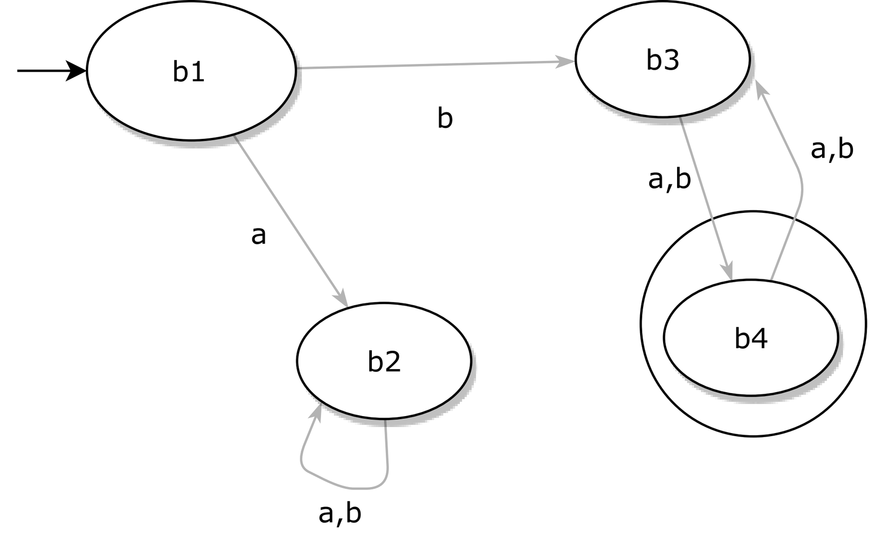
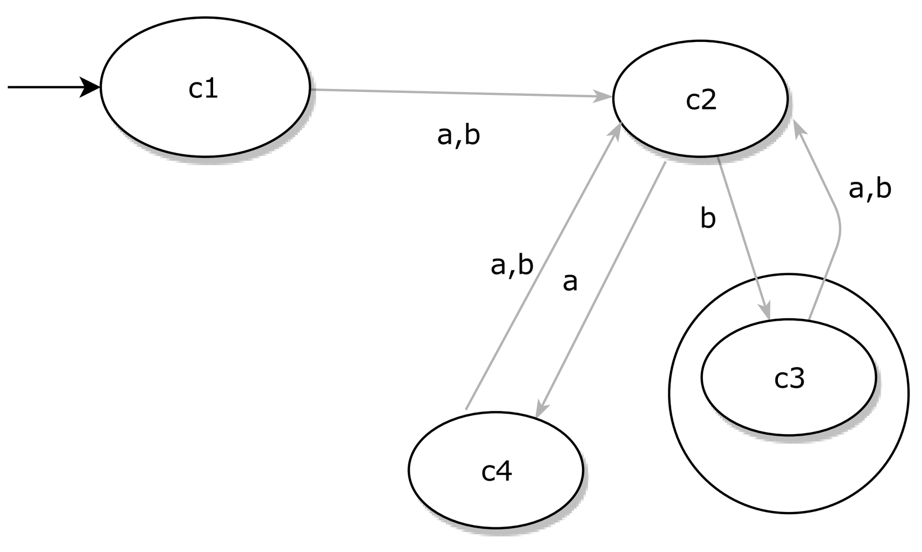
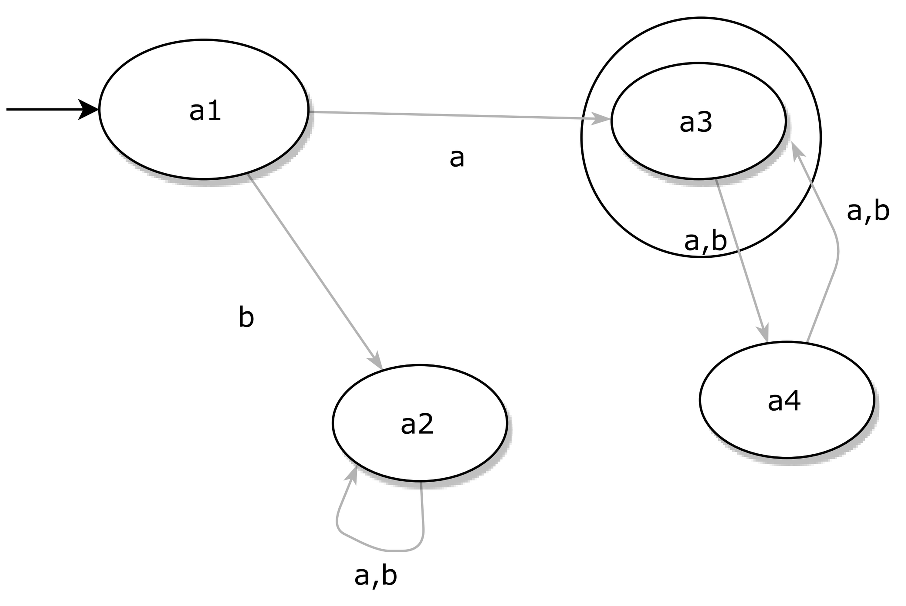
1.B



