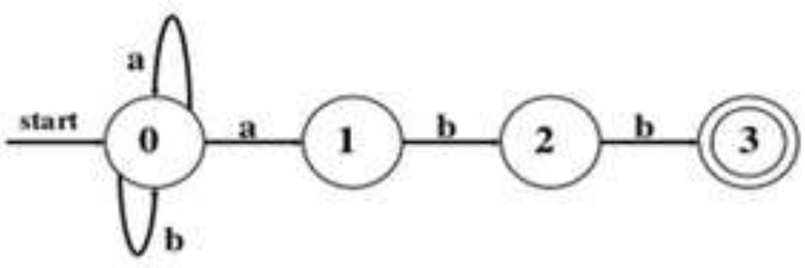
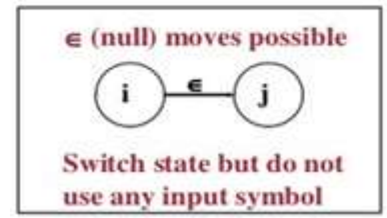
Finite State Automata

- FSA – Type 3 Grammar – Regular language
- Recognizer → input → x belongs to L →yes/no
- Regular Expression is compiled into a ***Recognizer*** by constructing *Transition diagram called Finite Automaton*
- Types:
  - Deterministic Finite State Automata (DFA)
  - Non-Deterministic Finite State Automata (NFA)
  - **ε** -Non-Deterministic Finite State Automata (**ε -** NFA)

FSA – NFA Example



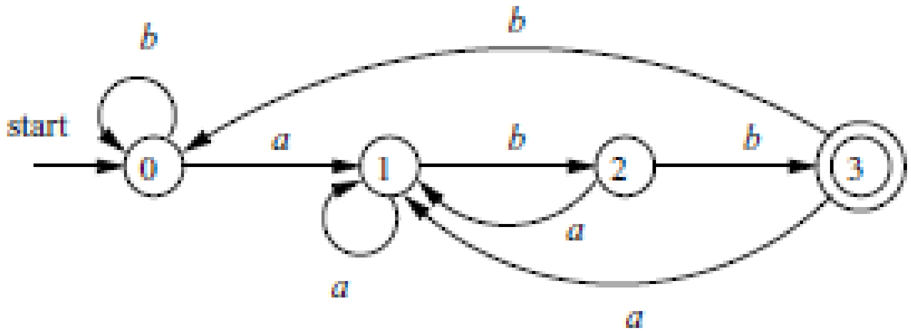


- Regular Expression→(a|b)* abb
- {Q,∑,qo,t,F}
- Q →0,1,2,3
- Inputs→a,b
- q0 →0
- F→3

	a	b
0	0,1	0
1	-	2
2	-	3
3	-	-

FSA – DFA *(a|b)*abb* Example

- Epsilon is not accepted
- For each state s and input symbol a, there is exactly one edge out of s labeled a



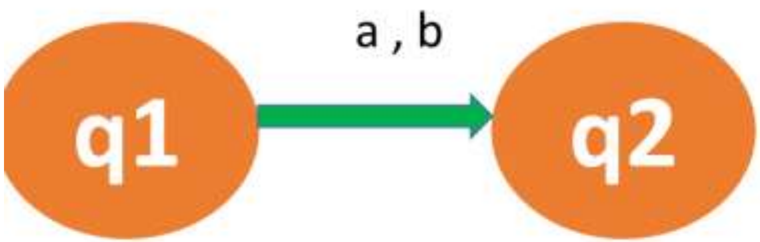
FSA

- Set of strings over {a,b} which ends with bb
- RL = {bb,abb,ababb,bbb,abbb,abababb,….}
- RE = (a+b)*bb
- **Upcoming Topics**

1. Regular expression to Automata
2. Regular expression to **ε-**NFA
3. ε-NFA to DFA
4. Direct conversion of Regular Expression to DFA

