DAYANANDA SAGAR UNIVERSITY

SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Theory of Computation

Module 1

- **INTRODUCTION TO FINITE AUTOMATA:**
- Study and Central Concepts of Automata Theory, Finite Automata -Yet Another Method for Defining Languages, Deterministic and Nondeterministic Finite Automata, Finite Automata with Epsilon – transitions. An Application: Text Search.

Non Deterministic Finite Automata (NFA)

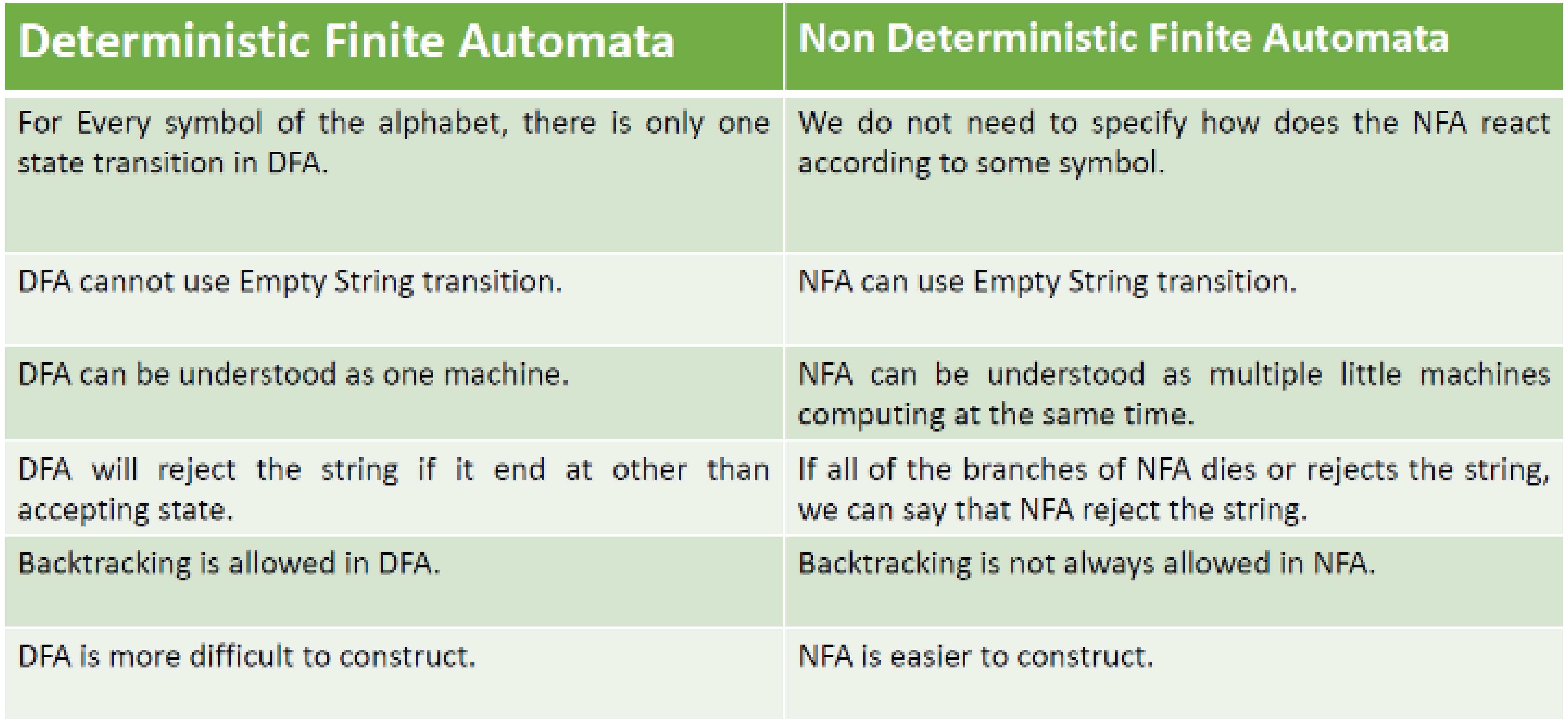
Non-Deterministic Finite Automata (NFA)

The Finite Automata is called Non Deterministic Finite Automata if there are more than one path for a specific input from current state to next state. Like DFA, NFA also have 5 tuples.

- A machine M = (Q,∑ ,δ,q0,F) Where ,
  - Q is finite set of states, which is non empty.
  - ∑ is input alphabet, indicates input set.
  - δ is transition function or mapping function. We can determine the next state using this function.

    δ : Q x ∑ -> *(In DFA δ : Q x ∑ ->)*
  - q0 is an initial state and is in Q
  - F is set of final states.

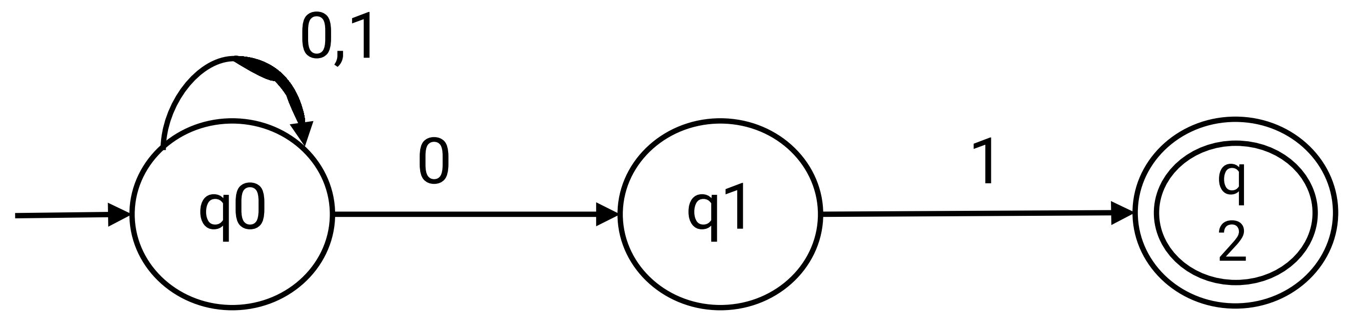
Difference Between DFA and NFA



Deterministic Finite Automata	Non Deterministic Finite Automata
For Every symbol of the alphabet, there is only one state transition in DFA.	We do not need to specify how does the NFA react according to some symbol.
DFA cannot use Empty String transition.	NFA can use Empty String transition.
DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
DFA will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
Backtracking is allowed in DFA.	Backtracking is not always allowed in NFA.
DFA is more difficult to construct.	NFA is easier to construct.