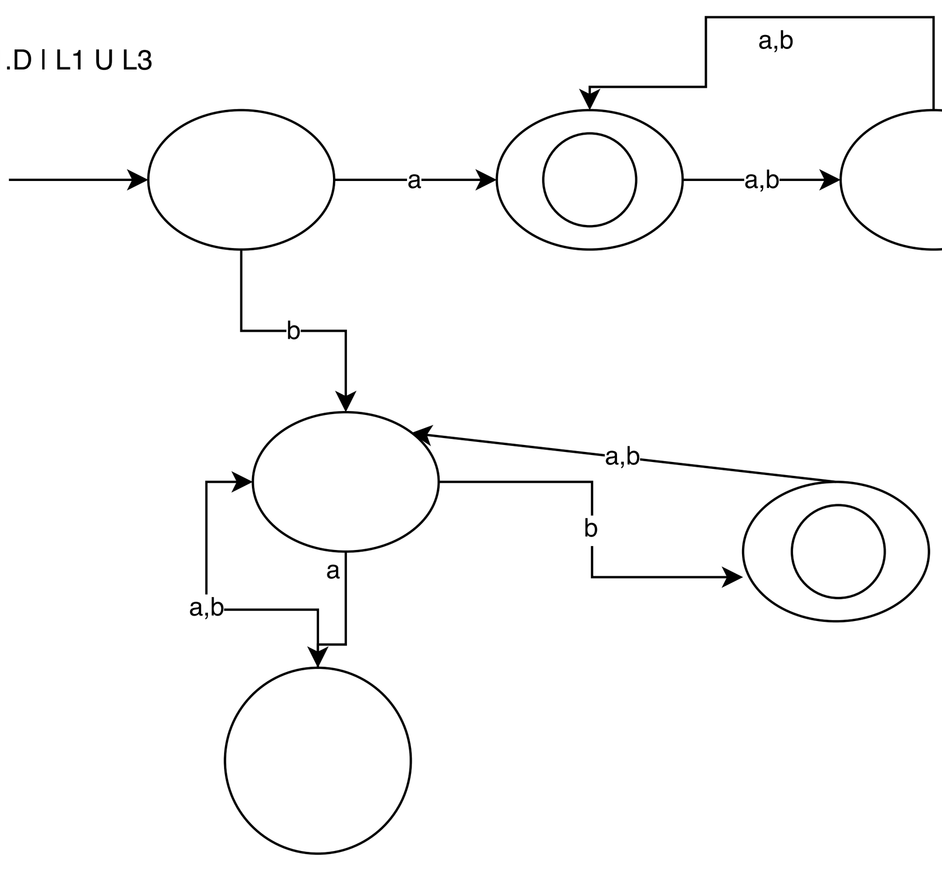
1.C



1.A

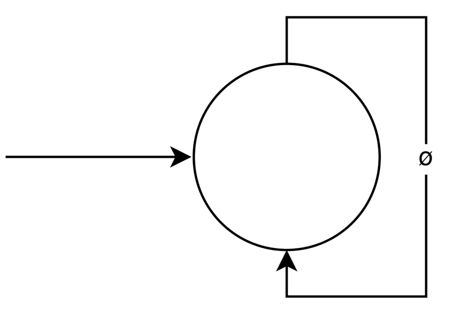


1.D l L1 U L3

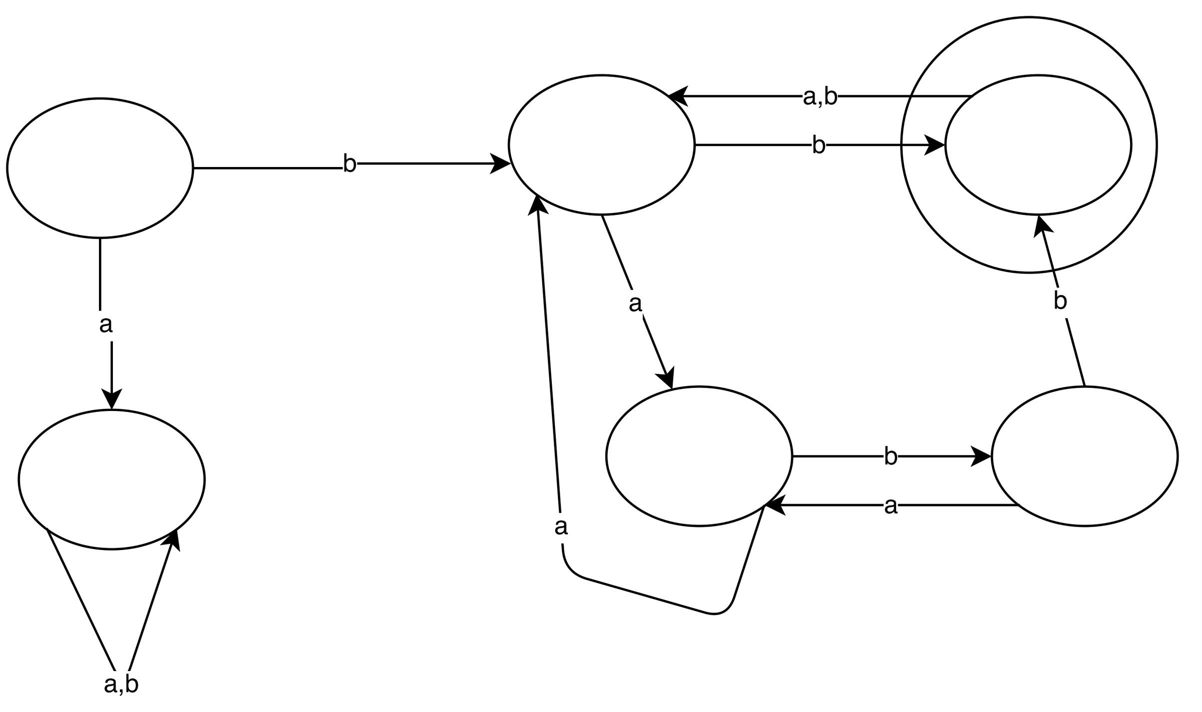


1.E Can't be even and odd.