1. Regular Expression to Automata

- a+b (or)a|b



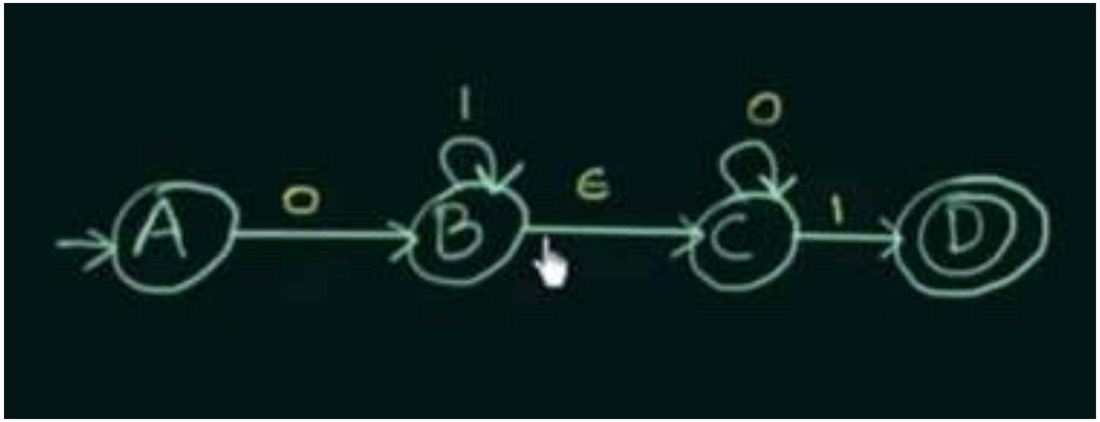
- ab

- a*

2.Regular Expression to NFA

ε-NFA

- **ε -** Empty
- $\{Q, \sum, qo, \delta, F\}$
- $\delta \rightarrow$ NFA $\rightarrow Q x \sum \rightarrow 2^Q$
- $\delta \rightarrow$ **ε**-NFA $\rightarrow Q x \sum U$ **ε** $\rightarrow 2^Q$
- <u>Example: 2 states $\rightarrow 2^2$transitions</u>

