NFA and Kleene’s Theorem

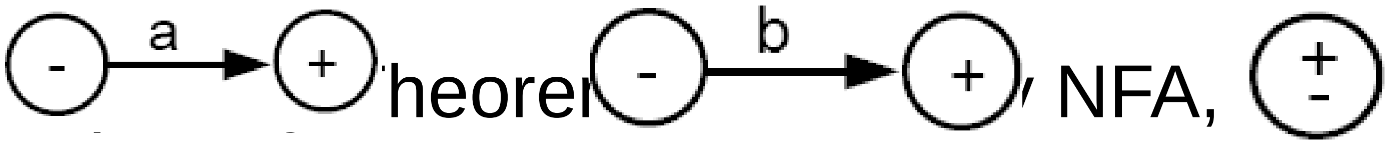
Shakir Ullah Shah

PROOF 2 OF RULE 1, PART 3, THEOREM 6

Rule 1 states that there are FAs for the languages {a}, {b}, and {Λ}.

Proof:

Step 1: The above three languages can all be accepted respectively by the NFAs below:



S[illegible]heore[illegible] NFA, [illegible]e is an equivalent FA. Hence, there must be FAs for these three languages as well.

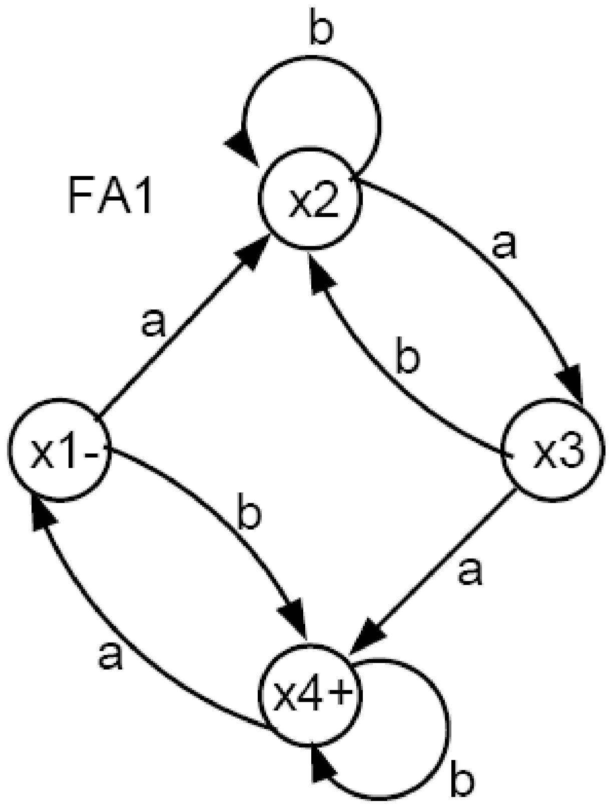
NFA corresponding to Union of FAs

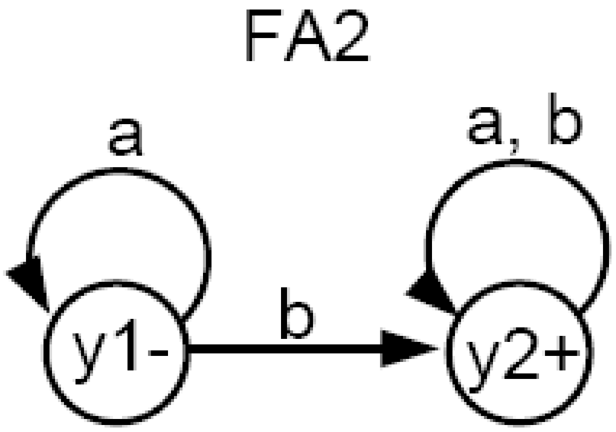
NFA corresponding to Union of FAs

- **Method:**
  Introduce a **new start state** and connect it with the states originally connected with the old start state with the same transitions as the old start state, then remove the –ve sign of old start state. This creates non-determinism and hence results in an NFA.

EXAMPLE

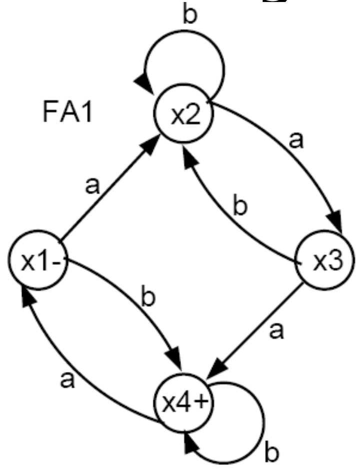
Consider the FA_1 and FA_2 below:

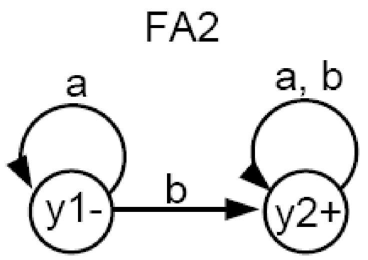


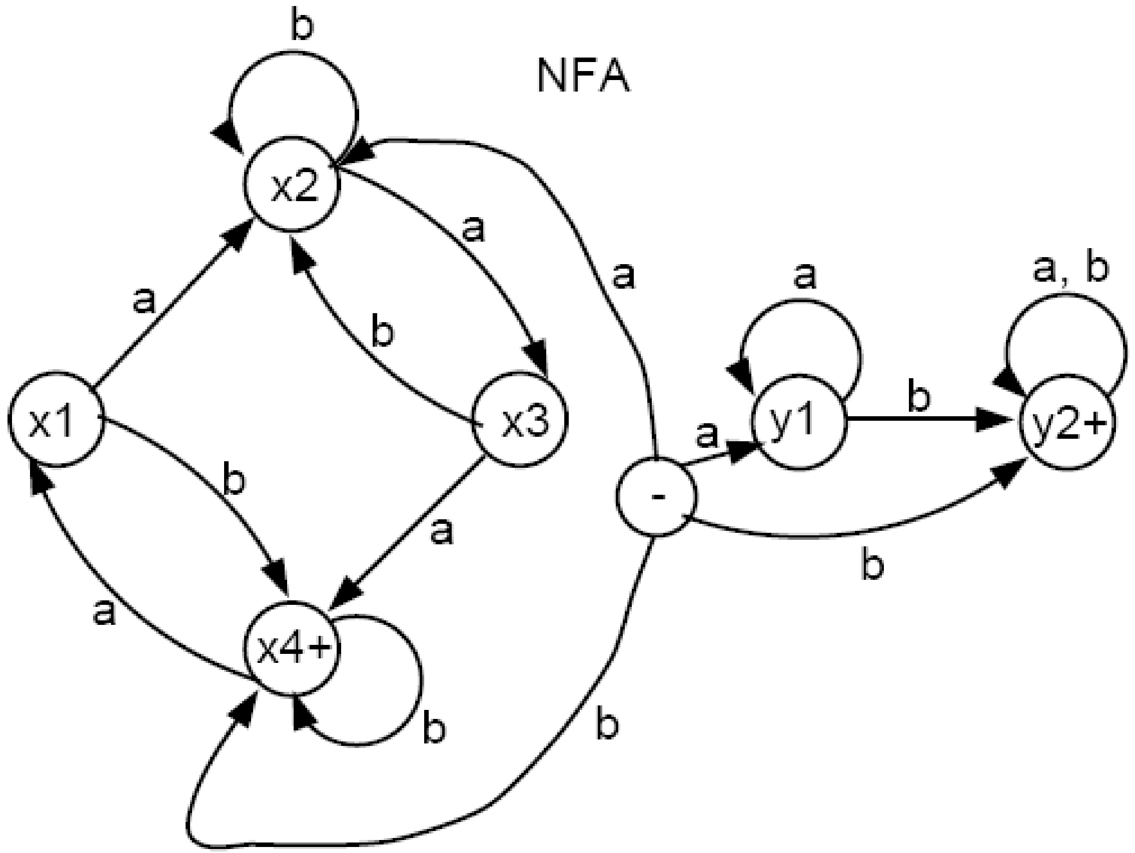


EXAMPLE

Using the above algorithm (Step 1) we produce the following NFA.

