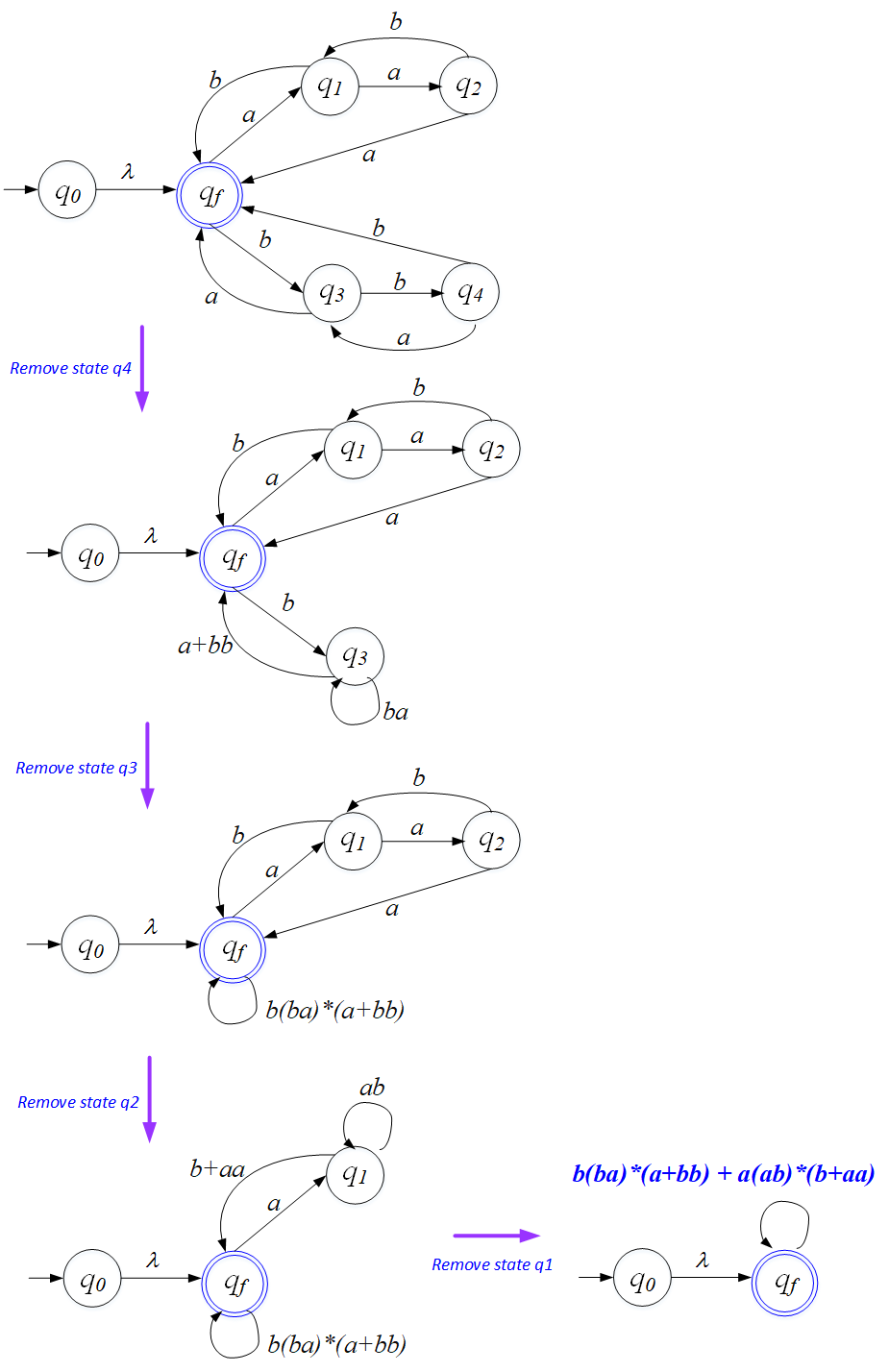
1. [0/10, optional] L = {*w* ∈ {*a, b*}\* | ( *na*(*w*) - *nb*(*w*) ) mod 3 = 0}. Hint: Apply Thm 3.2. .