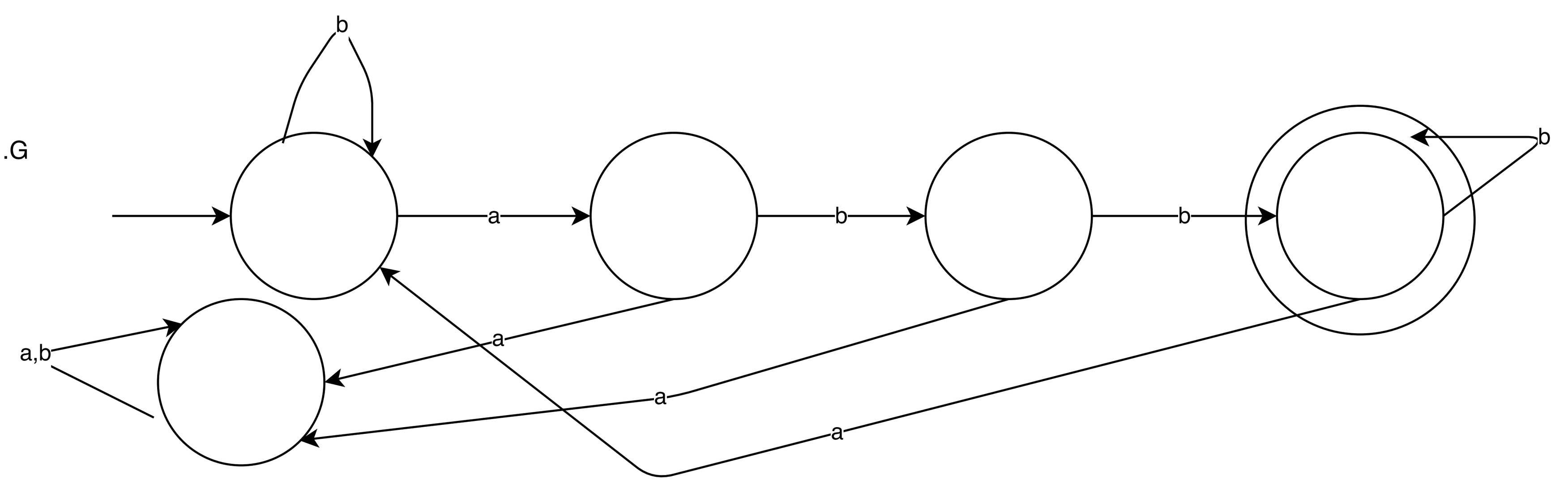
Empty Set



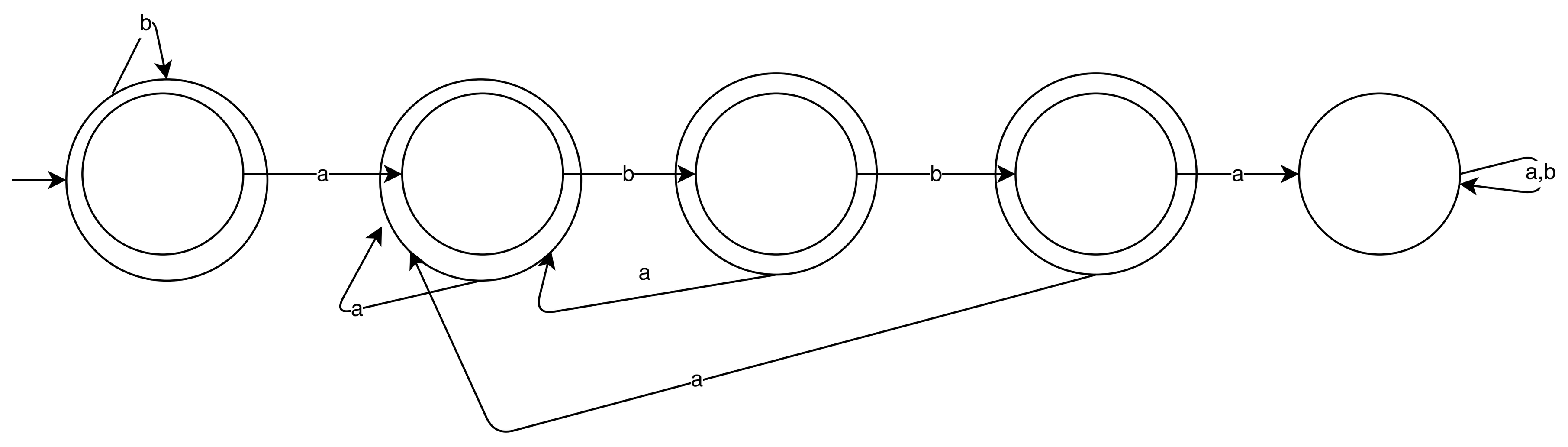
1.F



1.G



1.H

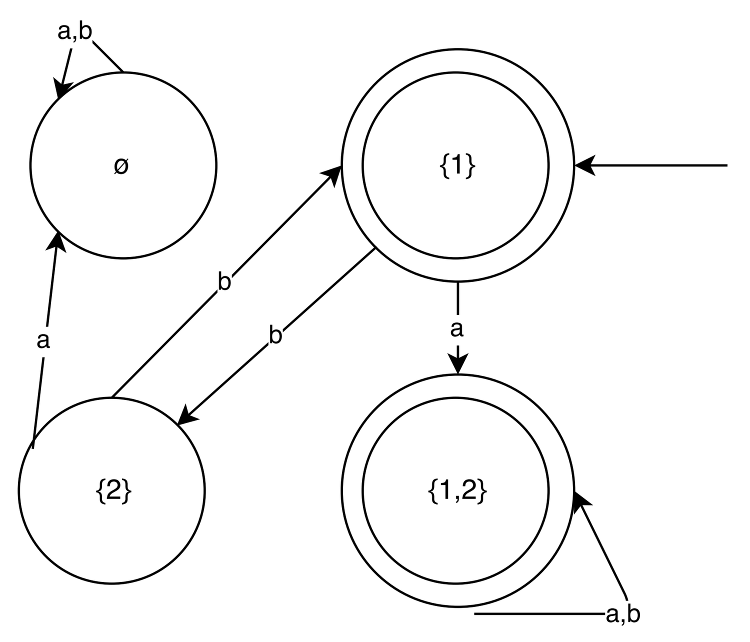


2.
- Every state M is a set of states of N.
- DFA transition function equals the union of all states of the NFA transition function.

