Automata, Languages and Computation

1. Identify the language that starts with 'a' and ends with 'a'

Ans Collection of strings

$L = \{ aa, aaa, aaaa, \ldots \}$

L → language

$w = \{ aa, aaa, aaaa \ldots \}$

w → string

$\Sigma = \{a\}$

Σ → alphabet

$L = \{ w / w \in \{a\} \}$

Alphabet - symbol

String - sequence of finite no. of symbols.

Language - collection of string's.

Symbol: A symbol is smallest building block, which c be any alphabet, letter or picture.

Alphabets (Σ):

Alphabets are a set of symbols, which are always finite.

String:

String is a finite sequence of symbols from some alphabet.

Unit-I

Automata :

Deterministic Finite Automata

Construct the DFA which accepts the string of form $x01y$, $x,y \in \{0,1\}^*$

$\Sigma = \{0, 1\}$

$L = \{01, 0010, 1011, 1010, 0011, \ldots\}$ → depends on * value.

Finite automata Symbols.

- ○ — state
- →○ — start state
- → — transition
- ◎ — Final state

* — kleen closure — length is from null to any length.

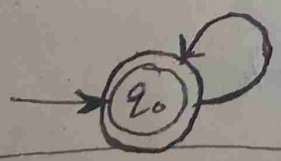
\+ — positive closure — length is from 1 to any length.

Construct DFA which accepts any strings that contain any number of a's and b's ; $\Sigma = \{a, b\}$

Ans $\Sigma = \{a, b\}$ alphabet

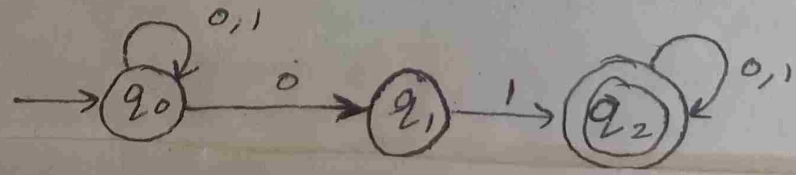
$L = \{e, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bba, bbb, \ldots\}$

To construct finite automata: $O(a+b)^+$



---

transition state diagram



State transition table

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	q_2	q_2

(or) q_2 ← Final states

ϵ - Empty string
ϕ - Empty set. $\{\}$
$\{\epsilon\}$ - not empty set

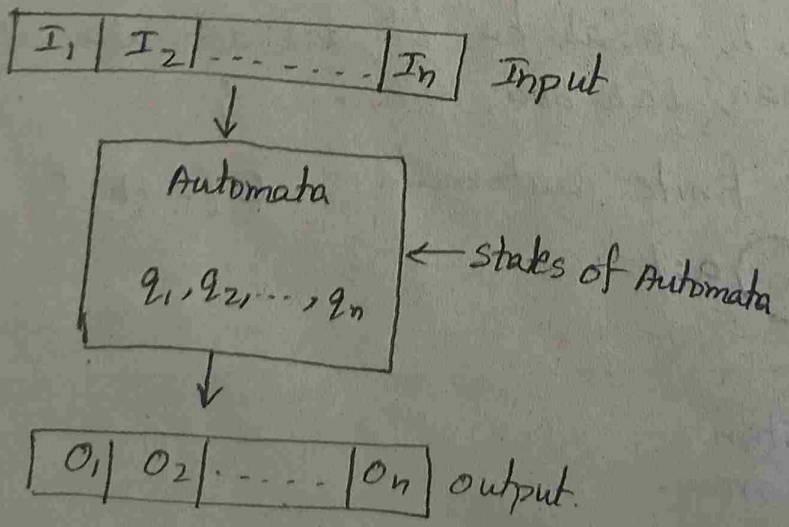
Formal definition of DFA (Deterministic Finite Automata)

It is a 5 tuple $\{Q, \Sigma, q, F, \delta\}$

$\{Q, \Sigma, \delta, q_0, F\}$

Q : Finite set of states
Σ : set of input symbols
q_0 : Initial state
F : set of Final states
δ : Transition Function which maps $Q \times \Sigma \rightarrow Q$

Block diagram



Formal definition of NFA (Nondeterministic Finite automata)

It is 5 tuple $\{Q, \Sigma, \delta, q_0, F\}$

Q : Finite set of states

Σ : set of i/p symbols

q_0 : initial state

F : set of final states

δ : Transition Function, $Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$

Applications

- Automata is a machine that can accept the strings of a Language L over an input alphabet Σ.
- Expressive Power of various Automata:

  The Expressive Power of any machine can be determined from the class or set of Languages accepted by that particular type of Machine. The increasing sequence of expressive power of machines:

  $$FA < DPDA < PDA < LBA < TM$$

  DPDA: Deterministic Push down Automata

  LBA: Linear bound automata.