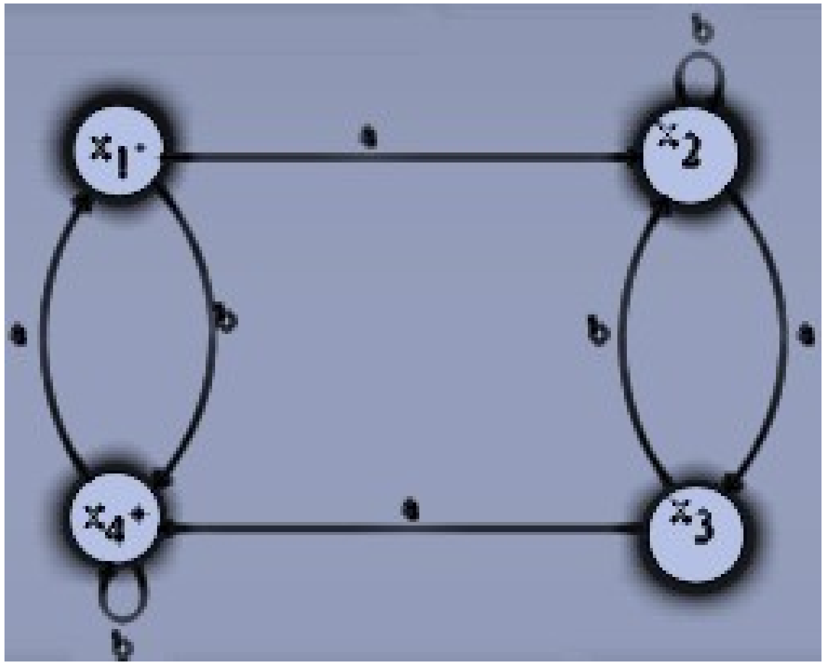




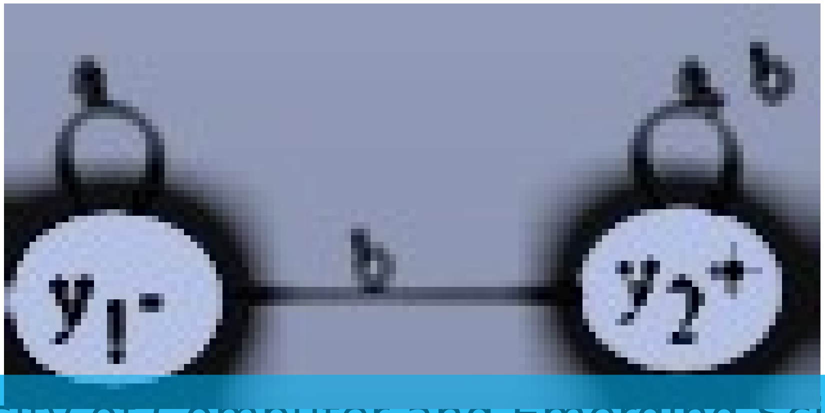


Example

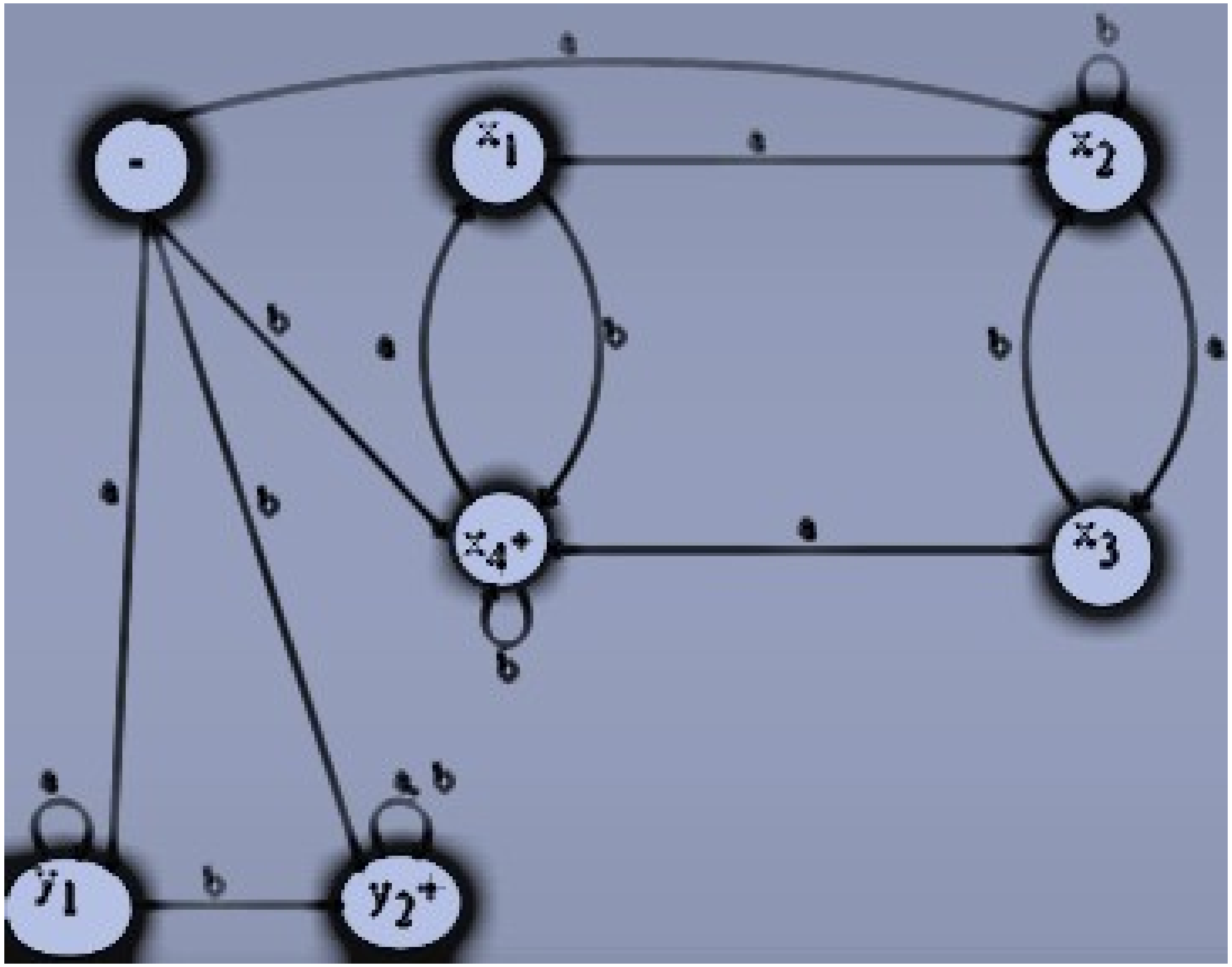
- FA1



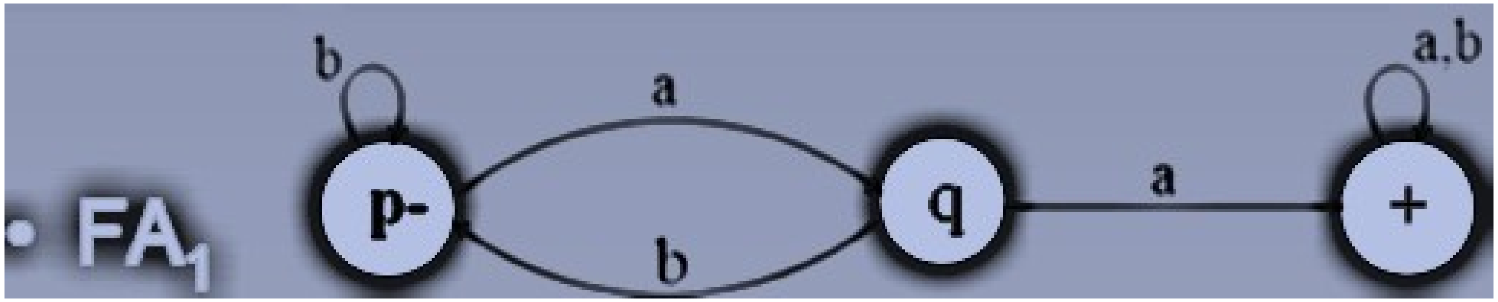
- FA2

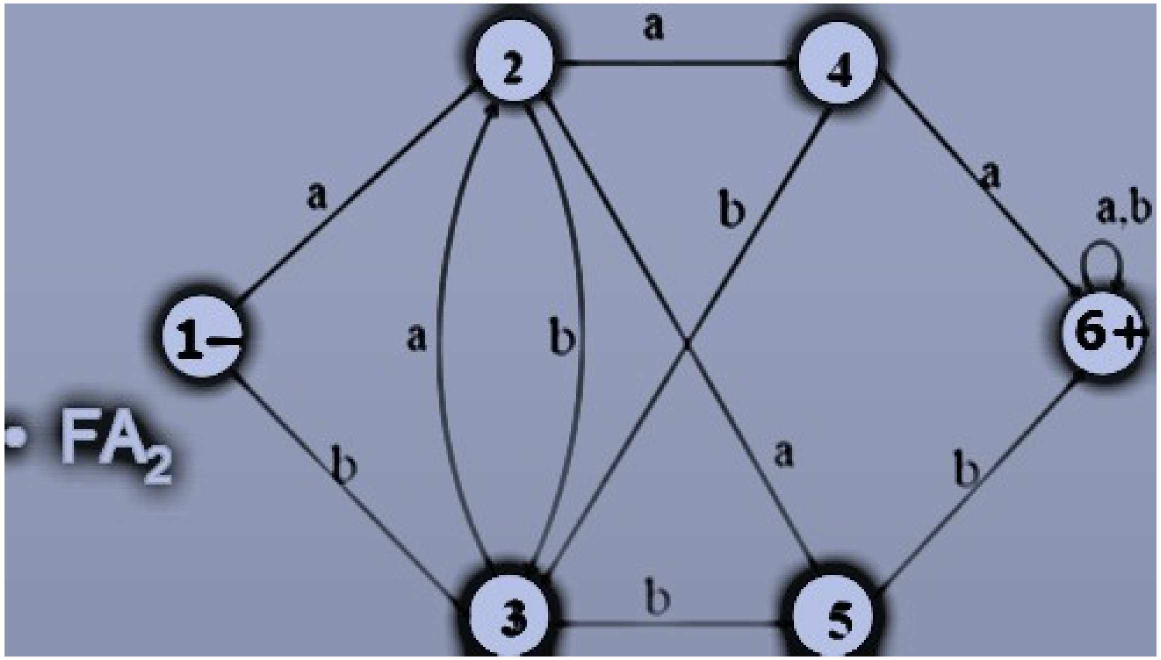


NFA equivalent to FA_1 U FA_2

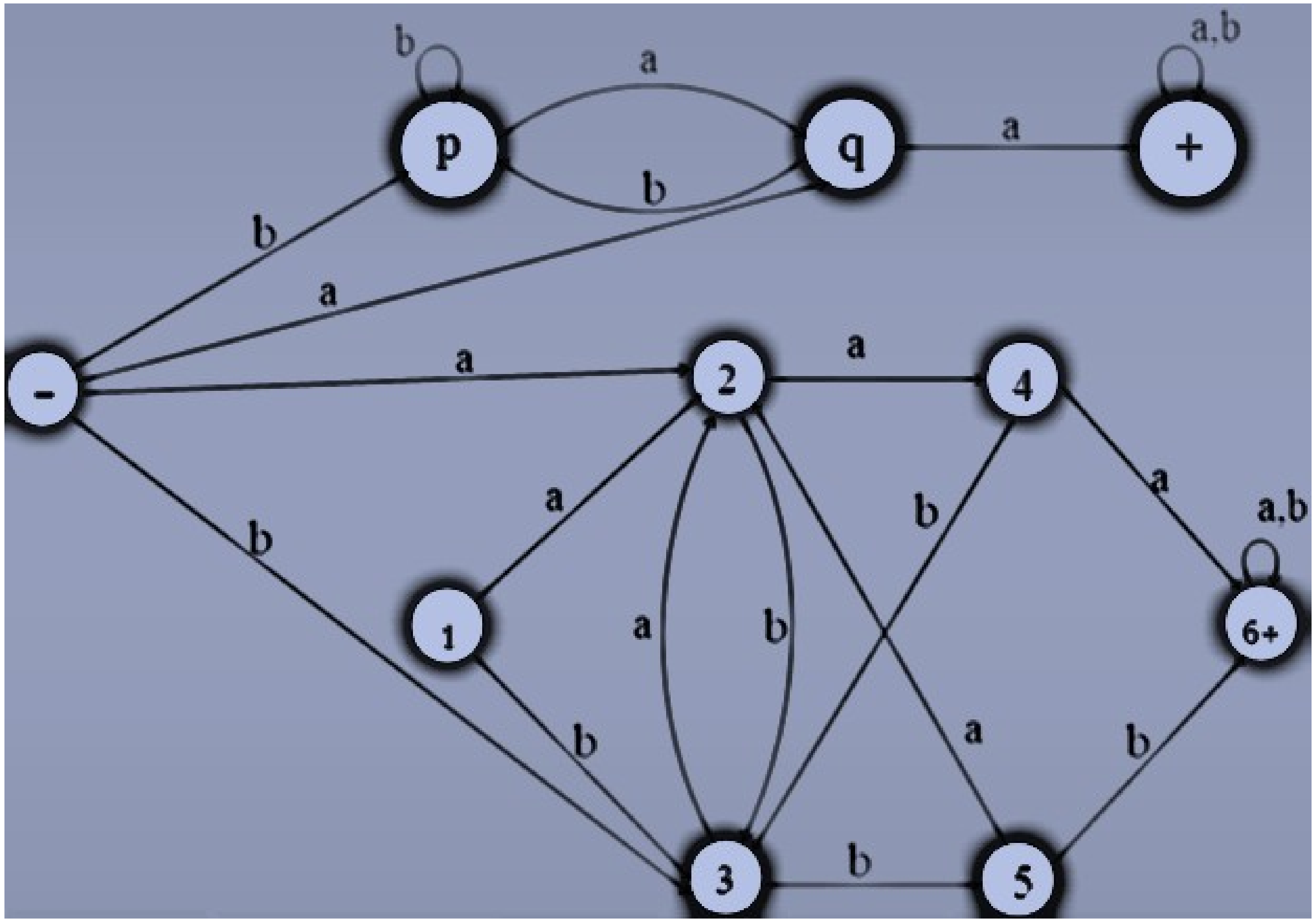


Example





NFA equivalent to $FA_1 \cup FA_2$



NFA corresponding to Concatenation of FAs

NFA corresponding to Concatenation of FAs

Method:

- Introduce additional transitions for each letter connecting each final state of the first FA with the states of second FA that are connected with the initial state of second FA corresponding to each letter of the alphabet.