2.A

NFA states = {1,2}
DFA states thus = {ø,{1},{2},{1,2}}

Next determine start and accept states of DFA.

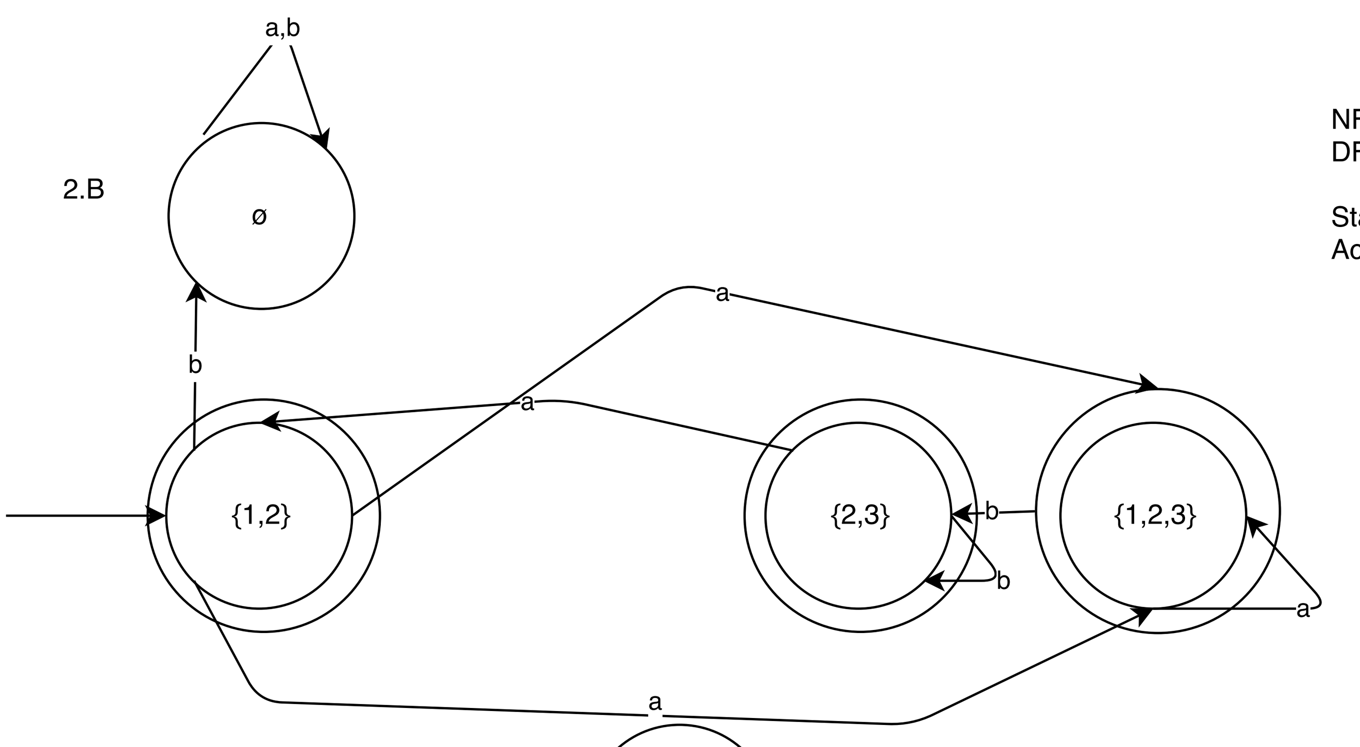


Start state is E({1}), since there are no empty sets the start state remains the same, as 1.
New Accept states are {{1},{1,2}}, since NFA accept state is {1}.

Now we determine transition function for the new DFA.
Each state goes from one place for input a, and one for input b.

Since 2 has no outgoing a it goes to the empty set.
State 2 has an outgoing b that goes to state 1.
both State 1 and 2 have incoming a's and b's thus it goes to itself.
State 1 has an outgoing a to itself and 2 thus it goes to the state {1,2}.
State 1 has an outgoing b to state 2.