  DFA and NFA are of same power because every NFA can be converted into DFA and every DFA can be converted into NFA. Turing machine TM is more powerful than any other machine

- Construct DFA for all the strings of the form $x01y$ where $x, y \in \{0,1\}^*$

Ans Formal Definition of DFA:

It is a 5 tuple $\{Q, \Sigma, \delta, q_0, F\}$

Q : Non empty finite set of states

Σ : set of i/p symbols.

δ : transition function, $(Q \times \Sigma \rightarrow Q)$

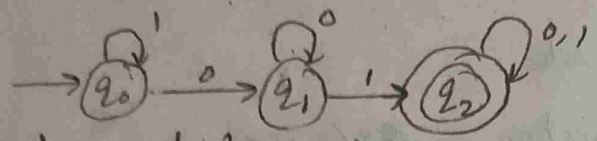
q_0 : Initial state

F : set of final states

$\Sigma = \{0, 1\}$

$L = \{01, 0010, 1011, \ldots\}$

State transition diagram



	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

acceptance of strings by DFA

$\omega = 11000100011$

$\hat{\delta}(q_0, 11000100011)$

$\rightarrow \delta(q_0, 1000100011)$

$\rightarrow \delta(q_0, 000100011)$

$\rightarrow \delta(q_1, 00100011)$

$\rightarrow \delta(q_1, 0100011)$

$\rightarrow \delta(q_1, 100011)$

$\rightarrow \delta(q_2, 00011)$

$\rightarrow \delta(q_2, 0011)$

$\rightarrow \delta(q_2, 011)$

$\rightarrow \delta(q_2, 11)$

$\rightarrow \delta(q_2, 1)$

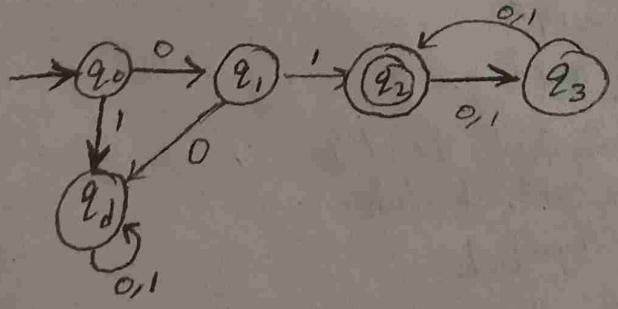
$\rightarrow \delta(q_2)$

- Construct DFA which accepts all the strings of even length and begins with 01, $\Sigma = \{0, 1\}$

Ans $\Sigma = \{0, 1\}$

$L = \{01, 0101, 0100, 0110, \ldots\}$

01, 0100, 0110, 0101, 0111, ...

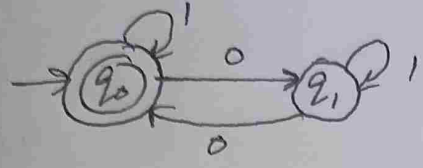


	0	1
$\rightarrow q_0$	q_1	q_d
q_1	q_d	q_2
$*q_2$	q_3	q_3
q_3	q_2	q_2
q_d	q_d	q_d

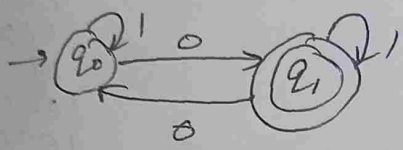
- Construct DFA that accepts all the strings which accepts strings with even no-of zeroes.

Ans: $\Sigma = \{0, 1\}$

$L = \{\epsilon, 00, 1, 0000, 11, 111, 1111, 001, 100, \ldots\}$



Even no-of strings.

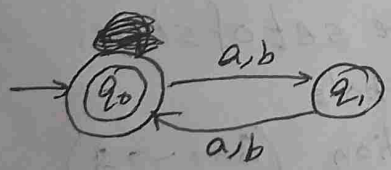


odd no-of strings.

- construct DFA which accepts all strings of length divisible by 2 $\Sigma = \{a, b\}^*$ ($w \bmod 2 = 0$)

Ans $\Sigma = \{a, b\}$

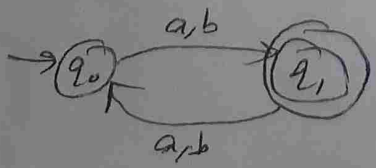
$L = \{\epsilon, aa, bb, ab, ba, aaaa, aabb, abab, bbbb, \ldots\}$



- Construct DFA which accepts all strings of odd length $\Sigma = \{a, b\}^*$ ($w \bmod 2 = 1$)

Ans $\Sigma = \{a, b\}$

$L = \{a, b, aaa, \ldots\}$

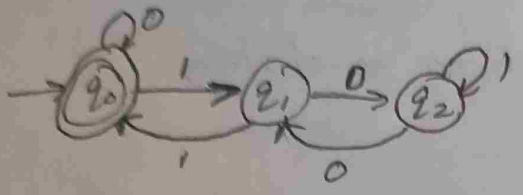


- Construct DFA which accepts all the binary strings divisible by 3.