- Remove the +ve sign of each of final states of first FA and –ve sign of the initial state of second FA. It

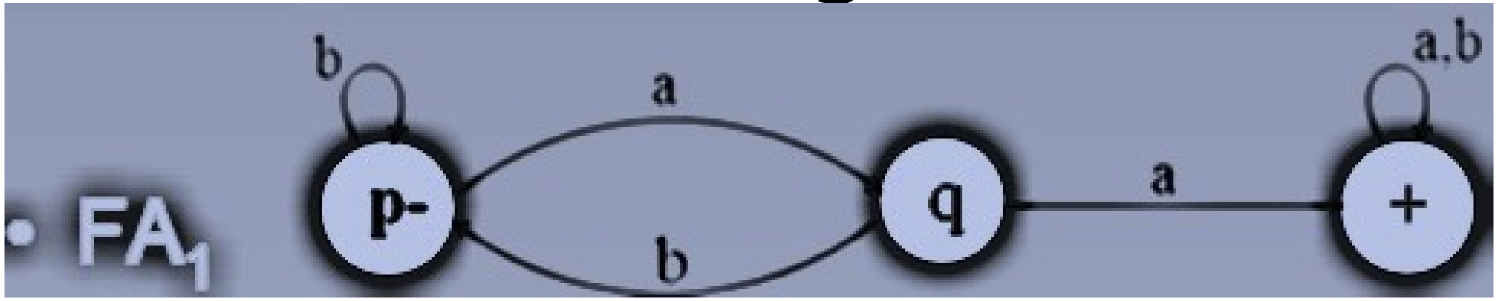
NFA corresponding to Concatenation of FAs

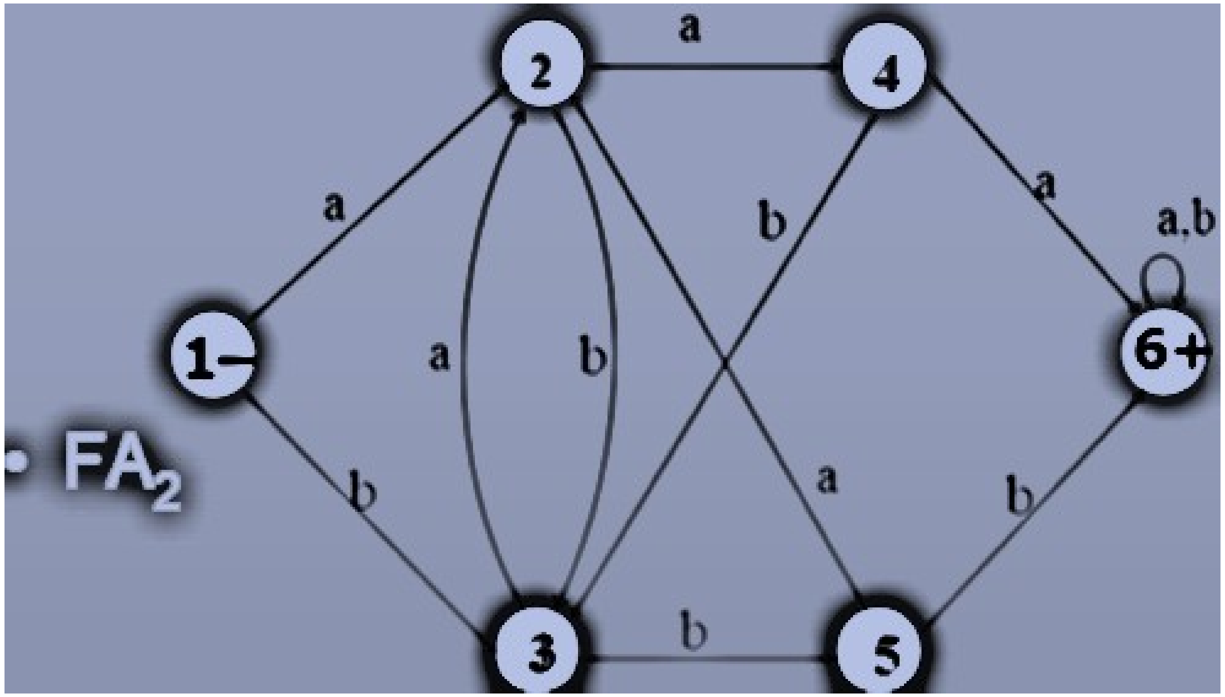
- **Note:**
  If first FA accepts the Null string then every string accepted by second FA must be accepted by the concatenation of FAs as well. This situation will automatically be accommodated using the method discussed earlier.
  However if the second FA accepts Null string, then every string accepted by first FA must be accepted by the required FA as well. This target can be achieved as , while introducing new