Example: NFA of string ending with 01

- String = {01,001,101,0001,0101,1001,1101…….}
- Language = {x01 | x є {0,1}*} or {(0+1)*01}
- NFA M = (Q,∑ ,δ,q0,F) Where
  - Q = {q0,q1,q2}
  - ∑ = {0,1}
  - q0 is initial state
  - F = {q2}
  - δ is define as table:

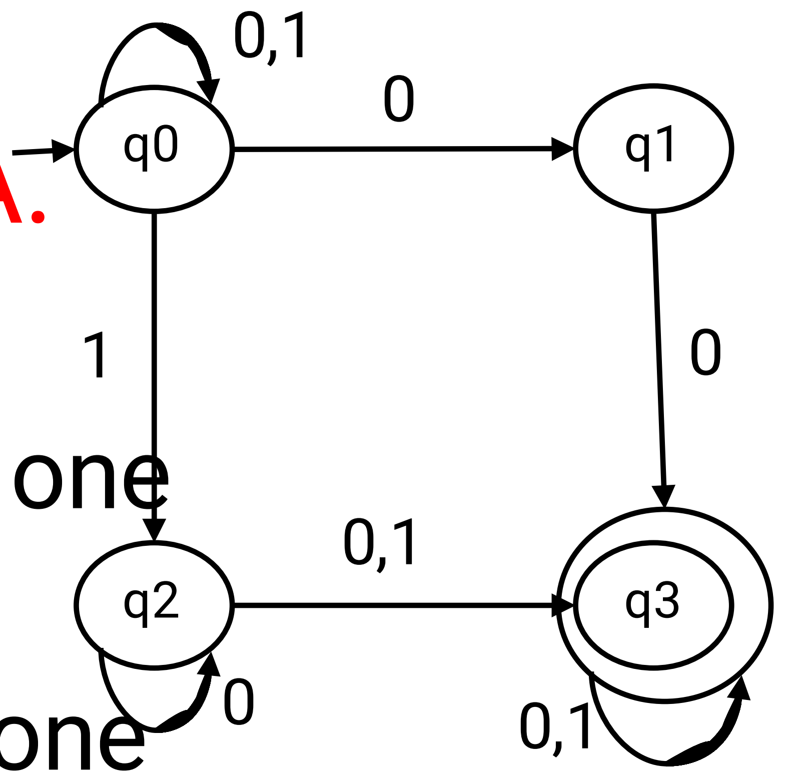


State/Input	0	1
→ q0	{q0,q1}	q0
q1	ϕ	q2
*q2	ϕ	ϕ

Processing of String using NFA

- To check the validity of any string for the NFA, always start with initial state.
- Pass current alphabet of string to current state and check entry in transition table or transition diagram. Move to next state according to the entry in the transition. Then move with next input symbol of string with new state.
- If there are more than one state in the entry then we will move with one, if we are not getting invalid path so we will back track and move to second path and vice versa
- At the end of string if we reach to the final state from any one path that means string is valid or accepted and if are not getting final state from either path then string is invalid or rejected.

Check the validity of the string 10 for the given NFA.



δ(q0,10)

{ δ(q0,1)=q0 & δ(q0,1)=q2 } two paths so we will select one

1. -> δ(q0,0) { δ(q0,1)=q0 }

{ δ(q0,0)=q0 & δ(q0,0)=q1 } two paths so we will select one

1.1 -> δ(q0,є) { δ(q0,0)=q0 } q0 is not final so backtrack and select another

1.2 -> δ(q1,є) { δ(q0,0)=q1 } q1 is also not final so again back track

2. -> δ(q2,0) { δ(q0,1)=q2 }

{ δ(q2,0)=q2 & δ(q2,0)=q3 } two paths so we will select one

2.1 -> δ(q2, є) { δ(q2,0)=q2 } q2 is not final so backtrack and select

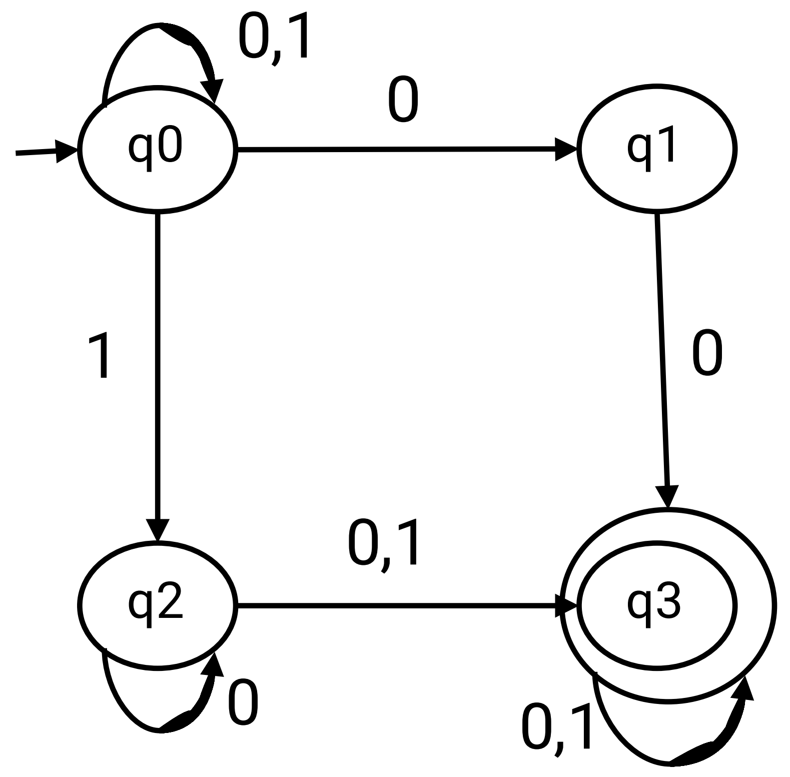
2.2 -> δ(q3, є) { δ(q2,0)=q3 } q3 is final so string is valid

Valid path is δ(q0,10) |- δ(q2,0) |- δ(q3, є)

Check the validity of the string 10 for the given NFA

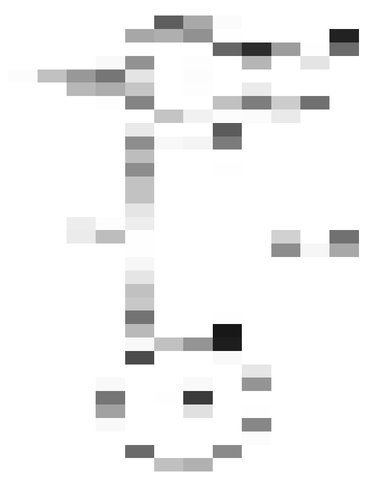
•

$$\begin{tikzcd}
 & \delta(q0,10) \arrow[dl] \arrow[dr] & & \\
\delta(q0,0)\ \{\delta(q0,1)=q0\} \arrow[d] \arrow[dr] & & \delta(q2,0)\ \{\delta(q0,1)=q0\} \arrow[d] \arrow[dr] & \\
\delta(q0, \epsilon)\ \{\delta(q0,0)=q0\} & \delta(q1, \epsilon)\ \{\delta(q0,0)=q1\} & \delta(q2, \epsilon)\ \{\delta(q2,0)=q2\} & \delta(q3, \epsilon)\ \{\delta(q2,0)=q3\} \\
\textit{Non Final} & \textit{Non Final} & \textit{Non Final} & \textit{Final}
\end{tikzcd}$$

