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.

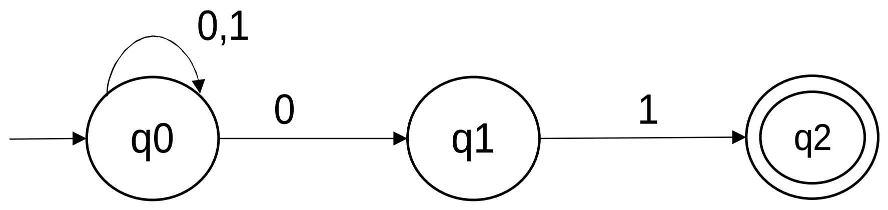
    $\delta : Q \times \Sigma \rightarrow 2^{Q}$ ***(In DFA $\delta : Q \times \Sigma \rightarrow Q$)***
  - q0 is an initial state and is in Q
  - F is set of final states.

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.

Example: NFA of string ending with 01

- String = {01,001,101,0001,0101,1001,1101.......}
- Language = {x01 | x ϵ {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:

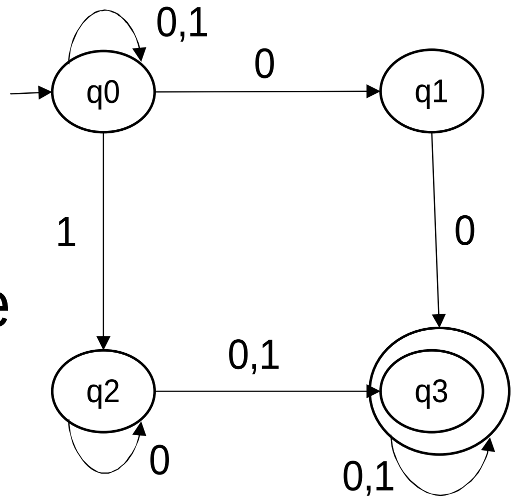


State/Input	0	1
→ q0	{q0,q1}	q0
q1	ϕ	q2
*q2	ϕ	ϕ

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 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,ϵ) { δ(q0,0)=q0 } q0 is not final so backtrack and select another

1.2 -> δ(q1,ϵ) { 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

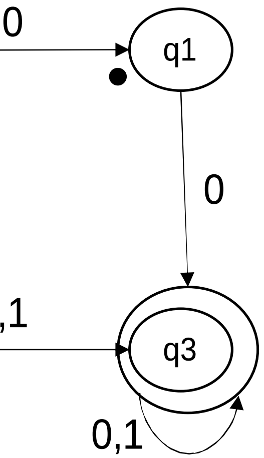
2.1 -> δ(q2, ϵ) { δ(q2,0)=q2 } q2 is not final so backtrack and select

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

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

Check the validity of the string 10 for the given NFA

q_0 — 1



δ(q0,10)

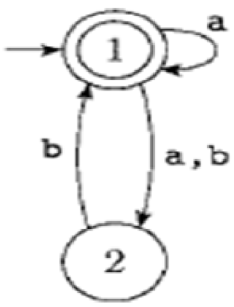
δ(q0,0)
{δ(q0,1)=q0}

δ(q2,0)
{δ(q0,1)=q0}

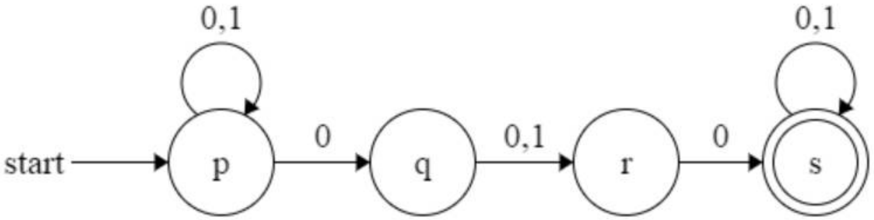
δ(q0, ϵ)	δ(q1, ϵ)	δ(q2, ϵ)	δ(q3, ϵ)
{δ(q0,0)=q0}	*{δ(q0,0)=q1}*	*{δ(q2,0)=q2}*	*{δ(q2,0)=q3}*
Non Final	***Non Final***	***Non Final***	***Final***