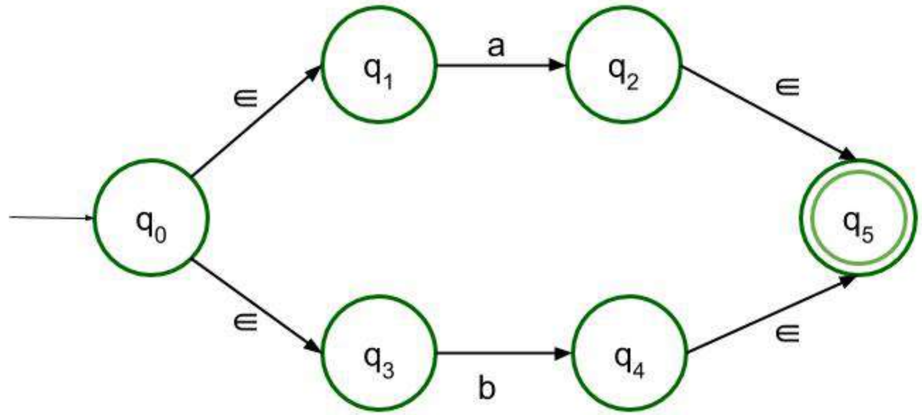


- Every state on **ε** goes to itself

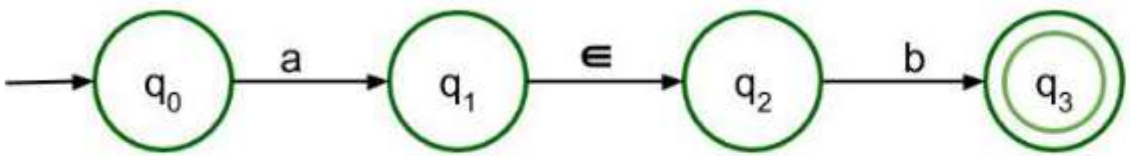
2.Regular expression to ε-NFA

Designing NFA using Thompson's Construction

- **(a+b)** OR **a|b**

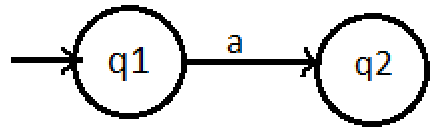


- **ab**

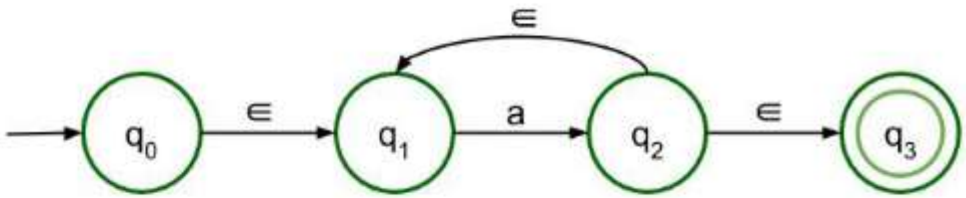