Note Continued …

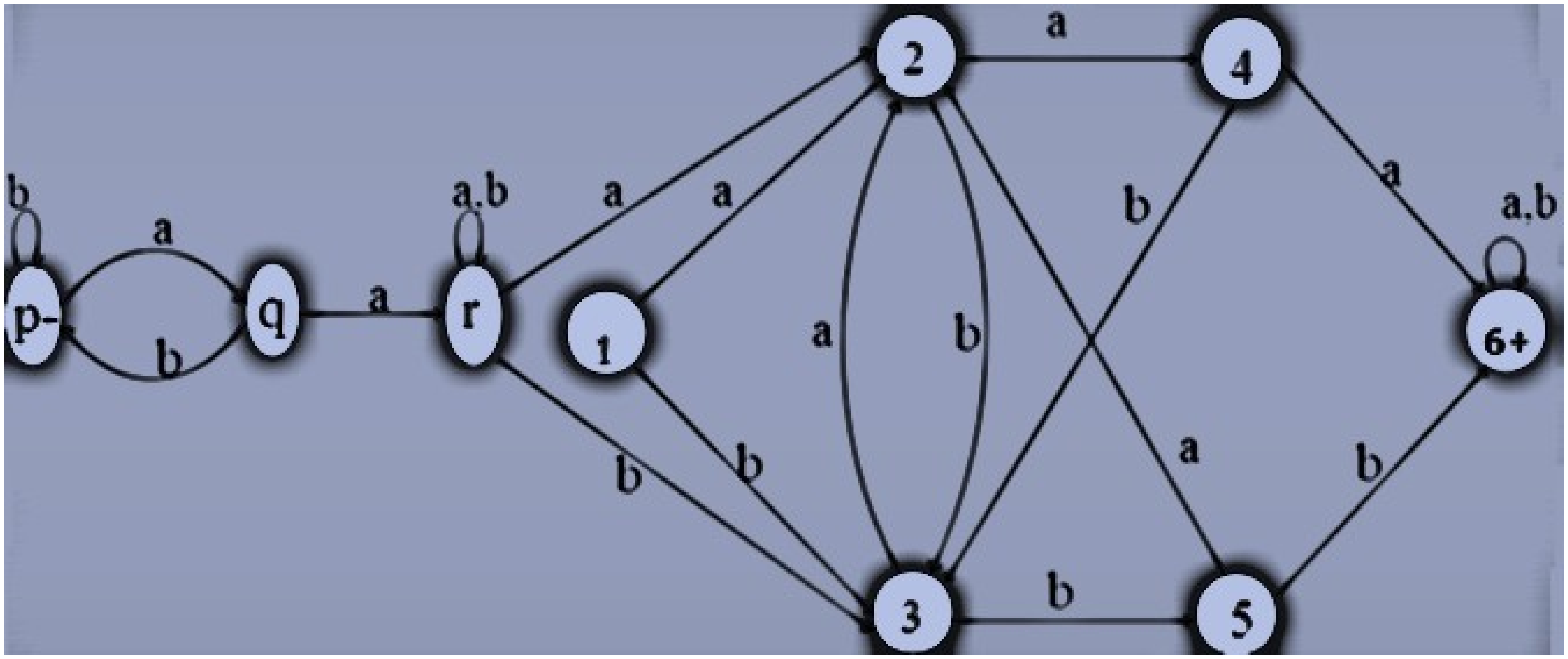
- Lastly if both FAs accepts the Null string, then the Null string must be accepted by the required FA. This situation will automatically be accommodated as the second FA accepts the Null string and hence the +ve signs of final states of first FA will not be removed.

Example (No FA accepts Null string)