No Answer.

* See the attached sample answer

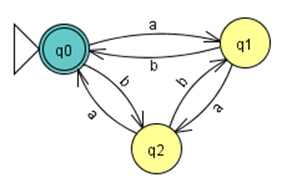
Case 1: NFA M, L(M) = L

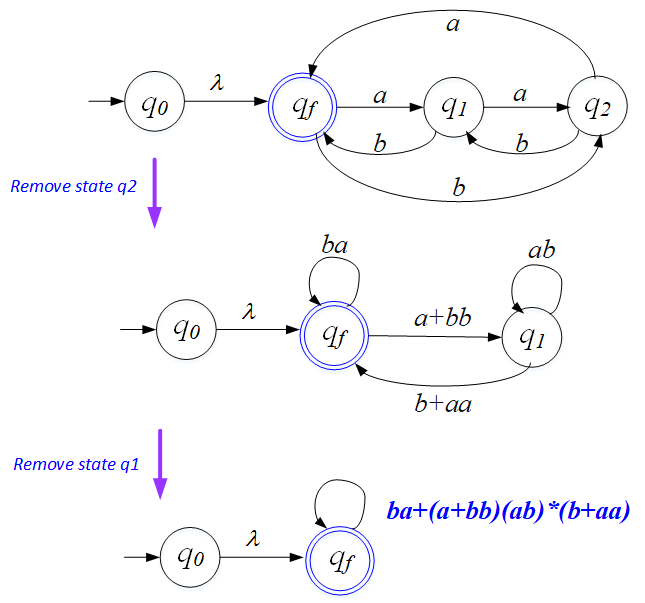




So, the REX is: *b(ba)\*(a+bb)+a(ab)\*(b+aa).*

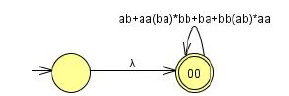
Case 2: NFA M





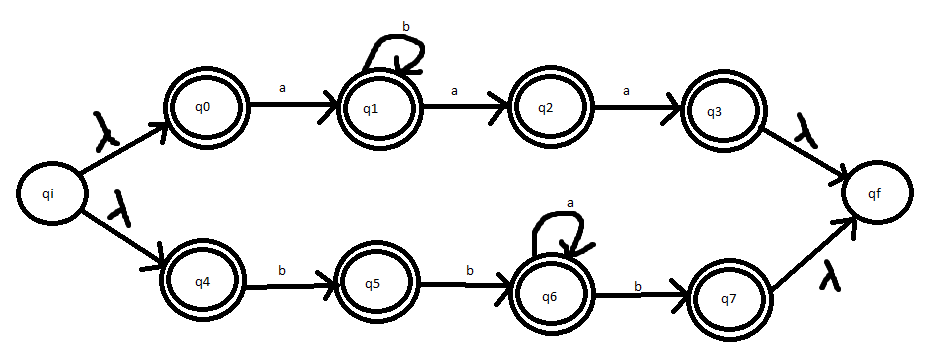
So, the REX is: *ba* + (*a+bb)*(*ab*)\*(*b*+*aa*).

Case 3:

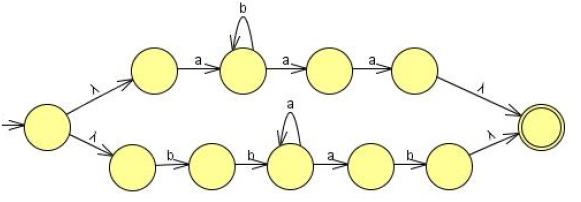


The REX is: (*ab* + *aa*(*ba*)\**bb* + *ba* + *bb*(*ab*)\**aa*)\*.