2. Regular expression to ε-NFA

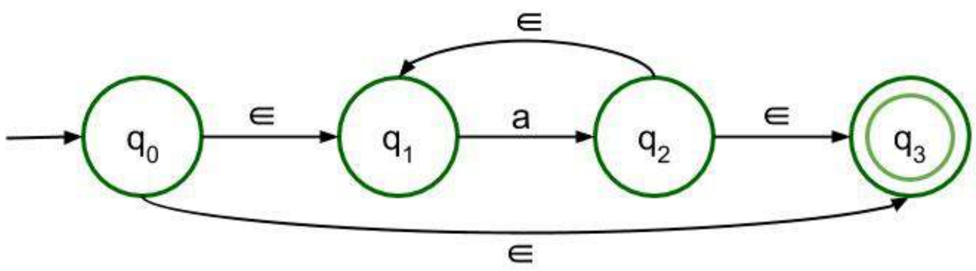
- **a**

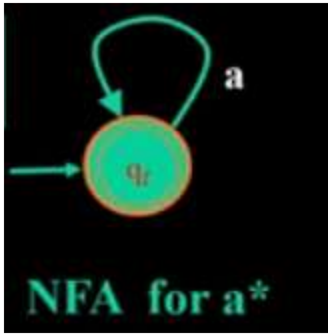


- **a⁺**

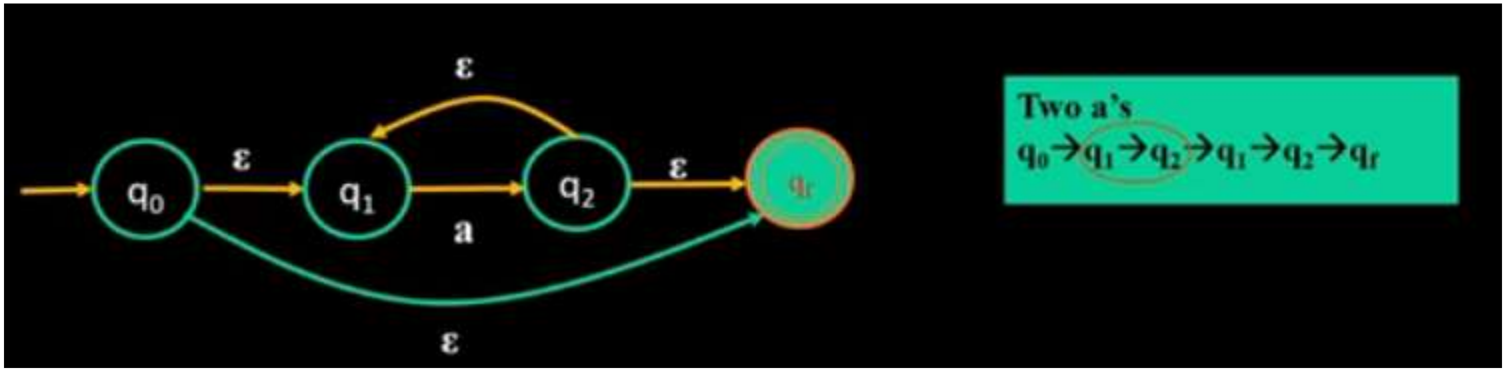


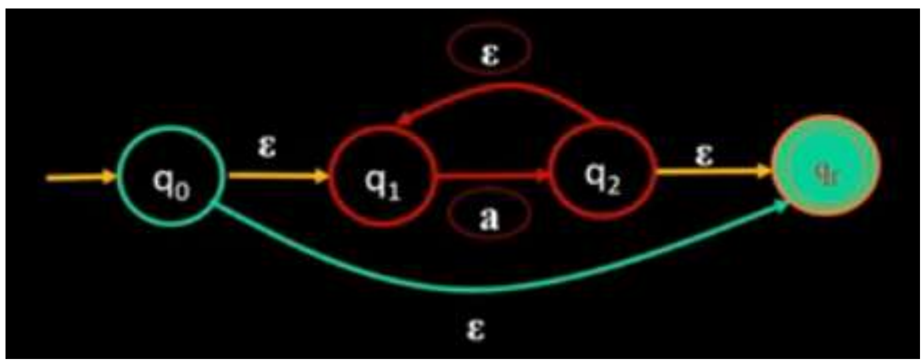
- **a***





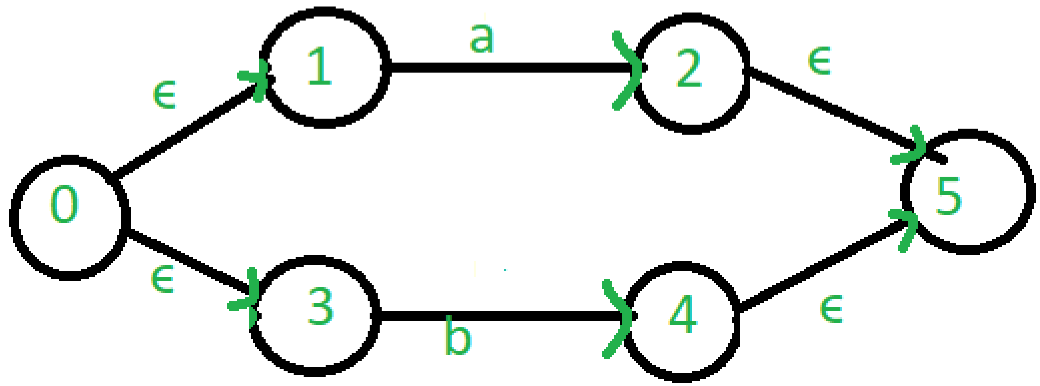
2. Regular expression to ε-NFA (a*)