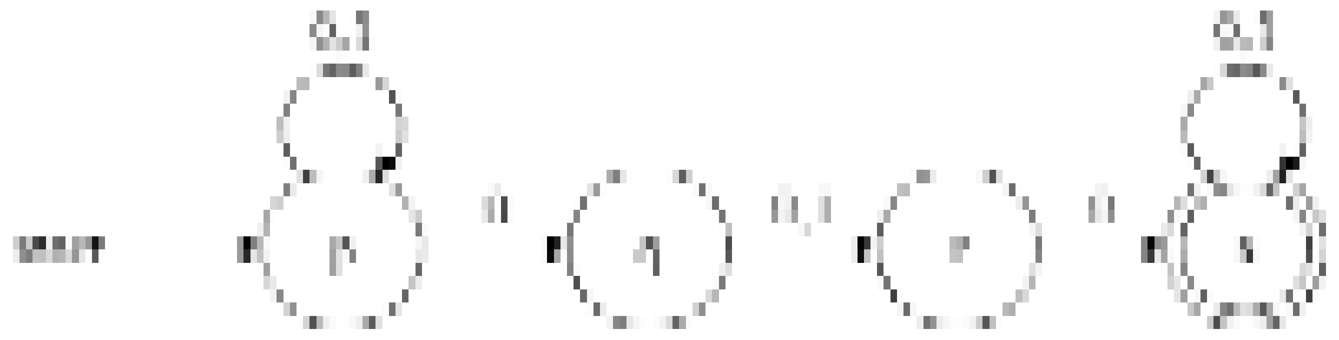


Exercise

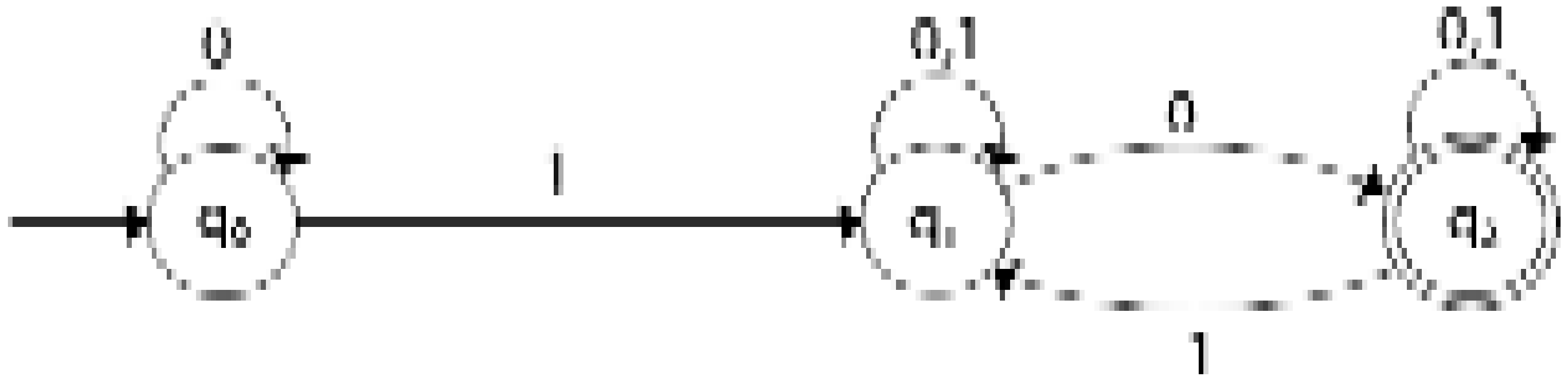
- Check the validity of string “aba” for the given NI



- Check the validity of string “00101” for the given NFA

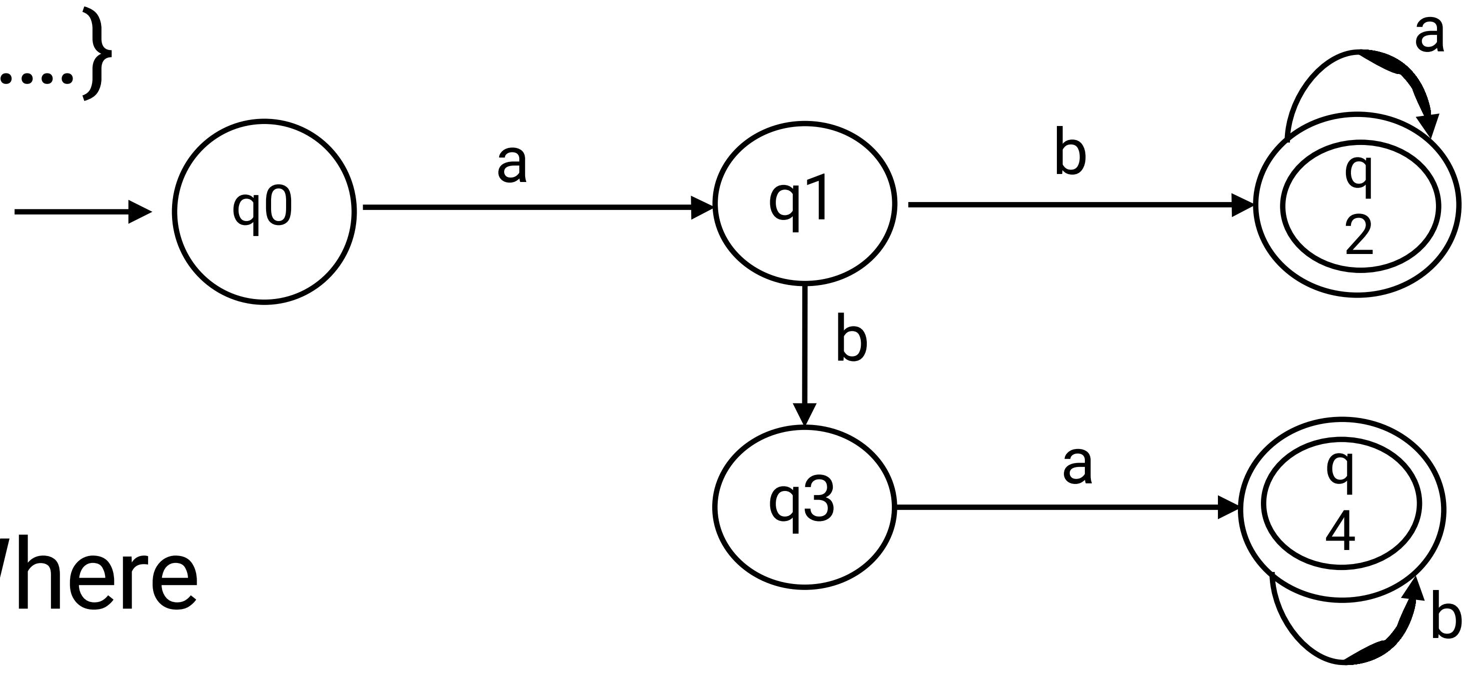


- Check the validity of string “1010” for the given NFA



Construct NFA which accept string {aba , n≥0} U {ab , n≥0}

- String = {ab,aba,abaa,abab,}ababb…….}
- Language = {aba,n≥0} U {ab,n≥0}

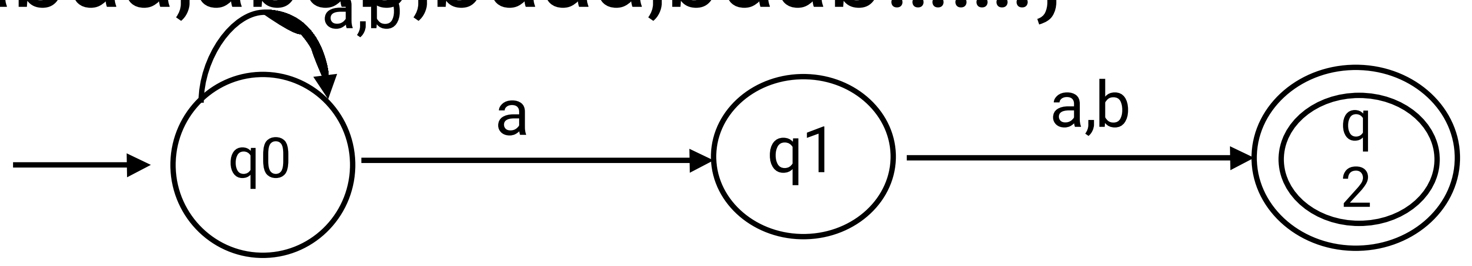


- NFA is define as M = (Q,∑ ,δ,q0,F) Where
- Q = {q0,q1,q2,q3,q4}
- ∑ = {a,b}
- q0 is initial state
- F = {q2,q4}
- δ is define as table:

State/Input	a	b
→ q0	q1	ϕ
q1	ϕ	{q2,q3}
*q2	q2	ϕ
q3	q4	ϕ
*q4	ϕ	q4

Construct NFA which accept string whose 2nd symbol from right is ' a' over {a,b}

- String = {ab,aa,aab,aaa,bab,baa,aaaa,aaab,abaa,abab,baaa,baab……}
- Language = {(a+b)*a(a+b)}

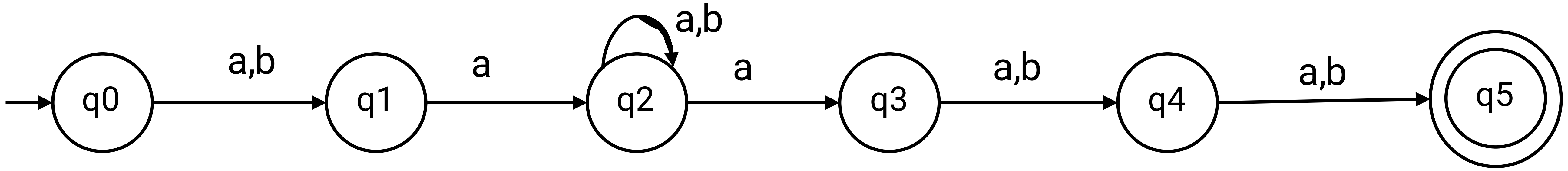


- NFA is define as M = (Q,∑ ,δ,q0,F) Where
- Q = {q0,q1,q2}
- ∑ = {a,b}
- q0 is initial state
- F = {q2}
- δ is define as table:

State/Input	a	b
→ q0	{q0,q1}	q0
q1	q2	q2
*q2	ϕ	ϕ

Construct NFA which accept string of length more than 4 whose 2nd symbol from left and 3rd symbol from right is ‘ a’ over {a,b}

- String = {aaaaa,baabb,aaabb,baaaa,aaaba,aaaab,baaab,baabb…….}
- Language = {(a+b)a(a+b)*a(a+b)(a+b)}



- NFA is define as M = (Q,∑ ,δ,q0,F) Where
- Q = {q0,q1,q2,q3,q4,q5}
- ∑ = {a,b}
- q0 is initial state
- F = {q5}
- δ is define as table:

State/Input	a	b
→ q0	q1	q1
q1	q2	ϕ
q2	{q2,q3}	q2
q3	q4	q4
q4	q5	q5
*q5	ϕ	ϕ

Exercise

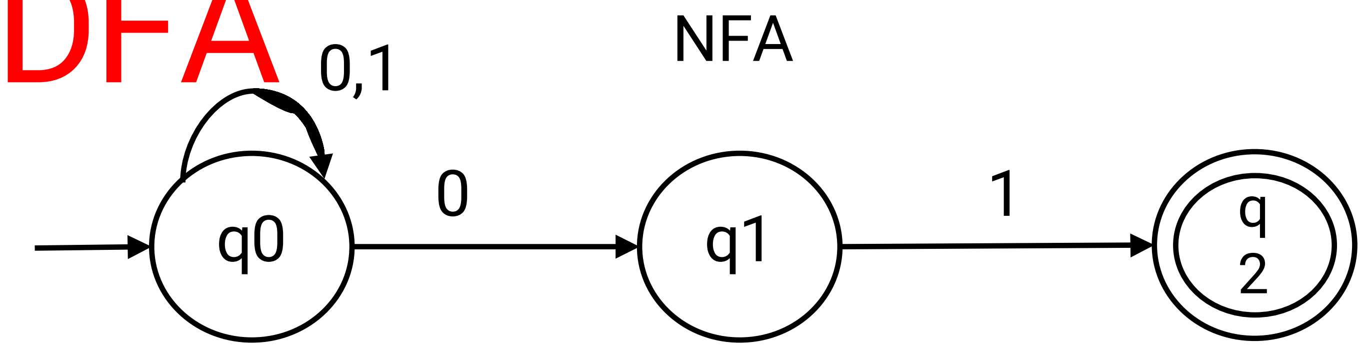
1. Construct NFA for the string over {a,b,c} that end with either ab or bc or ca
2. Construct NFA for the string generated by language L = a* + (ab)*
3. Construct NFA for the string over {0,1} that either start with 0 or end with 1.
4. Construct NFA for the string generated by language L = abc + ab(cb)*

Equivalence of NFA and DFA (Conversion from NFA to DFA)

- Let NFA N = (,∑ , and its equivalent DFA D = (,∑ , Such that L(N) = L(D).
- Input alphabets and start state of both the automata is always same
- Remaining tuples of DFA are calculated as:
- is the set of subset of is the power set of If has n states then has states.
- is the all set of D’s states that includes at least one final state of N
- For each set S ⊆ and for each input symbol a in ∑

(S,a) =

Convert given NFA into DFA



- NFA N = (,∑ ,Where
  - ▯ = {q0,q1,q2}
  - ▯ q0 is initial state
  - ▯ ∑ = {0,1}
  - ▯ = {q2}
  - ▯ as table
- Equivalent D = (,∑ ,
  - ▯ q0 is initial state
  - ▯ ∑ = {0,1}
  - ▯ , are calculated as:

Start with initial state q0

- (q0,0) = (q0,0) = {q0,q1} //New
- (q0,1) = (q0,1) = q0

New state {q0,q1}

- ({q0,q1},0) = (q0,0) U (q1,0)

  = {q0,q1} U ϕ = {q0,q1}
- ({q0,q1},1) = (q0,1) U (q1,1)

  = {q0} U {q2} = {q0,q2}

//New

New state {q0,q2}

- ({q0,q2},0) = (q0,0) U (q2,0)

  = {q0,q1} U ϕ = {q0,q1}
- ({q0,q2},1) = (q0,1) U (q2,1)

