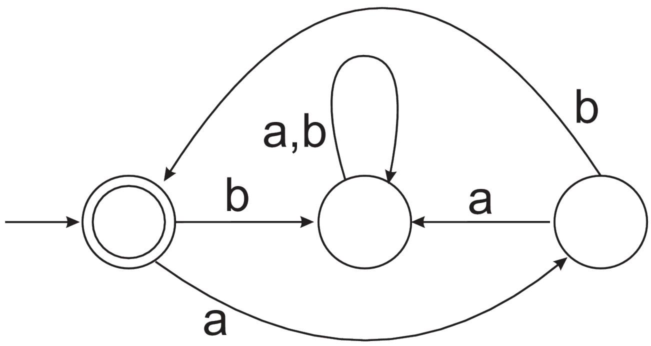
Question IV: DFAs and NFAs

1. By writing a regular expression or a grammar, describe the language accepted by the DFA



G= {(S,T,U),(a,b, *ε*), *(S),{*

With Productions,

S::= bT | aU | *ε*

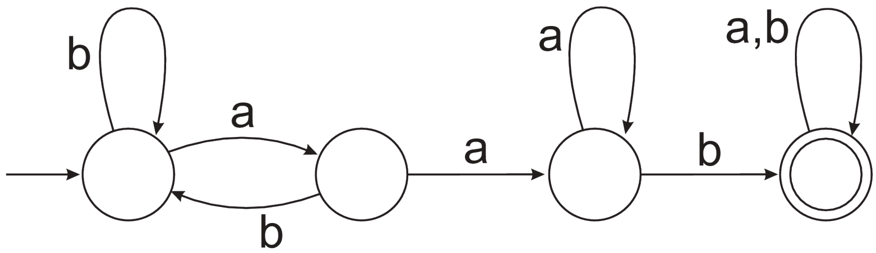
T::= aT | bT

U::= aT | bS

}

This could also be just ab* aslong as the "sink" state is never reached by taking an "b" from start state S or by taking "a" from state "U" to "T".

2. By writing a regular expression or a grammar, describe the language accepted by the DFA



G= {(S,T,A,B),(a,b, ε), (S),{

With Productions,

S::= bS | aT

T::= bS | aA

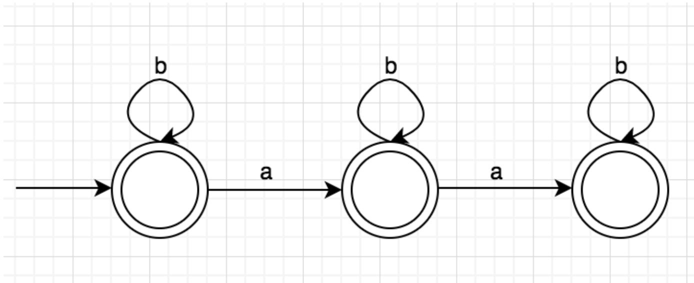
A::= aA | bB

B::= a | b | aB | bB | ε

}

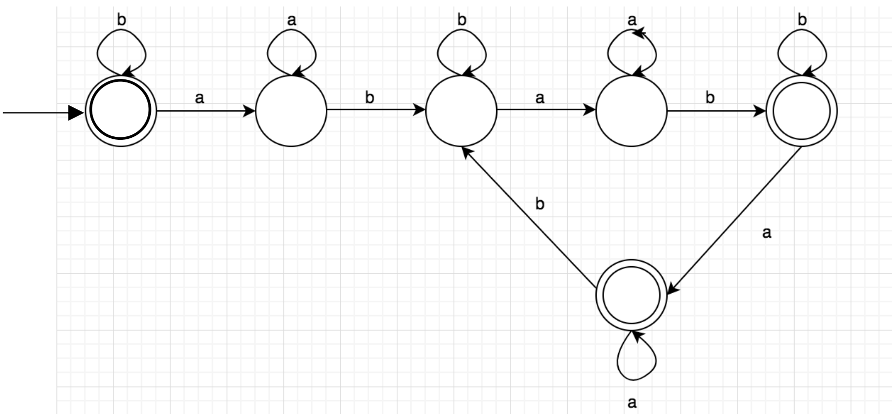
The language will only accept the language that contains atleast “aab” as a subword.

3. Construct NFAs (possibly with ϵϵ-moves) to recognise the languages on alphabet {a,b} such that:
   1. L={w∈{a,b}∗ | w contains at most two a’s }L={w∈{a,b}∗ | w contains at most two a’s }

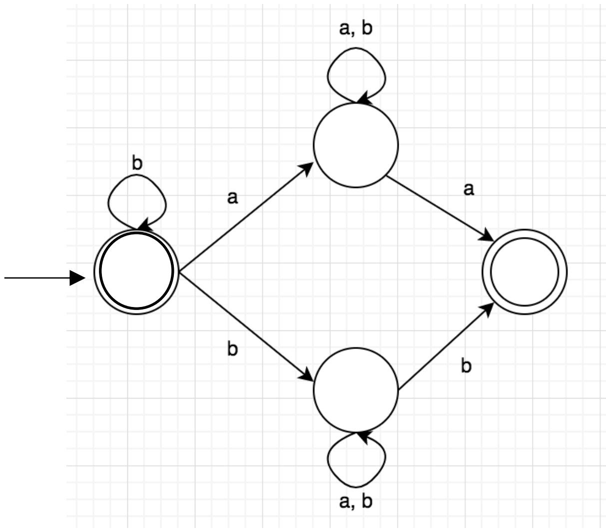


There exists three terminal states that can generate any amount of “b” however can contain either no input, all occurrences of “b” or up to two “a”s.