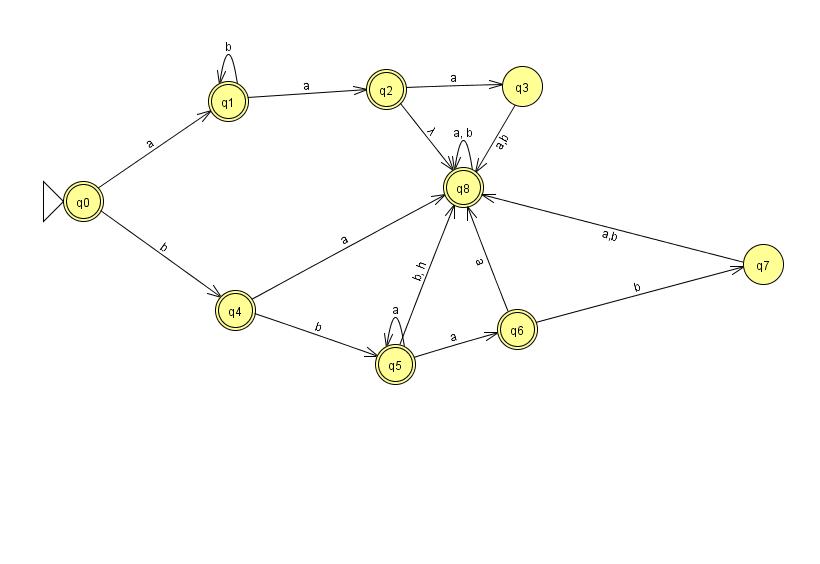
Q3. [7/10] Using the construction in Theorem 3.1, construct an NFA that accepts the complement of the

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

* Q3 and q7 are not the final states since your q3 and q7 are almost the final states before the complement.
* See the attached sample answer

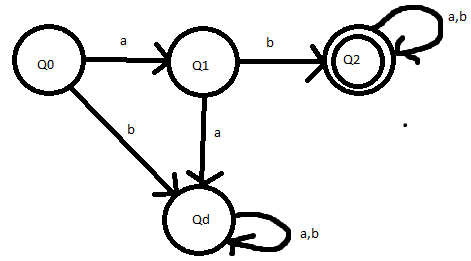


Convert the above NFA to a DFA. Then, complement of it.

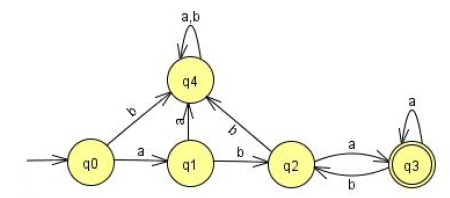


\*Q4. [10/20] Construct a ***minimal DFA*** that accepts the following language

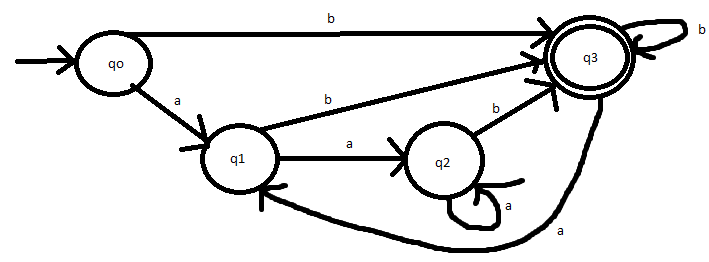
1. [3/10] L(*ab*(*a*+*ab*)\*(*a*+*aa*))



* See the attached solution



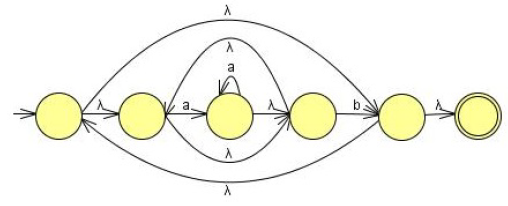
1. [7/10] L((*aa*\*)\**b*)\*)



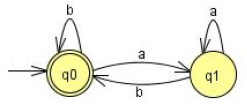
Hint: Start with constructing an NFA (by Theorem 3.1), convert it to DFA, then get the minimal DFA by mark & reduce procedures.