  = {q0} U ϕ = {q0}

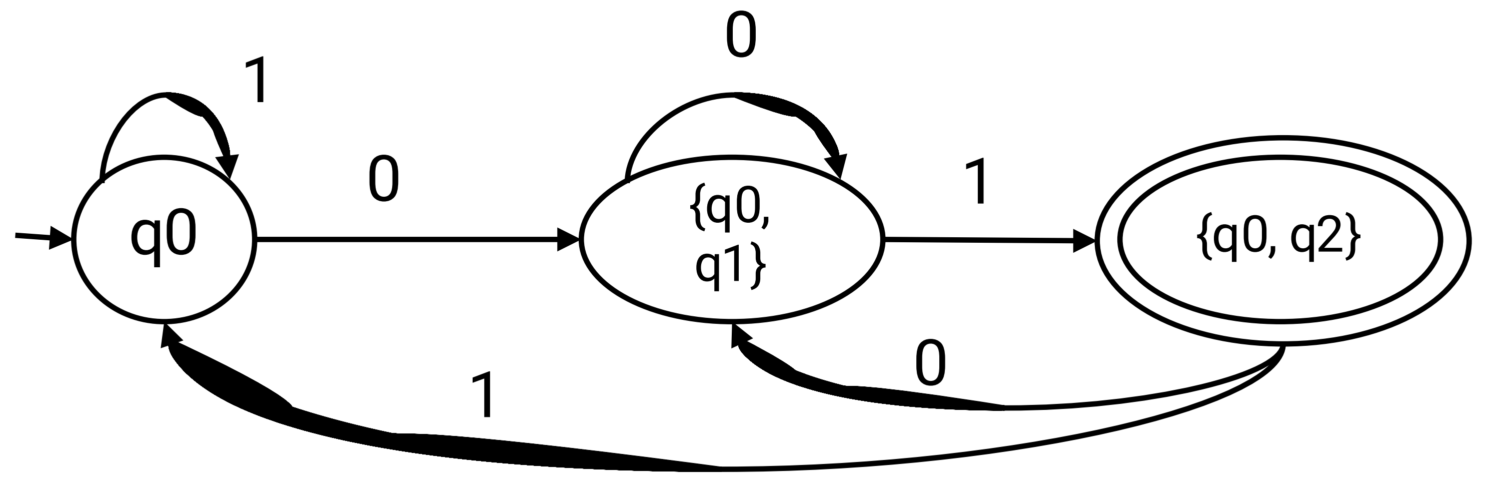
No New state so stop calculating states

- = {q0,{q0,q1},{q0,q2}}
- q2 is the final state of . In , {q0,q2} contains q2 so {q0,q2} is the final state of .

= {q0,q2}

- are as:

State/Input	0	1
→q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}
*{q0,q2}	{q0,q1}	q0



Convert given NFA into DFA

- NFA N = (,∑ ,Where
  - = {p,q,r,s}
  - p is initial state
  - ∑ = {0,1}
  - = {q,s}
  - as table
- Equivalent D = (,∑ ,
  - p is initial state
  - ∑ = {0,1}
  - , are calculated as:

Start with initial state p

- (p,0) = (p,0) = {q,s} //New
- (p,1) = (p,1) = q //New

Two New state {q,s} and q

- ({q,s},0) = (q,0) U (s,0)
  = {r} U ϕ = {r} //New
- ({q,s},1) = (q,1) U (s,1)
  = {q,r} U {p} = {p,q,r} // New
- (q,0) = (q,0) = {r}
- (q,1) = (q,1) = {q,r} //New

Three New state {r} , {q,r} and {p,q,r}

- (r,0) = (r,0) = {s} //New
- (r,1) = (r,1) = {p}
- ({q,r},0) = (q,0) U (r,0)
  = {r} U {s} = {r,s} //New
- ({q,r},1) = (q,1) U (r,1)
  = {q,r} U {p} = {p,q,r}

- ({p,q,r},0) = (q,0) U (r,0)

  = {q,s}U {r} U {s} = {q,r,s}

//New

- ({p,q,r},1) = (q,1) U (r,1)

  = {q} U {q,r} U {p} = {p,q,r}

Three New state {s} , {r,s} and {q,r,s}

- (s,0) = (s,0) = ϕ
- (s,1) = (s,1) = {p}
- ({r,s},0) = (r,0) U (s,0) = {s} U ϕ = {s}
- ({r,s},1) = (r,1) U (s,1)= {p} U {p} = {p}
- ({q,r,s},0) = (r,0) U (s,0)

  = {r}U {s} U ϕ = {r,s}

- ({q,r,s},1) = (r,1) U (s,1)

  = {q,r} U {p} U {p} = {p,q,r}

- No New state so stop calculating states

- ={p,q,r,s,{q,s},{q,r},{r,s},{p,q,r},{q,r,s}
- q & s are the final state of . So Except p & r all states of D are the final state because they include either q or s or both.

= {q,s,{q,s},{q,r},{r,s},{p,q,r},{q,r,s}}

- are as:

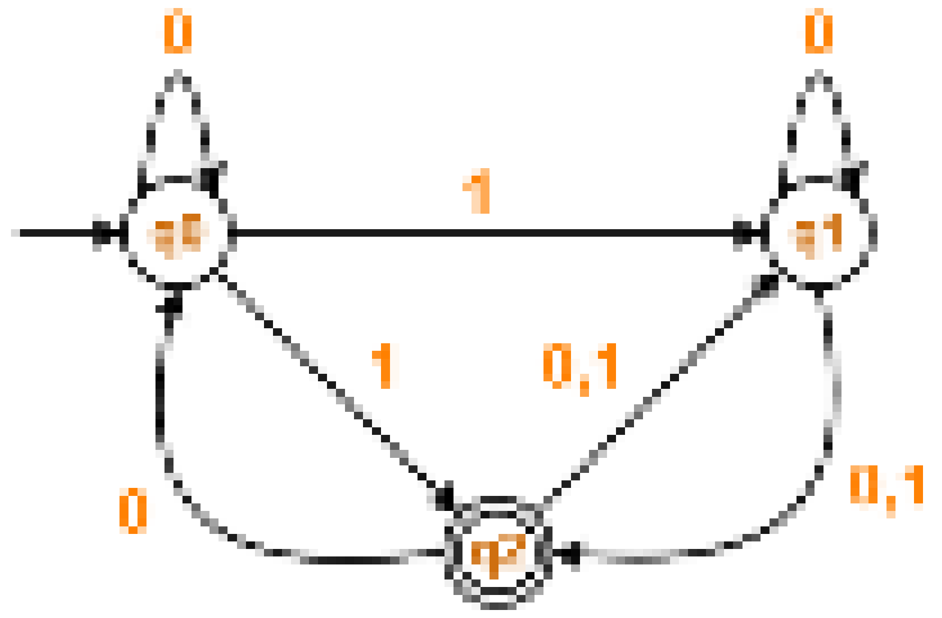
	0	1
→p	{q,s}	q
*q	r	{q,r}
r	s	p
*s	ϕ	p
*{q,s}	r	{p,q,r}
*{q,r}	{r,s}	{p,q,r}
*{r,s}	s	p
*{p,q,r}	{q,r,s}	{p,q,r}
*{q,r,s}	{r,s}	{p,q,r}

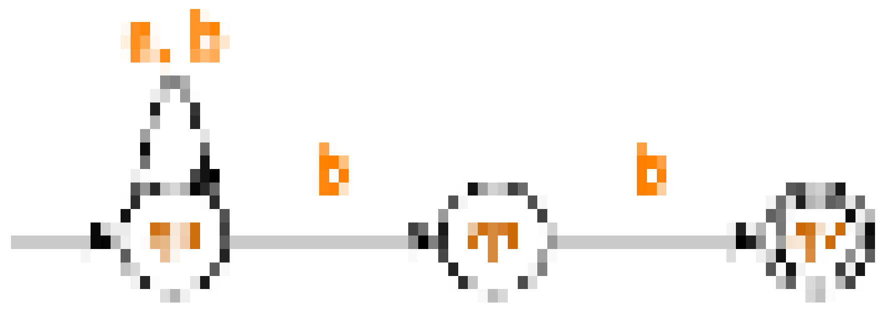
Exercise

- Convert the following NFA into its equivalent DFA

[illegible]