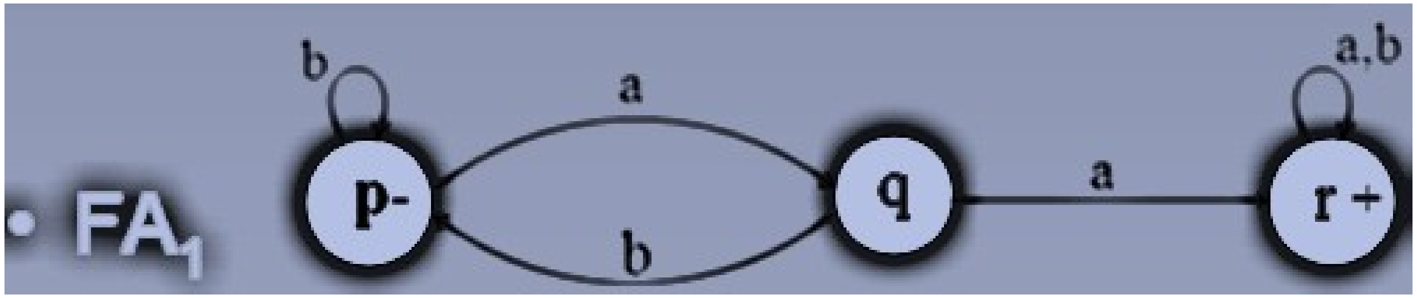


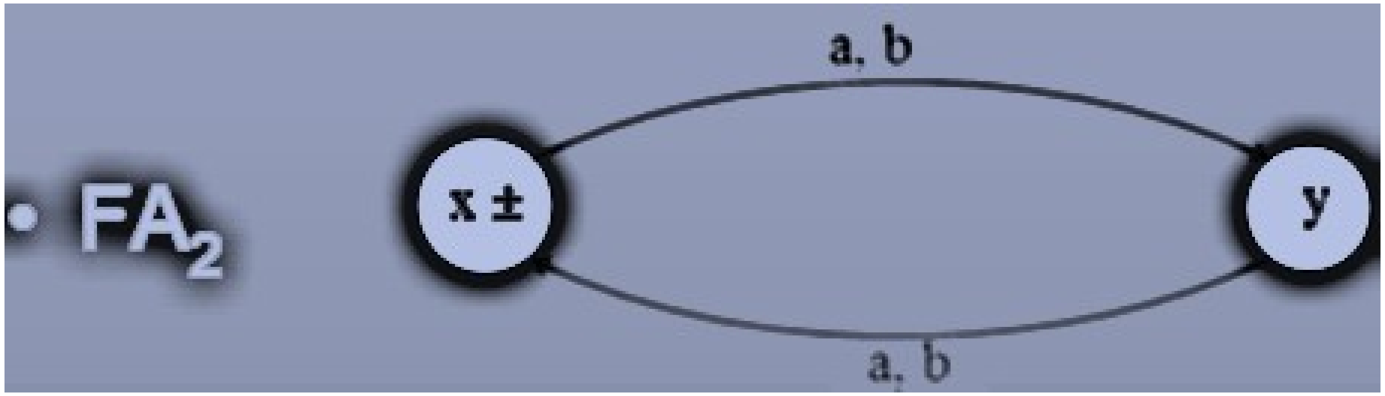


NFA equivalent to FA_1FA_2

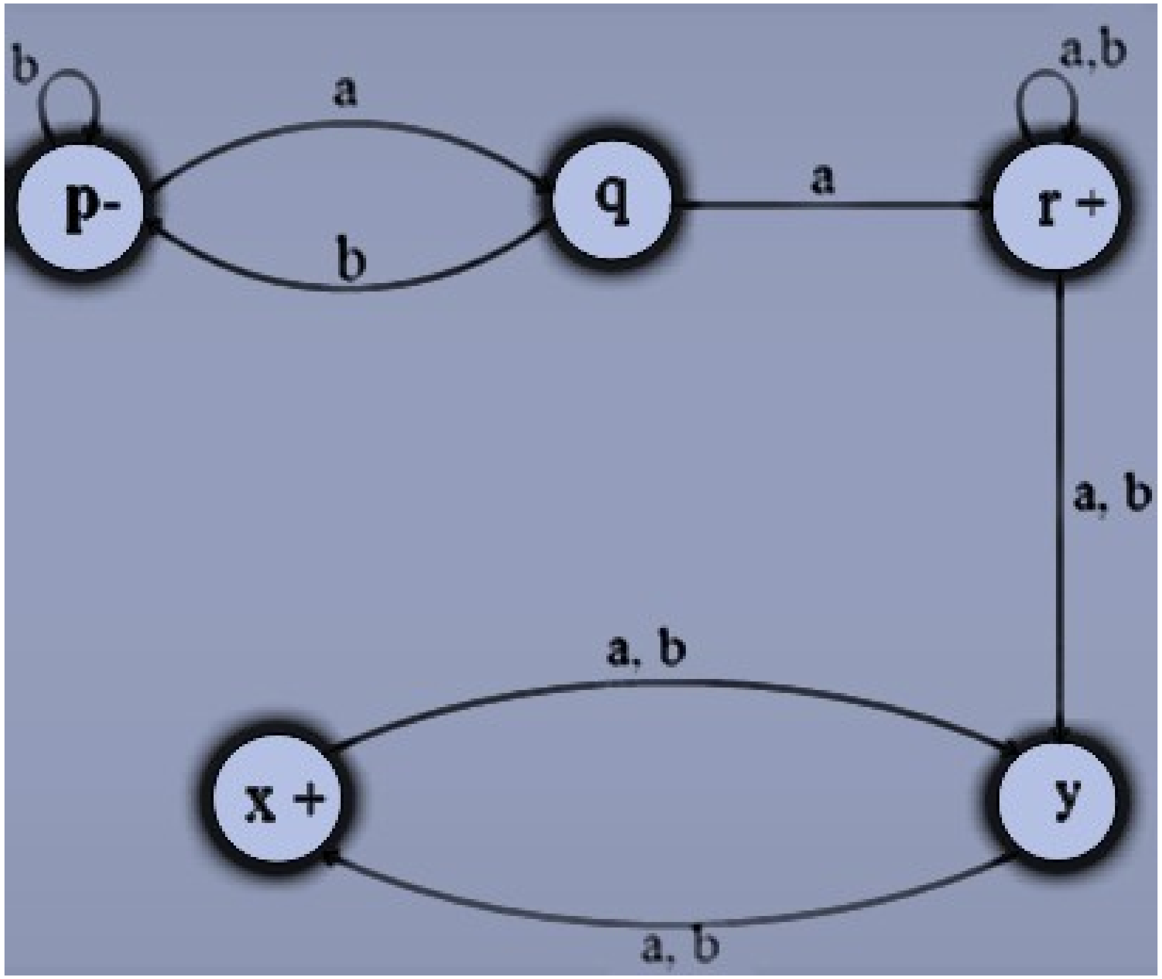


Example (FA_2 accepts Null string)



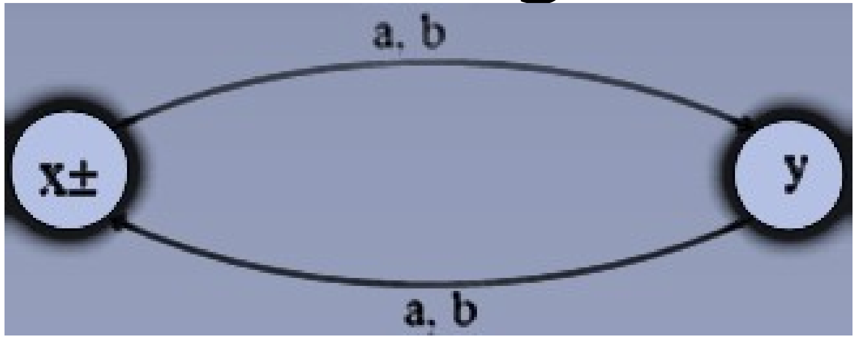


NFA equivalent to FA_1FA_2

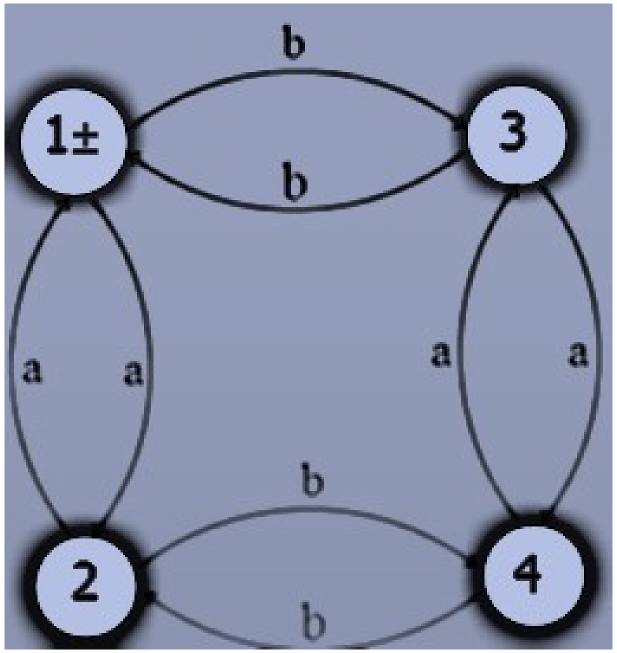


Example (Both FAs accept Null string)