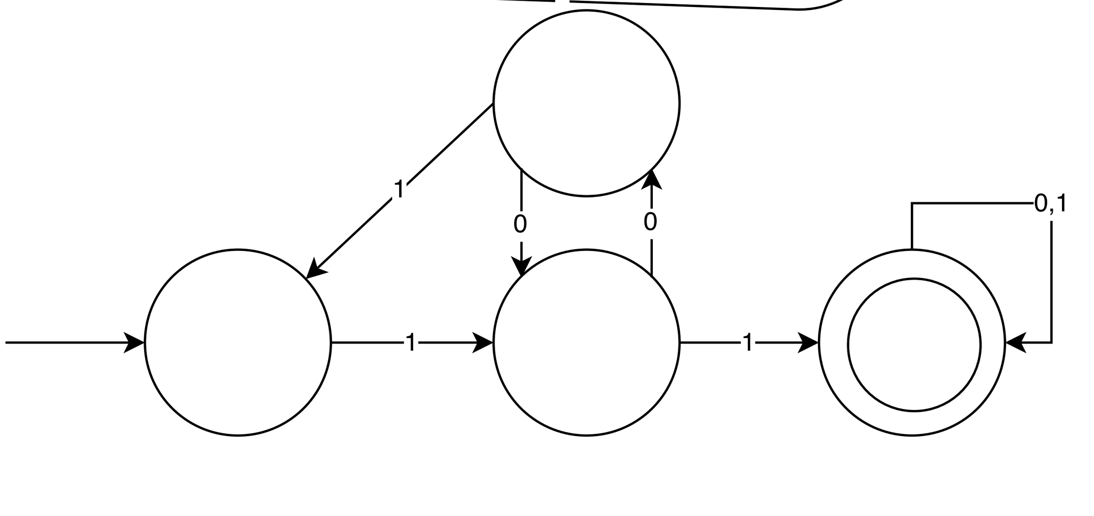
2.B



NFA states = {1,2,3}
DFA states = {ø,{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}}

Start state for NFA is {1}, but it has an empty set, thus the start state is {1,2}.
Accept state is 2, thus DFA accept states are {{2},{1,2},{2,3},{1,2,3}}.

3.



4. A
The formal definition for a regular language is the following:
The empty language ø, and the empty string {ε} are regular languages.
For each a ∈ Σ , the singleton language {a} is a regular language.
If A and B are regular languages, then A U B, A • B and A* are regular languages.
- All finite languages are regular.

For any language A, let A^R = {W^R l w ∈ A}. Let W = s1...sn and W^R = sn.....s1

If A is regular there exists a DFA M, such that A(M) = A.
We can show A^R is regular by constructing a DFA M' that exemplifies the relationship A and A^R hold, where A^R accepts all reversed strings that A accepts.

Let w^R = a'0a'1...a'n be a string over an alphabet A^R. The DFA M' accepts the state sequence r'0r'1...r'n if w is a string in DFA A.

1. Reverse the arrows of M using the transition function for M's arrows.
2. Make the start state an accepting state.
3. create a start state that points to the old accept states with pointers having empty string.

Now that A^R has a DFA M', it is a regular language.

4.B

B(M) = M, M is a DFA of B
B^R(M') = B^R, M' is a DFA of B^R

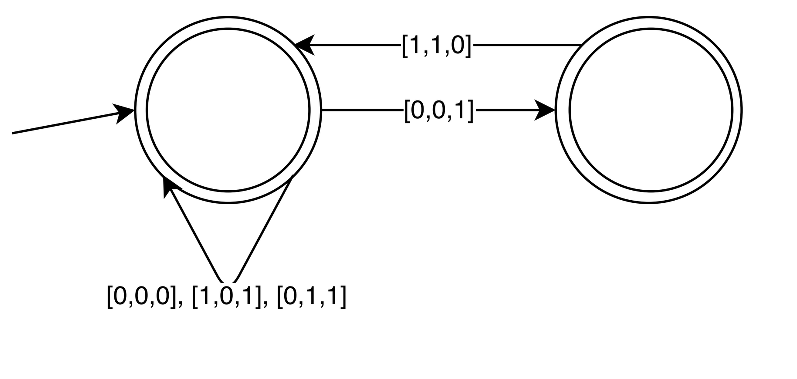
From the prior proof 4.A, we can see that if A is regular, A^R is regular. Thus, the opposite is true, if A^R is regular, A is regular.

If B^R has a DFA M', then it is regular.

In order for M' to be a valid DFA it needs a start state, transition function, and set of accept states.

Using the DFA description of M' in 4.A, we can see that M' will follow the same characteristics.

Now that B^R has a DFA M', B is a regular language.



Since this NFA Exists we can say that B is regular.