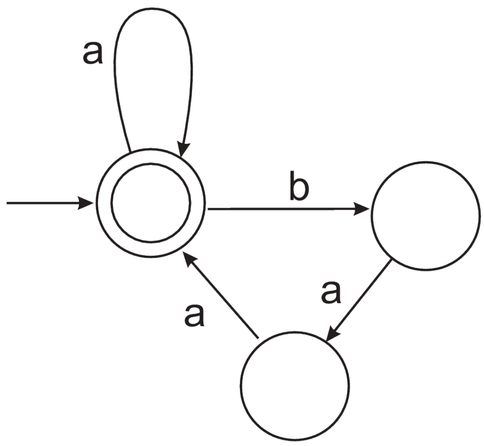
   2. L={w∈{a,b}∗ | w contains an even number of occurrences of ab as a subword }L={w∈{a,b}∗ | w contains an even number of occurrences of ab as a subword }



3. L={w∈{a,b}∗ | the first and the last letter of w are identical }L={w∈{a,b}∗ | the first and the last letter of w are identical }



4. By writing a regular expression or a grammar, describe the language accepted by the NFA



G= {(S,T,A),(a,b, *ε), (S),{*

With Productions,

S::= *ε | aS | bT*

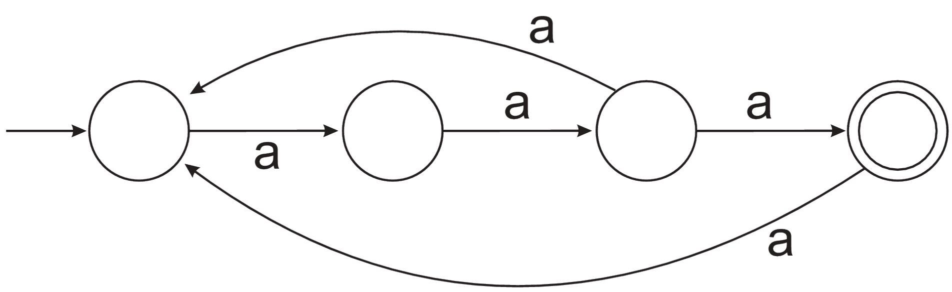
T::= aA

A::= aS

}

The language can contain the empty string or an infinite amount of a's or "baa" as a sub word.

5. By writing a regular expression or a grammar, describe the language accepted by the NFA



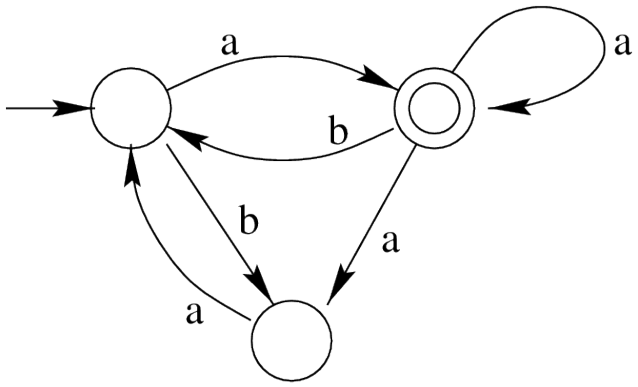
G= {(S,T,U,R),(a,b), (S),{

With Productions,

S::= aT
T::= aU
U::= aS | aR
R::= aS
}

The language can contain any amount of a's >= 3 with the exception of 4. 5, 8, 9 and 11

7. *(Hard)* Let L be the language recognised by the NFA



a. Construct a context-free grammar G that generates language L.

G= {(S,T,U),(a,b), (S),{

With Productions,

S::= aT | bU
T::= bS | aU | a | aT | ε
U::= aS
}

b. State with proof whether G is ambiguous or unambiguous.

The Grammar "G" is ambiguous as you can write the same expressions using different rules of productions. To prove this I am going to use the expression;

"aaabbaabbaaa"

Derevation One;

Rule	Application	Result
Start → S	Start	S
S → aA	S	aA
A → aA	aA	aaA
A → aA	aaA	aaaA
A → bS	aaaA	aaabS
S → bB	aaabS	aaabbB
B → aS	aaabbB	aaabbaS
S → aA	aaabbaS	aaabbaaA
A → bS	aaabbaaA	aaabbaabS
S → bB	aaabbaabS	aaabbaabbB
B → aS	aaabbaabbB	aaabbaabbaS
S → aA	aaabbaabbaS	aaabbaabbaaA
A → a	aaabbaabbaaA	Aaabbaabbaaa

Derevation Two;

Rule	Application	Result
Start → S	Start	S
S → aA	S	aA
A → aA	aA	aaA
A → aA	aaA	aaaA
A → bS	aaaA	aaabS
S → bB	aaabS	aaabbB
B → aS	aaabbB	aaabbaS
S → aA	aaabbaS	aaabbaaA
A → bS	aaabbaaA	aaabbaabS
S → bB	aaabbaabS	aaabbaabbB
B → aS	aaabbaabbB	aaabbaabbaS
S → aA	aaabbaabbaS	aaabbaabbaaA
A → aA	aaabbaabbaaA	aaabbaabbaaaA
A → ε	aaabbaabbaaaA	aaabbaabbaaa

Hence the Grammar G is ambiguous.