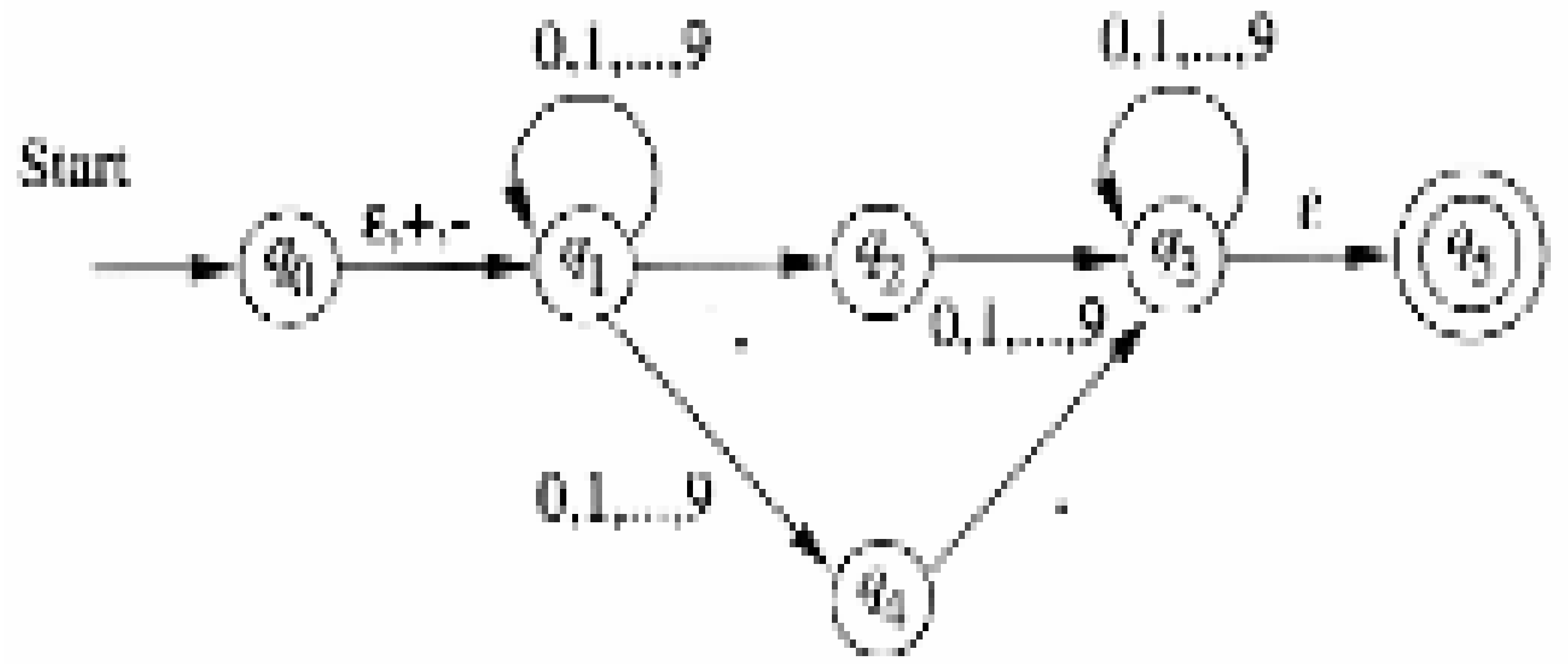
[illegible]





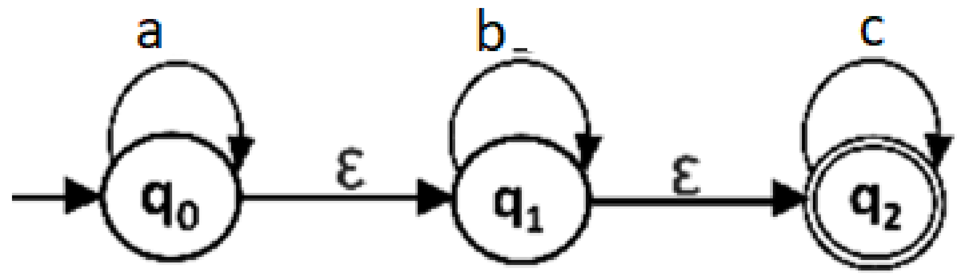
Epsilon NFA (є-NFA) / Finite Automata with Epsilon Transition

- NFA includes transition without taking any input as epsilon transition.
- If NFA contains є transition then NFA is called as є-NFA
- That allowed change of state without reading any input
- Example:- є-NFA that accepts decimal number
  1. An optional + or – sign
  2. A string of digits
  3. Decimal Point Sign
  4. A string of digits
- є-NFA E = (,∑ ,Where
  - ▯ = {q0,q1,q2,q3,q4,q5}
  - ▯ q0 is initial state
  - ▯ ∑ = {є, +,-,.,0-9}
  - ▯ = {q5}
  - ▯ as diagram



Construct є-NFA which accept set of string consisting of zero or more a’ s then followed by zero or more b’ s then followed by zero or more c’ s

- String = {є,a,b,c,aa,ab,ac,bb,bc,cc,abc,aabc,abbc,abcc…….}
- Language = {a*b*c*}



- є-NFA E = (,∑ ,Where
  - □ = {q0,q1,q2,q3,q4,q5}
  - □ q0 is initial state
  - □ ∑ = {є, +,-,.,0-9}
  - □ = {q5}
  - □ as diagram & table

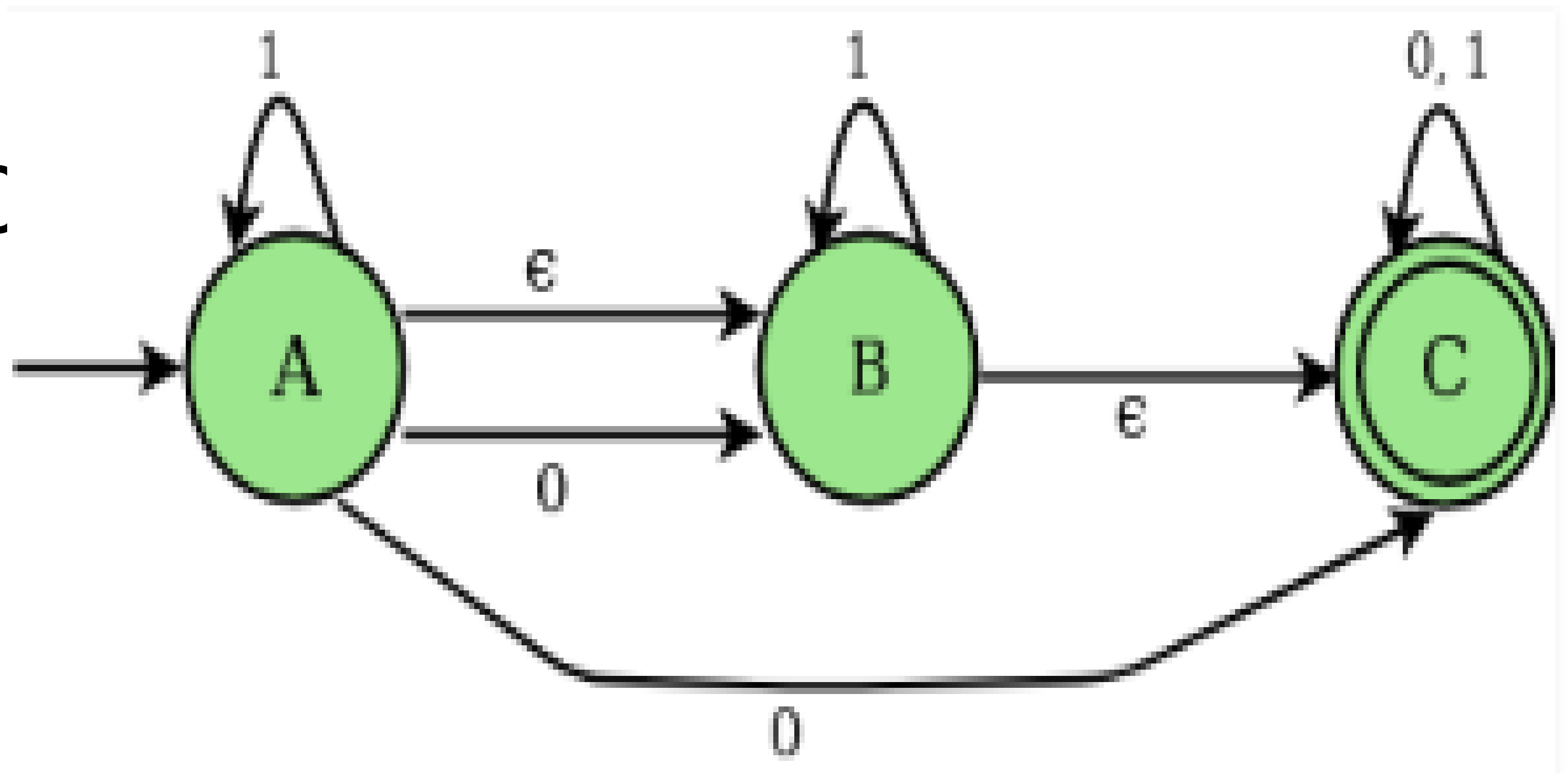
State/Input	є	a	b	c
→ q0	q1	q0	ϕ	ϕ
q1	q2	ϕ	q1	ϕ
*q2	ϕ	ϕ	ϕ	Q2