* See the attached solution

Start with

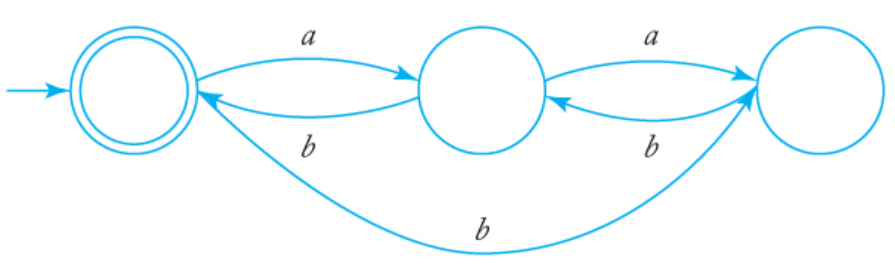


Convert NFA-to-DFA then reduce the # of states by mark-reduce in Chapter 2 to get the minimal DFA:



\*Q5. [12/20] Find ***regular expressions*** for the languages accepted by the following automaton.

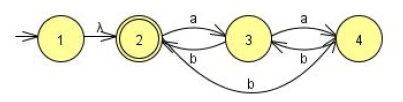
1. [6/10]



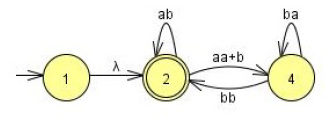
RE= L[(a((ab)\*+b)) + (b((ba)\*+bb)]

* See the attached solution

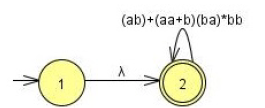
First, we have to modify the NFA so that it satisfies the conditions imposed by the construction in Theorem 3.2, one of which is q0 ∉= F.



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

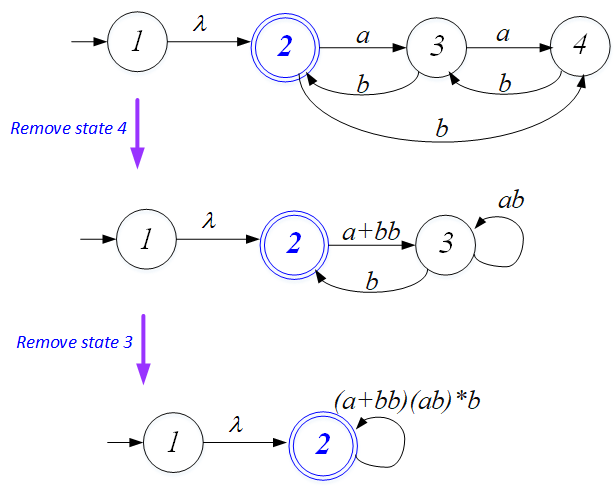
Remove 4:

2 🡪 2: 2 🡪 4 🡪 2: none

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

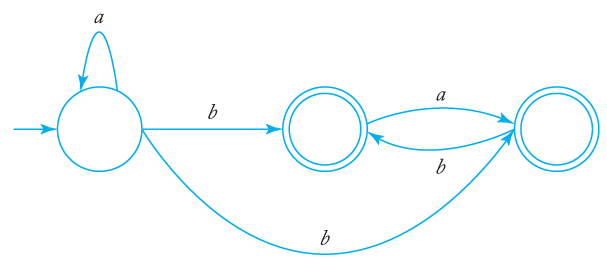
3 🡪 2 : 3 🡪 4 🡪 2 : none So, 3 🡪 2 = *b* (original)

3 🡪 3: 3 🡪 4 🡪 3: *ab*



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

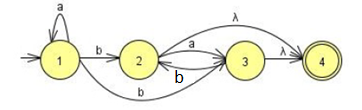
1. [6/10]



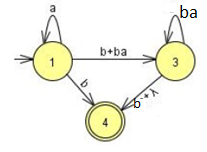
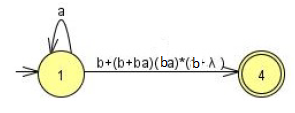
RE = L[(a\*b) + (a\*b(ba)\*)]