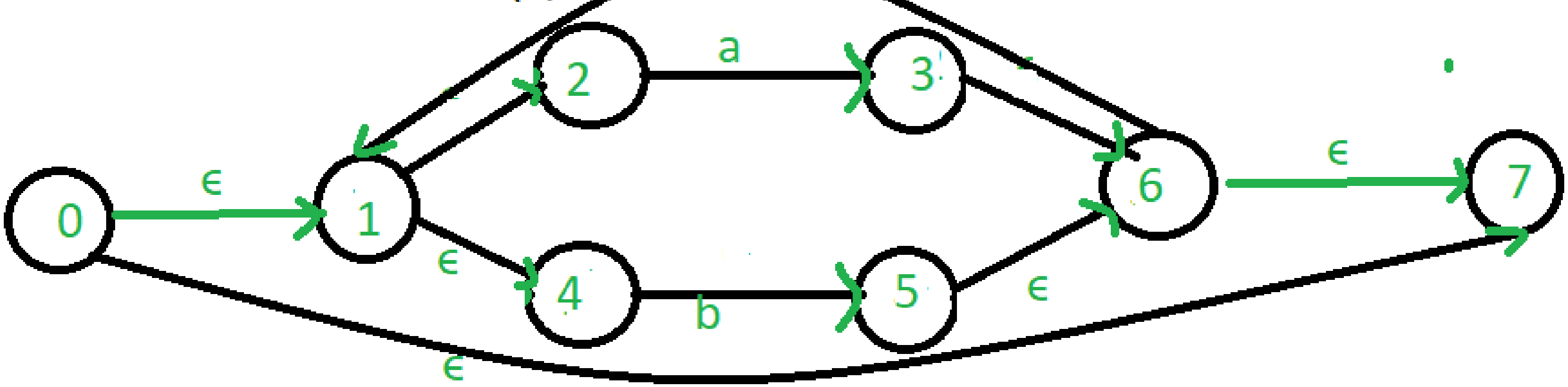


2. Regular expression to ε-NFA (a+b)*a

Step-1 First we create ϵ-NFA for (a/b)