Exercise

1. Construct ϵ NFA for string over {0,1} which end with either 0110 or 010 or 00
2. Construct ϵ NFA for string over {a,b} which having substring either aba or aa.
3. Construct ϵ NFA for string over {0,1} which having substring either 110 or 11
4. Construct ϵ NFA for string over {0,1} which is either only 1 or last symbol is 1.
5. Construct ϵ NFA for the string generated by language L = a* + (ab)*
6. Construct ϵ NFA for the string over {0,1} that either start with 0 or end with 1.

Epsilon Closure

- Epsilon closure for a given state X is a set of states which can be reached from the states X with only (null) or ε moves including the state X itself.
- In other words, ε-closure for a state can be obtained by union operation of the ε-closure of the states which can be reached from X with a single ε move in recursive manner.
- Example
  - ε Closure (A) = {A,B,C
  - ε Closure (B) = {B,C}
  - ε Closure (A) = {C}

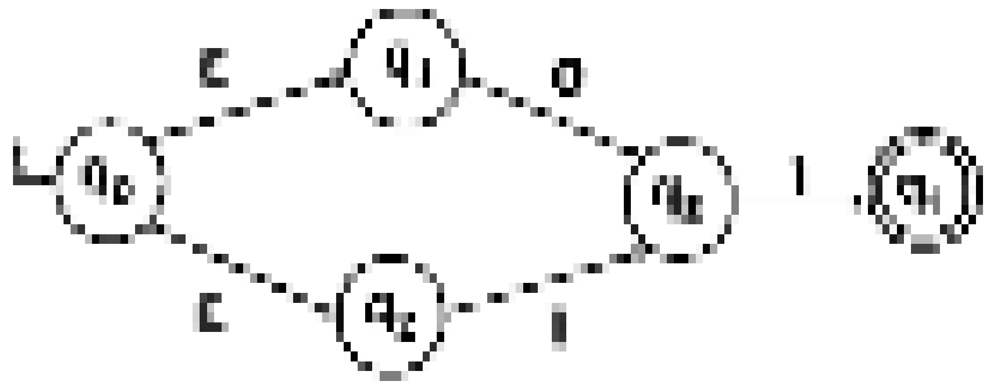


Conversion from ϵ-NFA to DFA

- Let εNFA E = (,∑ , and its equivalent DFA D = (,∑ , Such that L(E) = L(D).

1. is the subset of of D is ε-closed subsets of
2. = εclosure(q0)
3. is the set of states that contains at least one final state of E
4. For all state S in and all input a in ∑ , (S,a) is calculated as
   a) Let S = {p1,p2,p3…pk}
   b) Compute let this set be {r1,r2,r3…..rm}
   c) Then =

Convert given εNFA into DFA



- εNFA E = (,∑ , where
  - = {q0,q1,q2,q3,q4}
  - q0 is initial state
  - ∑ = {є, 0,1}
  - = {q4}
  - as above diagram
- Equivalent DFA D = (,∑ ,
  - ∑ = {0,1}
  - = εclosure(q0)

- εClosure of all states of E
  - εClosure(q0) = {q0,q1,q2}
  - εClosure(q1) = {q1}
  - εClosure(q2) = {q2}
  - εClosure(q3) = {q3}
  - εClosure(q4) = {q4}
- So = εclosure(q0) = {q0,q1,q2} = A

Start with intial state of D

- (A,0) = (q1,0) U (q2,0)
  = ϕ U {q3} U ϕ = {q3}
  = εClosure(q3) = q3 //New = B
- ,1) = (q1,1) U (q2,1)
  = ϕ U ϕ U {q3} = {q3}
  = εClosure(q3) = q3

New state q3 /B

- (B,0) = (q3,0) = ϕ
- (B,1) = (q3,1) = {q4}

  = εClosure(q4) = {q4} //New = C

New State q4/C

- (C,0) = (q4,0) = ϕ
- (C,1) = (q4,1) = ϕ

No New state so stop calculating states

DFA D = (,∑ ,

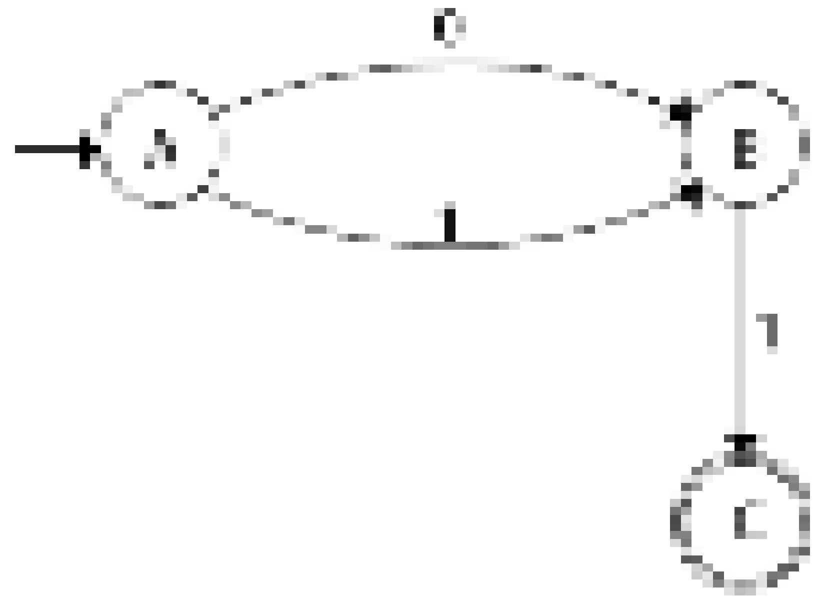
□ = {A,B,C}

□ ∑ = {0,1}

□ = A

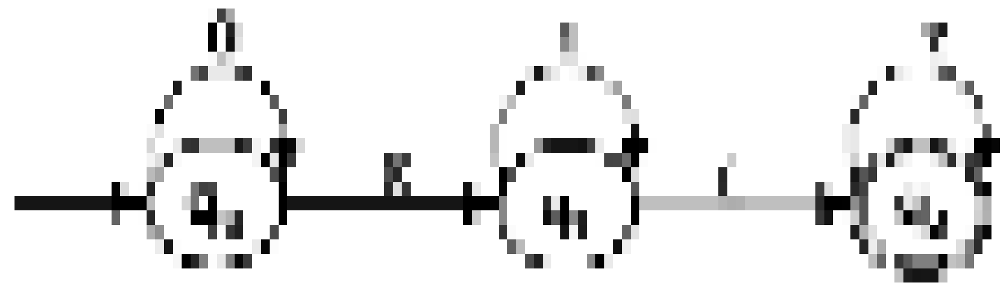
□ = C {q4 is final of E and q4 is in C of