* See the attached solution

**is equivalent to**

**with a single final state.**

Remove q2: Remove q3:

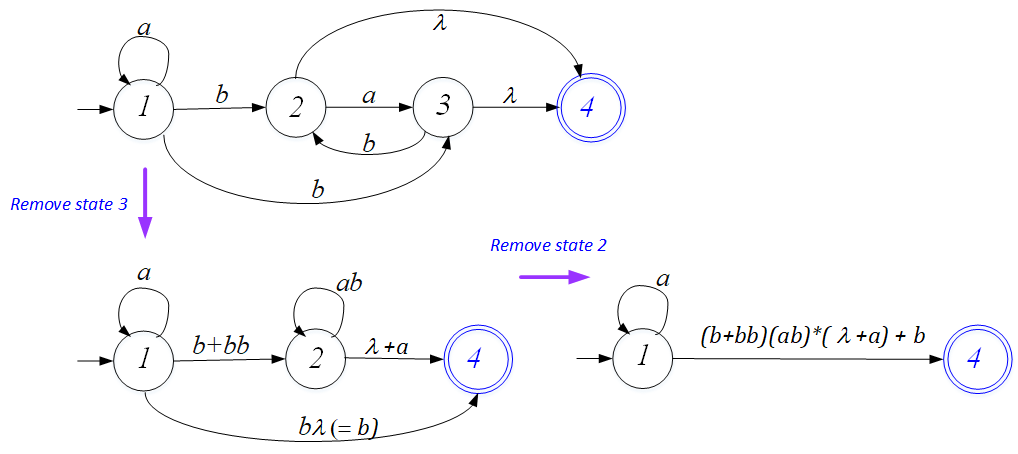
 

The regular expression then is: **r = *a*\*(b + (b + b*a*)(b*a*)\*(b + λ))**

**= *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*))**

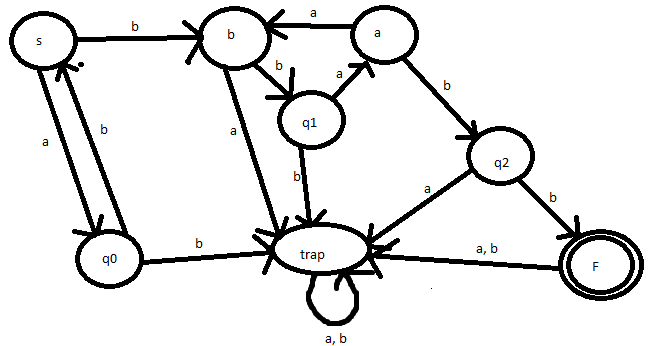
**= *a*\*(b + (*λ* + *b*)b(*ab*)\*( λ+*a*))**

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

**Any of them are correct.**

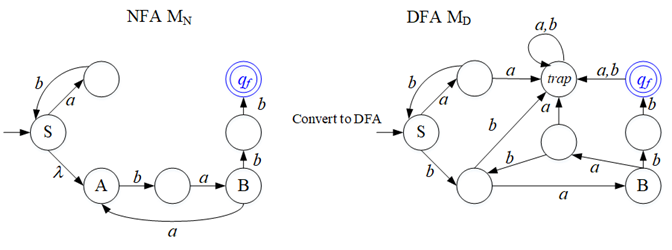
Q6. [7/10] Construct a ***DFA*** that accepts the language generated by the *grammar*

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

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

G = {V, T, S, P}

V = {S, A, B}

T = {a, b}

P ={S->aA, A-> aA |aB | λ, B -> bA}

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

G = {V, T, S, P}

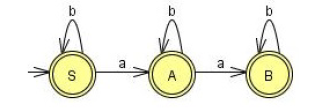
V = {S, A, B}

T = {a,b}

P = {S->bS | aB| λ, A->bA | aB| λ, B-> bB | λ}

* See the attached sample solution

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