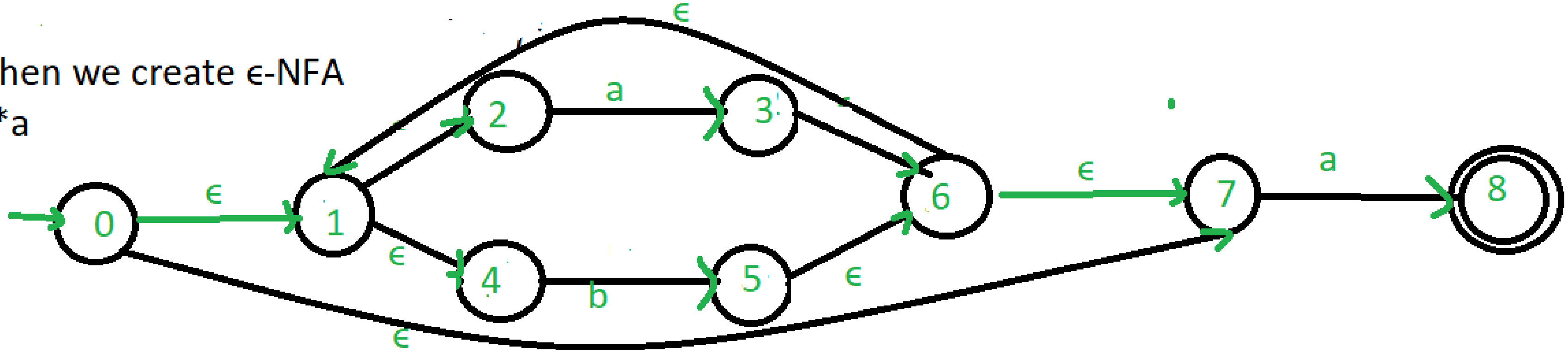


Step-2 Then create ϵ-NFA for (a/b)*

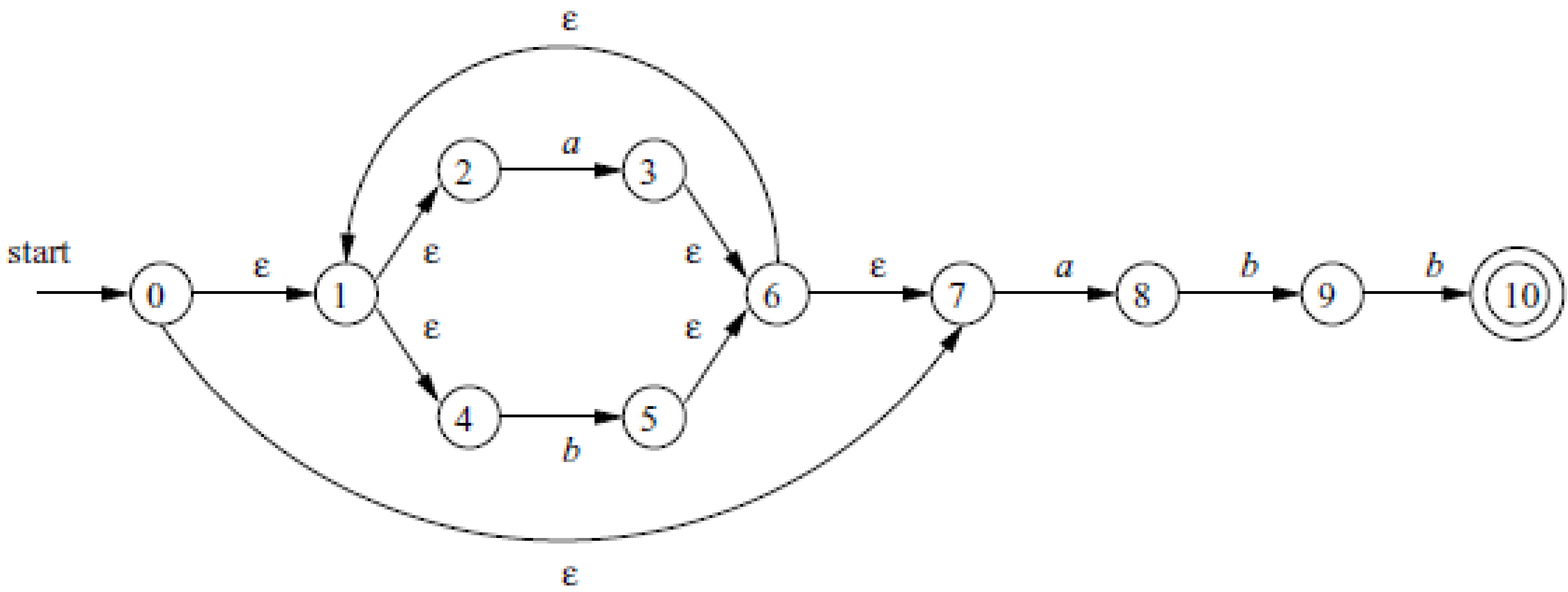


2. Regular expression to ε-NFA (a+b)*a

Step-3 Then we create ϵ-NFA for(a/b)*a



2. Regular expression to **ε-**NFA (a+b)*abb

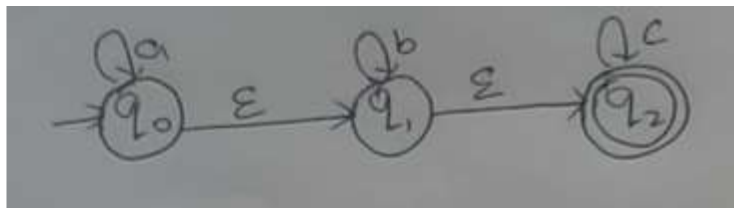


RE to Epsilon NFA - Assignment

- Regular expression → ab*
- Regular expression for strings that begin with 0 and end with 1 →0(0+1)*1

3. ε-NFA to DFA

- Convert ε-NFA to DFA for the regular expression a*b*c*
- *Solution:*
- Step 1: construct the epsilon NFA



- Step 2:
  - Find the ε-closure (q0) = x = A
  - δ(A,a) = q → ε-closure(q)
  - δ(A,b) = r → ε-closure (r)
  - δ(A,c) = s → ε-closure(s)

Scenario Explicating NFA

(Given by Karunya of III CSE B)

Non deterministic choices:

Nondeterministic computation can be thought of as a self-reproducing agent traveling in the state space.

1. At the start of computation the agent is in the initial state.

Non deterministic choices

2. Both before and after receiving an input symbol, the agent follows each
ϙ-labeled outgoing edge by producing its own
clone and sending it along the edge. Thus, the original
remains in the current location.

Non deterministic choices:

3. On receiving an input symbol, say a,
(a) If there is only one a-labeled outgoing edge, the agent follows the edge.
(b) If there is no a-labeled outgoing edge, the agent evaporates.

Non deterministic choices

4. When two agents collide in a state, they merge themselves
into one.