Exercise

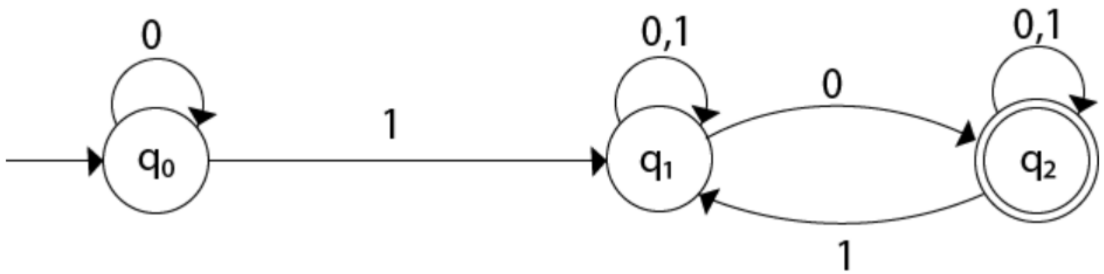
- Check the validity of string “aba” for the given NFA



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

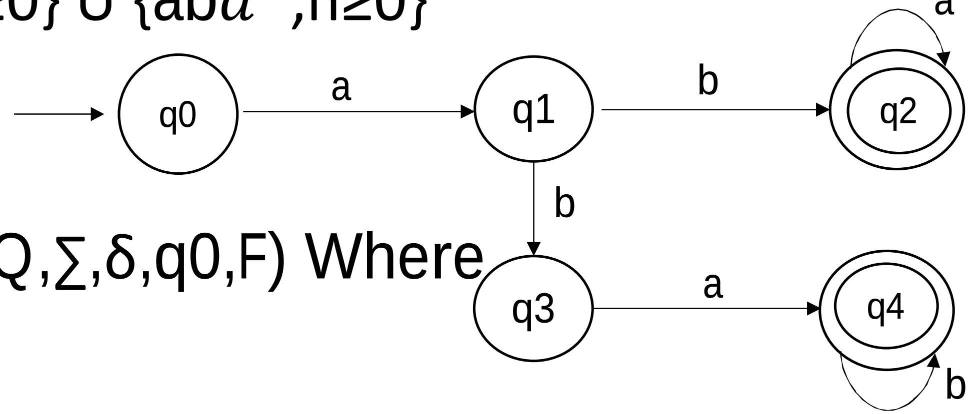


- Check the validity of string “1010” for the given NFA



Construct NFA which accept string {abab^n , n≥0} U {ab a^n , n≥0}

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

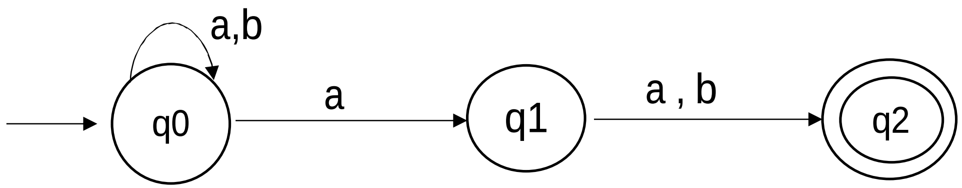


- NFA is define as M = (Q,∑,δ,q0,F) Where
- Q = {q0,q1,q2,q3,q4}
- ∑ = {a,b}
- q0 is initial state
- F = {q2,q4}
- δ is define as table:

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 2^{nd} 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)}

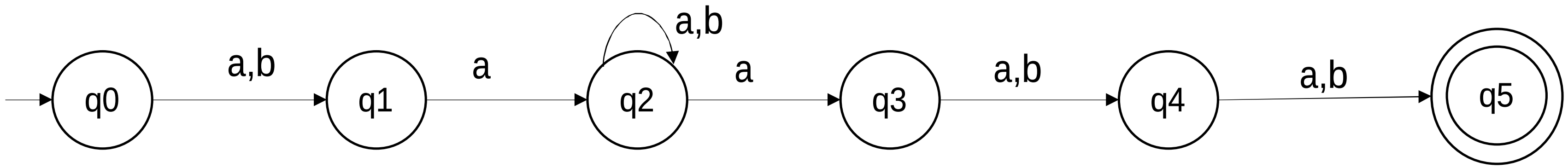


- NFA is define as M = (Q,∑,δ,q0,F) Where
- Q = {q0,q1,q2}
- ∑ = {a,b}
- q0 is initial state
- F = {q2}
- δ is define as table:

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

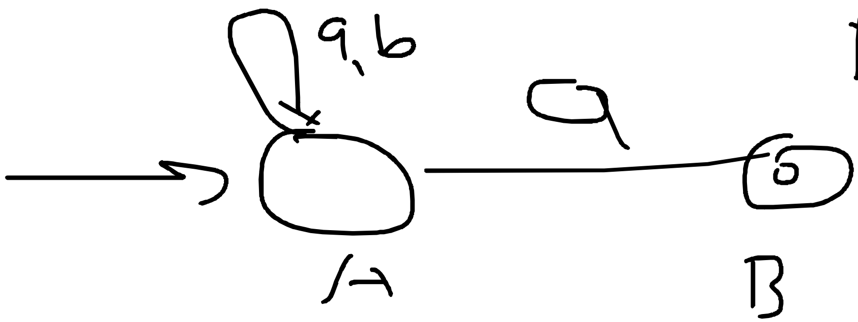
Exercise

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 over {0,1} that either start with 01 or end with 01.
3. Construct NFA for the string over {0,1} that either start with 01 and end with 01.
4. Construct NFA for the string generated by language L = abc + ab(cb)*
5. Construct NFA for the string over {0,1} that have substring 01 or 10.

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

- Let NFA N = (Q_N,∑,δ_N, $q0$, F_N) and its equivalent DFA D = (Q_D,∑,δ_D, $q0$, F_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:
- Q_D is the set of subset of Q_N i.e Q_D is the power set of Q_N. If Q_N has n states then Q_D has 2^n states.
- F_D is the all set of D's states that includes at least one final state of N
- For each set S ⊆ Q_N and for each input symbol a in ∑

$$\delta_D(S,a) = \bigcup_{p\ in\ S} \delta_N(p, a)$$

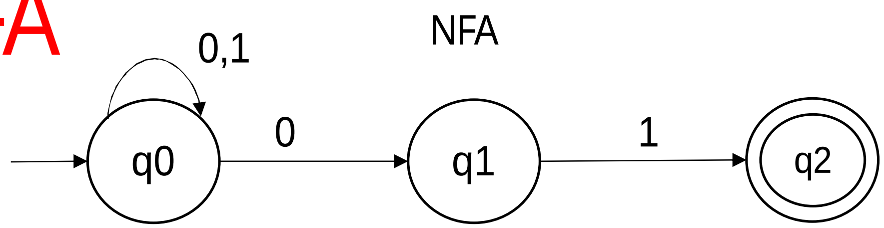


NFA

	a	b
→ A	{A,B}	{A}
B	∅	∅

DFA

Convert given NFA into DFA



- NFA N = $(Q_N, \Sigma, \delta_N, q0, F_N)$ Where
  - Q_N = {q0,q1,q2}
  - q0 is initial state
  - Σ = {0,1}
  - F_N = {q2}
  - δ_N as table

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

- Equivalent D = $(Q_D, \Sigma, \delta_D, q0, F_D)$
  - q0 is initial state
  - Σ = {0,1}
  - δ_D, Q_D, F_D are calculated as:

Start with initial state q0

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

New state {q0,q1}

- δ_D({q0,q1},0) = δ_N(q0,0) U δ_N(q1,0)
  = {q0,q1} U ϕ = {q0,q1}
- δ_D({q0,q1},1) = δ_N(q0,1) U δ_N(q1,1)
  = {q0} U {q2} = {q0,q2} //New