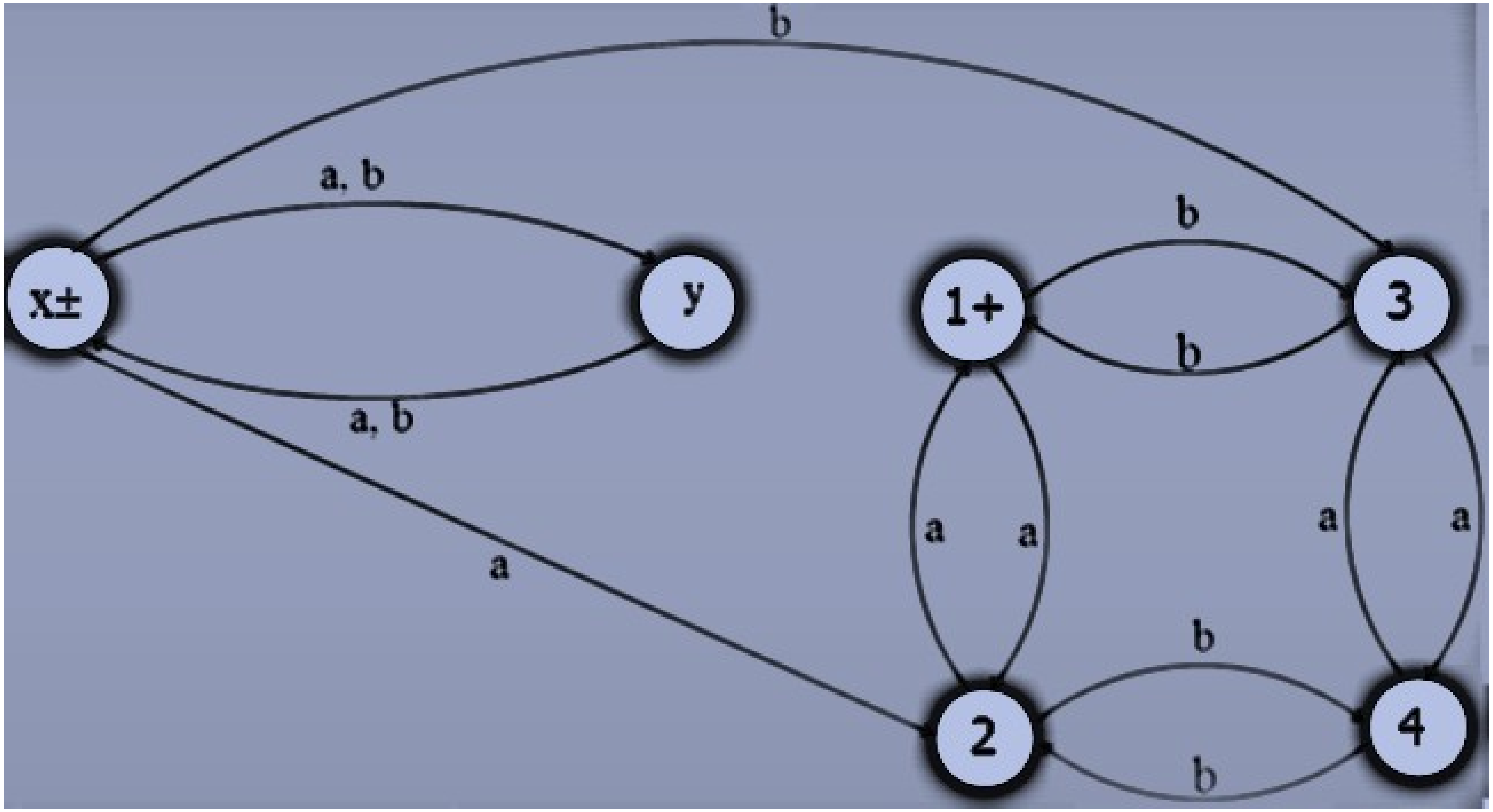
- FA$_1$



- FA$_2$



NFA equivalent to FA_1FA_2



NFA corresponding to the Closure of an FA

NFA corresponding to the Closure of an FA

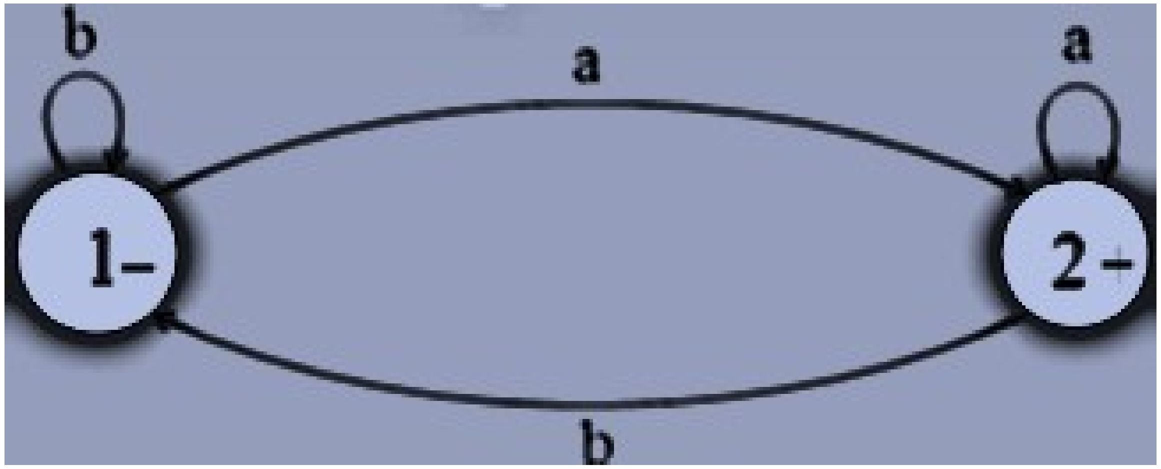
- Apparently, it seems that since closure of an FA accepts the Null string, so the required NFA may be obtained considering the initial state of given FA to be final as well, but this may allow the unwanted string to be accepted as well. For example, an FA, with two states, accepting the language of strings, defined over. $\Sigma = \{a, b\}$, **ending in a**, will accept all

NFA for the Closure of an FA Continued …

- **Method:**
  Thus, to accommodate this situation, introduce an initial state which should be final as well (so that the Null string is accepted) and connect it with the states originally connected with the old start state with the same transitions as the old start state, then remove the –ve sign of old start state.
- Introduce new transitions, for each letter, at each of the final states (including new final state) with those

Example

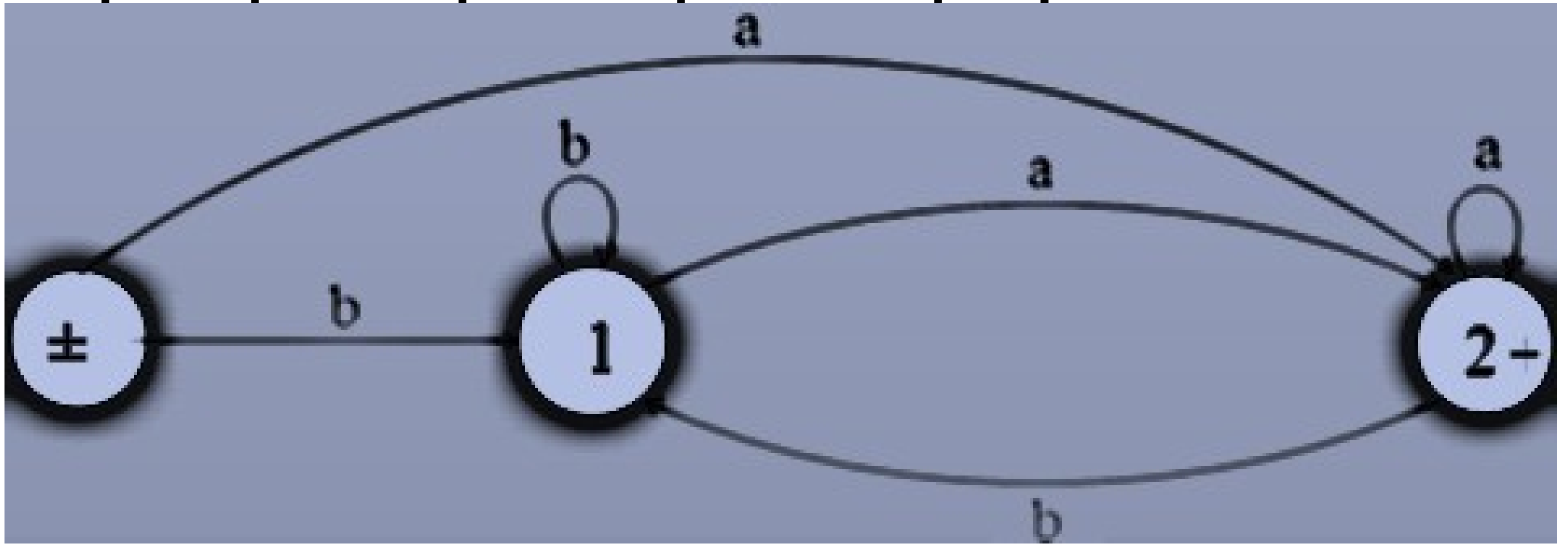
- Consider the following FA



- It may be observed that the FA* accepts only the additional string which is the Null string

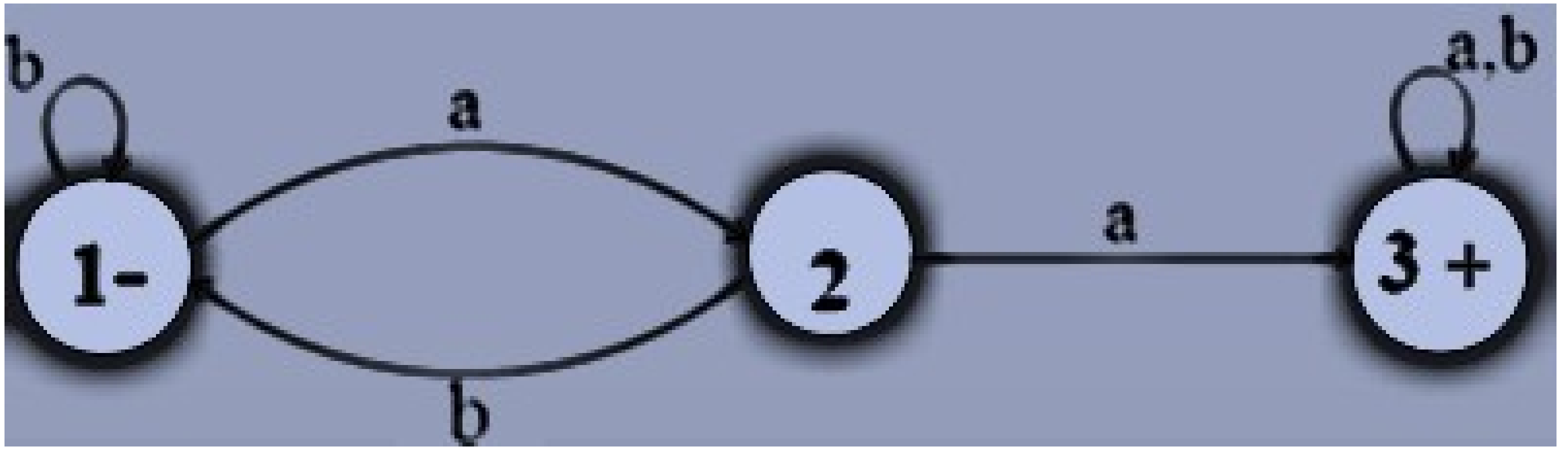
Example Continued …

- As observed in the previous example the required NFA can be constructed only if the new initial state is



Example

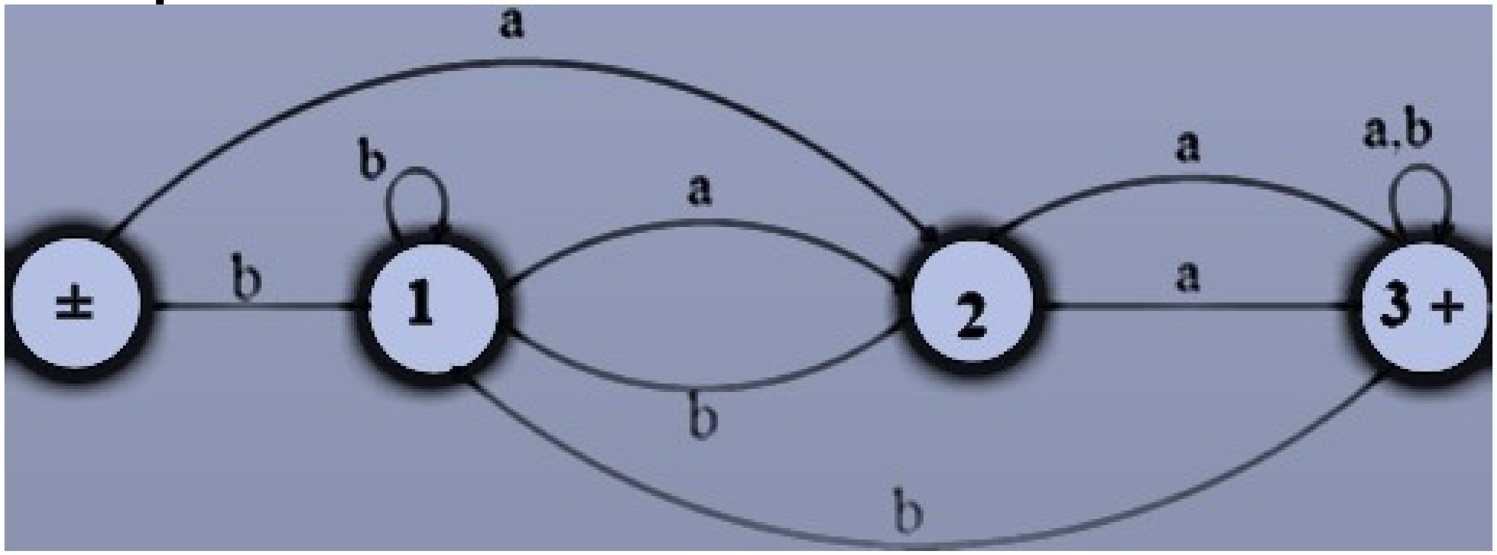
- Consider the following FA



It may be observed that the FA* accepts only the additional string which is the Null string.