- is



State/Input	0	1
→A	B	B
B	ϕ	C
*C	ϕ	ϕ

Convert given εNFA into DFA



- εNFA E = (,∑ , where
  - = {q0,q1,q2}
  - q0 is initial state
  - ∑ = {ϵ, 0,1,2}
  - = {q2}
  - as above diagram
- Equivalent DFA D = (,∑ ,
  - ∑ = {0,1,2}
  - = εclosure(q0)

- εClosure of all states of E
  - εClosure(q0) = {q0,q1,q2}
  - εClosure(q1) = {q1,q2}
  - εClosure(q2) = {q2}
- So = εclosure(q0) = {q0,q1,q2} = A

Start with intial state of D

- (A,0) = (q1,0) U (q2,0)

  = {q0} U ϕ U ϕ = {q0}

  = εClosure(q0) = {q0,q1,q2} = A
- ,1) = (q1,1) U (q2,1)

  = ϕ U {q1} U ϕ = {q1}

  = εClosure(q1) = {q1,q2} = B //New
- ,2) = (q1,1) U (q2,1)

  = ϕ U ϕ U {q2} = {q2}

  = εClosure(q2) = {q2} = C //New

Two New state {q1,q2} /B and q2/C

- (B,0)= (q2,0) = ϕ U ϕ = ϕ
- (B,1) = (q2,1) = {q1} U ϕ = {q1}
  = εClosure(q1) = {q1,q2} = B
- (B,2) = (q2,2) = ϕ U {q2} = {q2}
  = εClosure(q2) = {q2} = C

- (C,0) = (q2,0) = ϕ
- (C,1) = (q2,1) = ϕ
- (C,2) = (q2,2) = q2 = εClosure(q2) = q2 = C

No New state so stop calculating states

DFA D = (,∑ ,

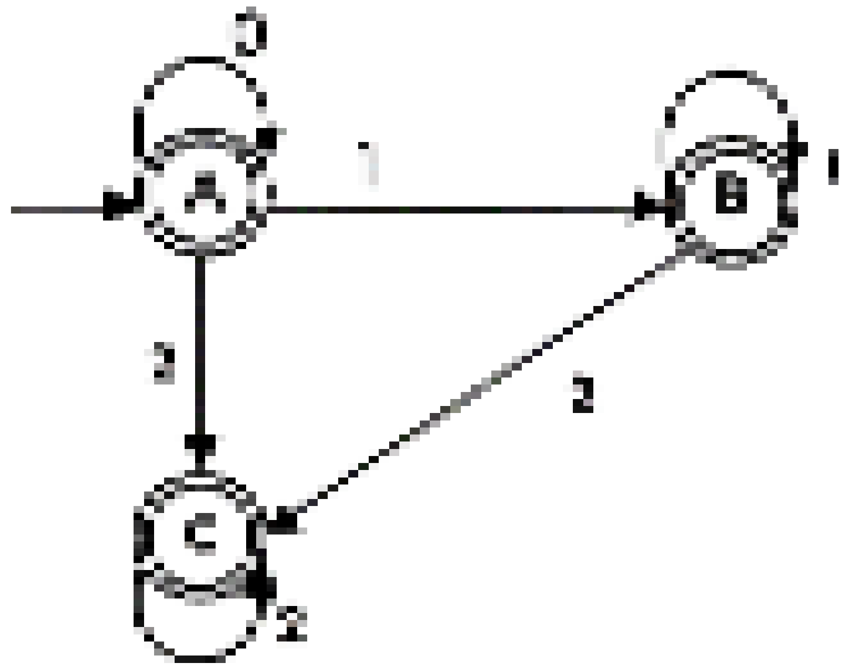
□ = {A,B,C}

□ ∑ = {0,1,2}

□ = A

□ = {A,B,C} {q2 is final of E and all state of D contains q2 so all are final }

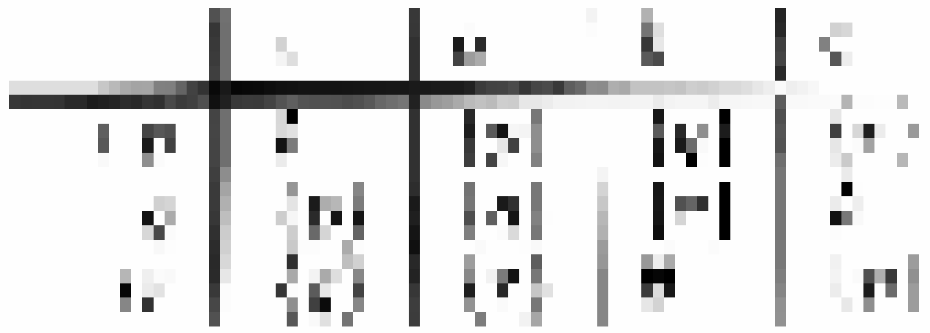
- is



State/Input	0	1	2
→A*	A	B	C
*B	ϕ	B	C
*C	ϕ	ϕ	C

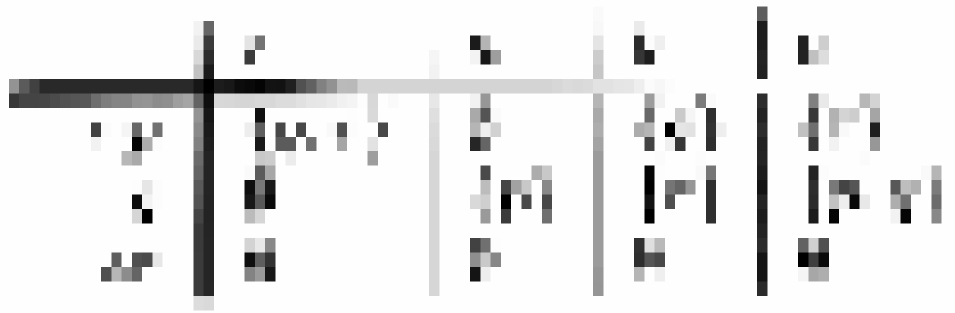
Exercise

1. Consider the following εNFA
    a) Compute εClosure of each state
    b) Give all string of length three or less accepted by automata
    c) Convert the above machine into DFA.



	ε	a	b	c
→p	∅	{p}	{q}	{r}
q	{p}	{q}	{r}	∅
*r	{q}	{r}	∅	{p}

2. Consider the following εNFA
    a) Compute εClosure of each state
    b) Give all string of length three or less accepted by automata
    c) Convert the above machine into DFA



	ε	a	b	c
→p	{q, r}	∅	{q}	{r}
q	∅	{p}	{r}	{p, q}
*r	∅	∅	∅	∅