New state {q0,q2}

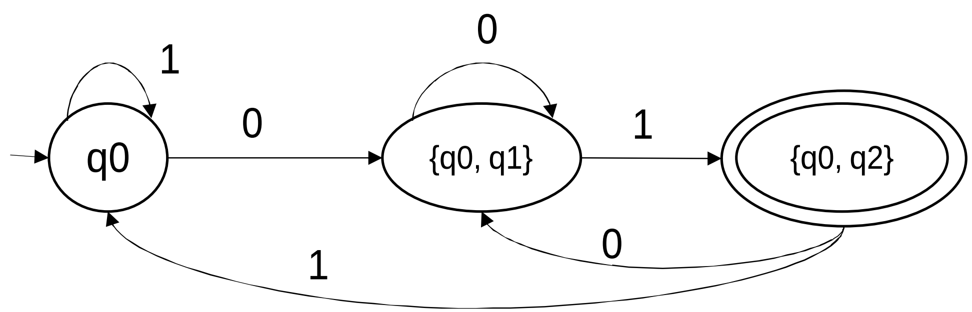
- δ_D({q0,q2},0) = δ_N(q0,0) U δ_N(q2,0)
  = {q0,q1} U ϕ = {q0,q1}
- δ_D({q0,q2},1) = δ_N(q0,1) U δ_N(q2,1)
  = {q0} U ϕ = {q0}

No New state so stop calculating states

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

  F_D = {q0,q2}
- δ_D 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

	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

	0
P	

- NFA N = (Q_N,∑,δ_N, $q0$, F_N) Where
  - Q_N = {p,q,r,s}
  - p is initial state
  - ∑ = {0,1}
  - F_N = {q,s}
  - δ_N as table
- Equivalent D = (Q_D,∑,δ_D, $q0$, F_D)
  - p is initial state
  - ∑ = {0,1}
  - δ_D, Q_D, F_D are calculated as:

Start with initial state p

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

Two New state {q,s} and q

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

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

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

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

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

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

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

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

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

  = {q,r} U {p} U {p} = {p,q,r}
- No New state so stop calculating states

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

  F_D = {q,s,{q,s},{q,r},{r,s},{p,q,r},{q,r,s}}
- δ_D are as:

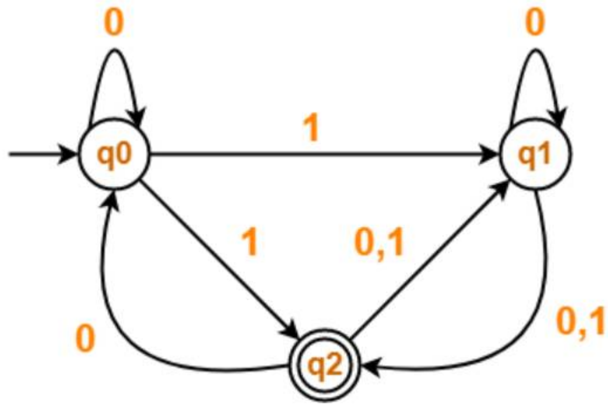
	0	1
→p	{q,s}	q
*q	r	{q,r}
r	s	p
*s	ϕ	p
*{q,s}	r	{p,q,r}
*{q,r}	{r,s}	{p,q,r}
*{r,s}	s	p
*{p,q,r}	{q,r,s}	{p,q,r}
*{q,r,s}	{r,s}	{p,q,r}

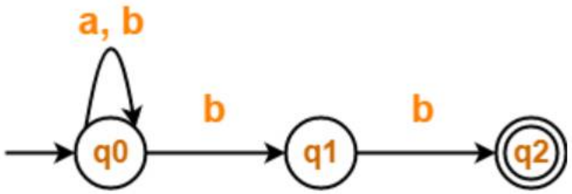
Exercise

- Convert the following NFA into its equivalent DFA

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r\}$	$\{r\}$
r	$\{s\}$	$\emptyset$
$*s$	$\{s\}$	$\{s\}$

	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	$\{r, s\}$	$\{t\}$
r	$\{p, r\}$	$\{t\}$
$*s$	$\emptyset$	$\emptyset$
$*t$	$\emptyset$	$\emptyset$





	0	1
$\rightarrow p$	$\{q, s\}$	$\{q\}$
$*q$	$\{r\}$	$\{q, r\}$
r	$\{s\}$	$\{p\}$
$*s$	$\emptyset$	$\{p\}$

r s

	0	1
$\rightarrow p$	qs	q
qs	r	pqr
q	r	qr
r	s	p
pqr	qrs	pqr
qr	rs	pqr
s	$\emptyset$	p
qrs	rs	pqr