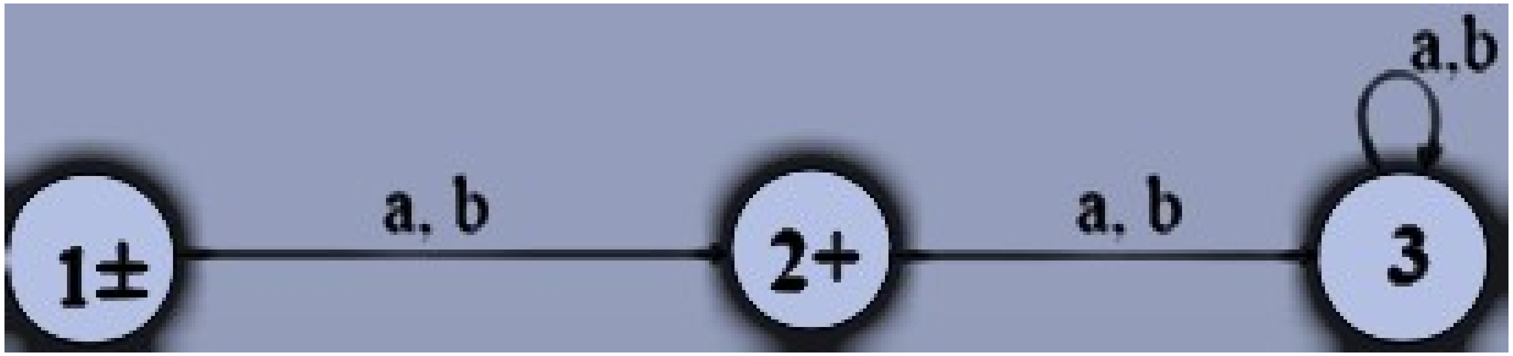
Example Continued …

- Considering the state 1 to be final as well, will allow the unwanted strings be accepted as well. Hence the required NFA is constructed



Example

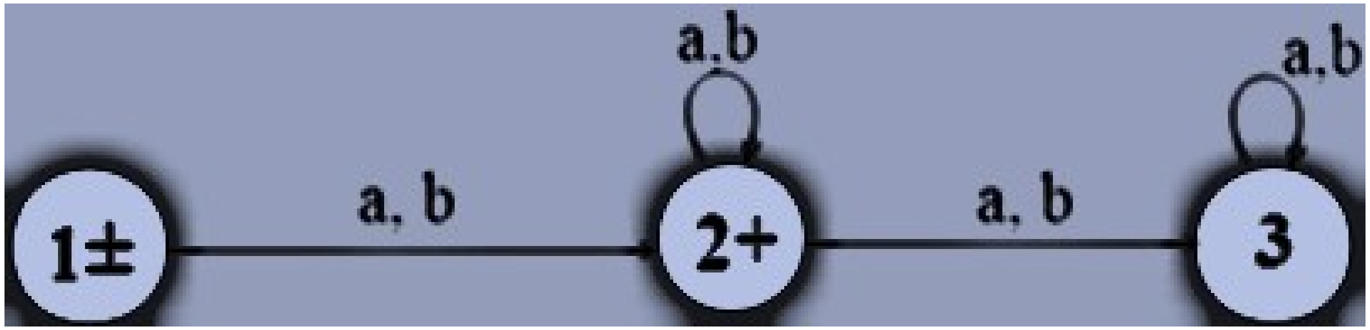
- Consider the following FA



- It can be observed that FA* not only accepts the Null string but every other string as well.

Example continued …

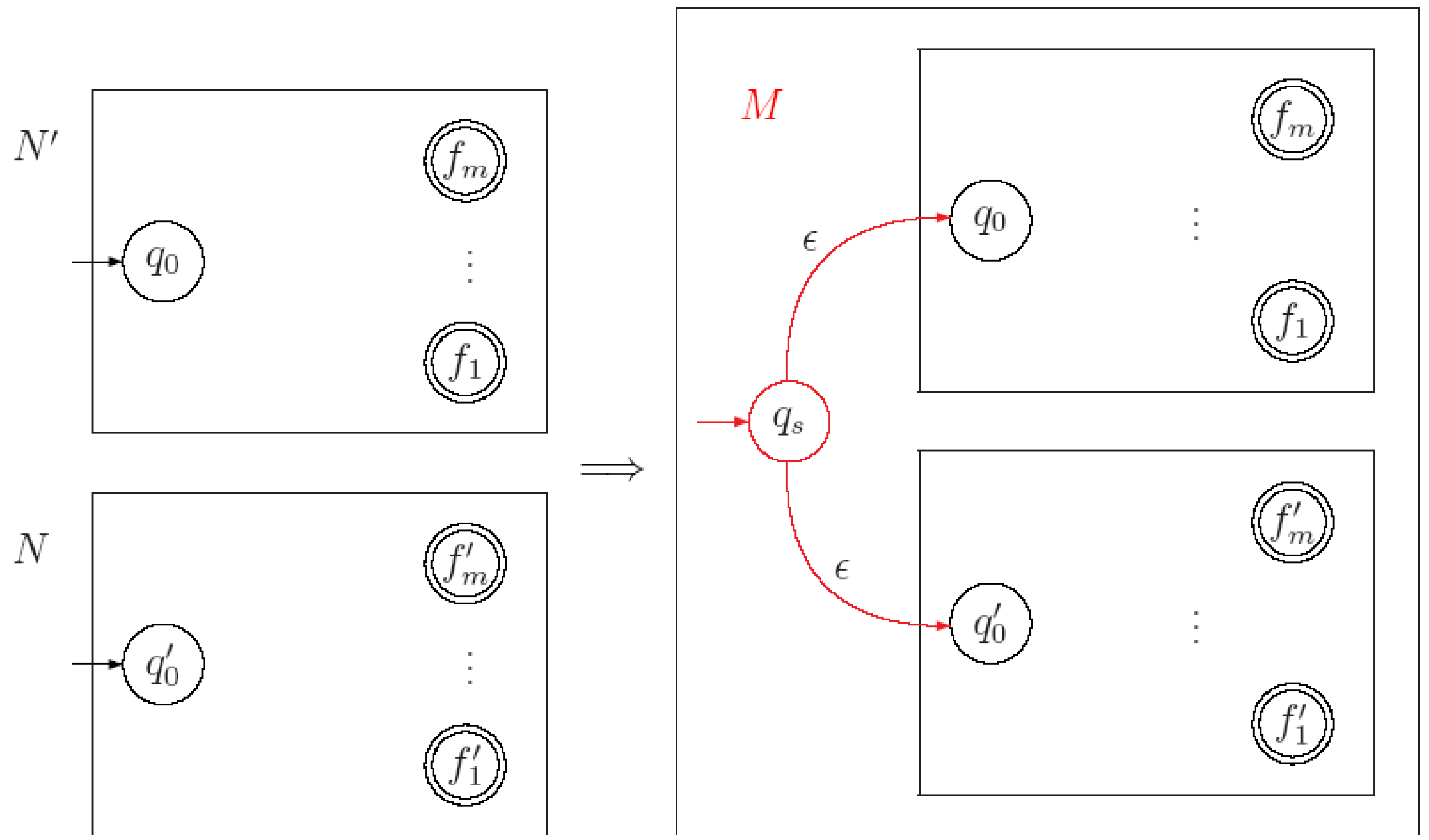
- Here we don’t need separate initial and final state. Hence an NFA corresponding to FA* may be



NFA closure under union

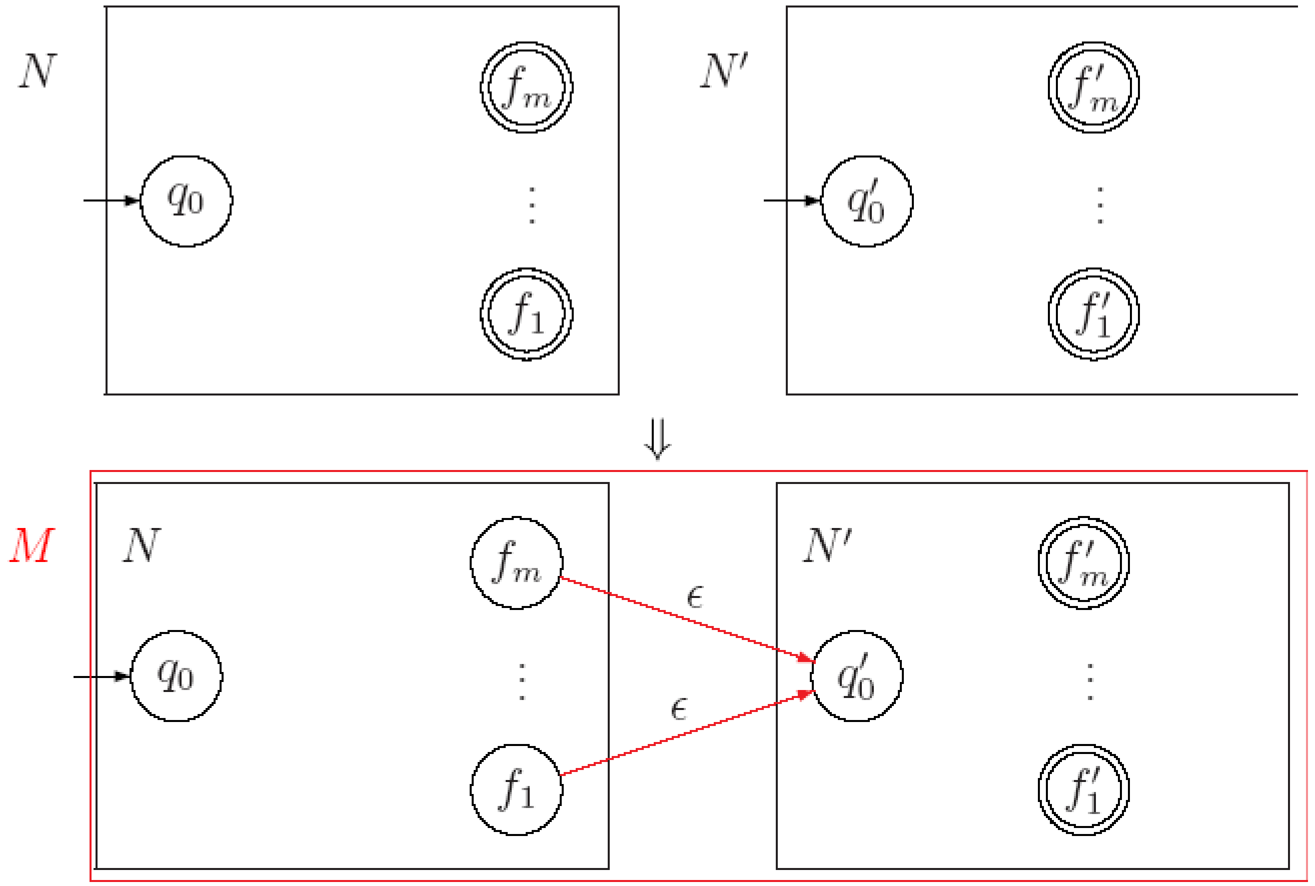
- Given two NFAs, say $N^{/}$ and N, we would like to build an NFA for ∪the language ($N^{/}$)   L (N).
- The idea is to create a new initial state qs and connect it with a ε -transition to the two initial states of $N^{/}$and N. Visually, the resulting NFA M looks as follows.

NFA closure under union



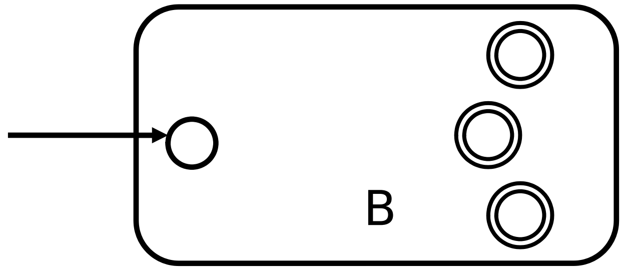
NFA closure under concatenation

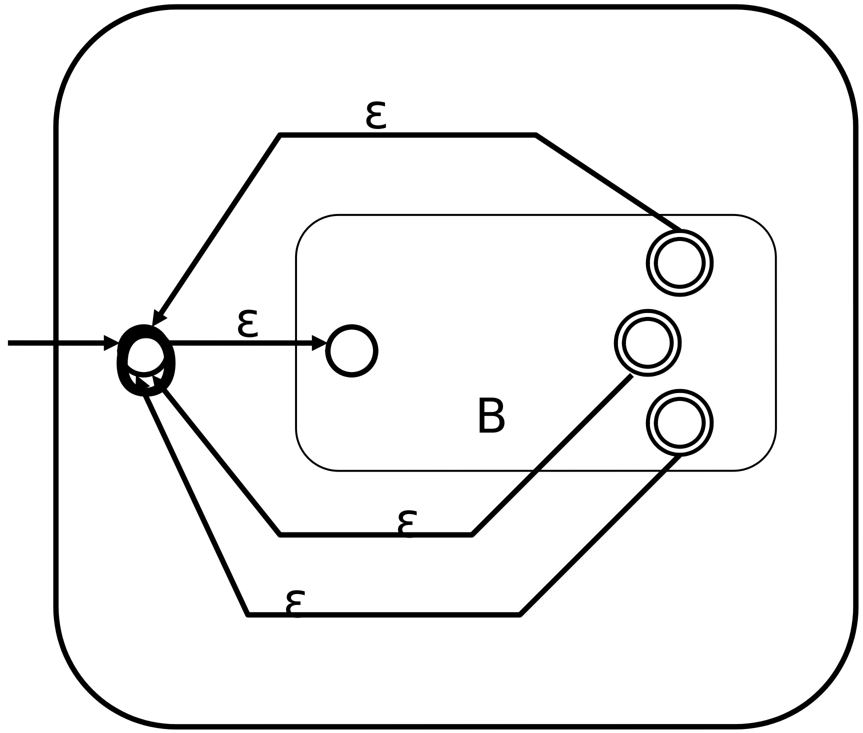
Add Λ-transitions from all final states of first one NFA to the start state of second NFA.
Then declare the start state (of the new automata) to be the start state of first one NFA.
The final states (of the new automata) to be the final states of second one NFA.



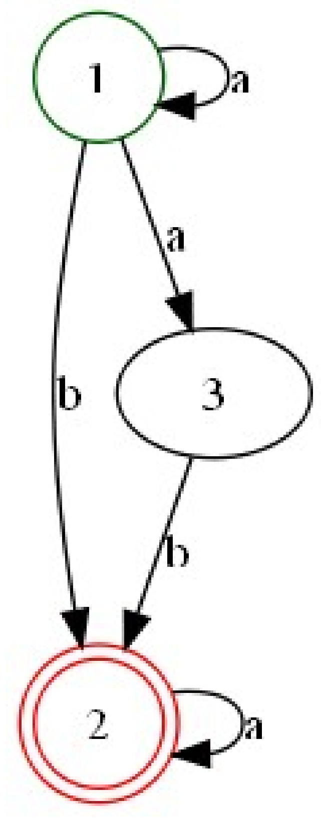
N
f_m
q_0
f_1
N'
f'_m
q'_0
f'_1
⇓
M
N
f_m
q_0
f_1
ϵ
ϵ
N'
f'_m
q'_0
f'_1

NFA closure under star

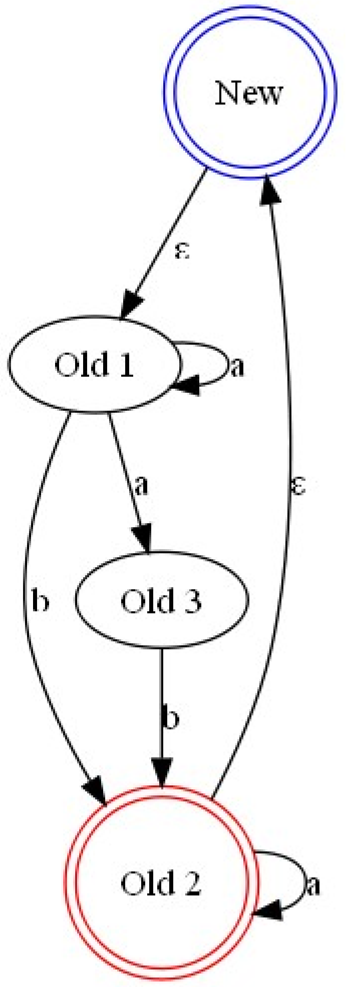




Example



- Add a new state.
- Make it the start state in the new NFA, and an accepting state.
- Add an ε-arc from this state to the old start state.
- Add ε-arcs from every final state to this new state



New
ε
Old 1
a
a
ε
b
Old 3
b
Old 2
a