Ans $\Sigma = \{0, 1\}$

$L = \{0, 3, 6, 9, 12, 15 \ldots\}$

$= \{0, 11, 110, 1001, 1100, 1111, 10010, 1010\}$



0 - 0
1 - 1
10 - 2
11 - 0
100 - 1
101 - 2
110 - 0
111 - 1
1000 - 2
1001 - 0
1010 - 1

- Construct DFA which accepts even no's of 'A's and even no of 'B's $\Sigma\{A, B\}$

Ans $\Sigma = \{A, B\}$

$L = \{\varepsilon, AA, BB, AABB, ABAB, BABA \ldots\}$

Formal definition of DFA

5 tuple $(Q, \Sigma, \delta, q_0, F)$

Q - Non empty finite set of states

Σ - Set of i/p symbols

δ - transition function $(Q \times \Sigma \to Q)$

q_0 - Initial state

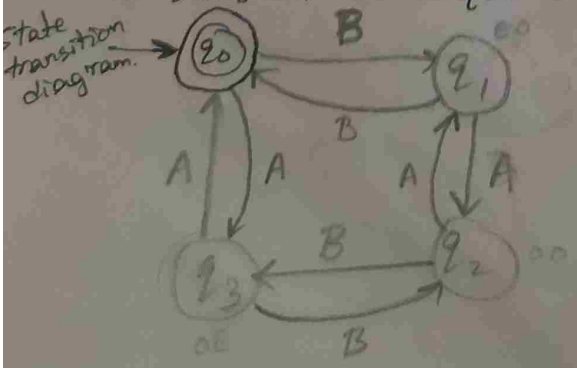
F - set of finite states.

Σ - Alphabet - Set of finite symbols.

w - string - finite sequence of symbols from some alphabets.

L - collection of strings.

State transition diagram.



abababaaabaB

- Construct DFA which accepts all binary no's divisible by 5 $\Sigma = \{0, 1\}$
- Construct DFA which accepts all strings that
  - → starts with substring AB
  - → contains substring "{AB}"

→ Construct DFA which accepts odd no' of A's and odd no. of B's $\Sigma \{A, B\}$

→ construct DFA which accepts even A's and odd B's $\Sigma \{A, B\}$

→ Construct DFA which accepts odd A's & even B's $\Sigma \{A, B\}$

State transition Table

	A	B
q_0	q_3	q_1
q_1	q_2	q_0
q_2	q_1	q_3
q_3	q_0	q_2

Extending transition function to strings

abbbaaaab

$\delta^\wedge(q_0, abab)$

$\delta^\wedge(q_0, \epsilon) = q_0$

$\delta^\wedge(q_0, a) = \delta(\delta^\wedge(q_0, \epsilon), a)$

$= \delta(q_0, a) = q_3$

$\delta^\wedge(q_0, ab) = \delta(\delta^\wedge(q_0, a), b)$

$= \delta(q_3, b) = q_2$

$\delta^\wedge(q_0, aba) = \delta(\delta^\wedge(q_0, ab), a)$

$= \delta(q_2, a) = q_1$

$\delta^\wedge(q_0, abab) = \delta(\delta^\wedge(q_0, aba), b)$

$= \delta(q_1, b) = q_0$ ✓

Accepted.

$\delta^\wedge(q_0, abbbaaaab)$

$\delta^\wedge(q_0, \epsilon) = q_0$

$\delta^\wedge(q_0, a) = \delta(\delta^\wedge(q_0, \epsilon), a)$
$= \delta(q_0, a) = q_3$

$\delta^\wedge(q_0, ab) = \delta(\delta^\wedge(q_0, a), b)$
$= \delta(q_3, b) = q_2$

$\delta^\wedge(q_0, abb) = \delta(\delta^\wedge(q_0, ab), b)$
$= \delta(q_2, b)$
$= q_3$

$\delta^\wedge(q_0, abbb) = \delta(\delta^\wedge(q_0, abb), b)$
$= \delta(q_3, b)$
$= q_2$

$\delta^\wedge(q_0, abbba) = \delta(\delta^\wedge(q_0, abbb), a)$
$= \delta(q_2, a) = q_1$

$\delta^\wedge(q_0, abbbaa) = \delta(\delta^\wedge(q_0, abbba), a)$
$= \delta(q_1, a) = q_2$

$\delta^\wedge(q_0, abbbaaa) = \delta(\delta^\wedge(q_0, abbbaa), a)$
$= \delta(q_2, a) = q_1$

$\delta^\wedge(q_0, abbbaaaa) = \delta(\delta^\wedge(q_0, abbbaaa), a)$
$= \delta(q_1, a) = q_2$

$\delta^\wedge(q_0, abbbaaaab) = \delta(\delta^\wedge(q_0, abbbaaaa), b)$
$= \delta(q_2, b)$
$= q_3$

↓
not final state
so not accepted.

Formal definition of NFA: (Non-deterministic Finite Automata)

It is a 5 Tuple $\{Q, \Sigma, q_0, F, \delta\}$

$\{Q, \Sigma, \delta, q_0, F\}$

Q: Finite set of states

Σ: Set of input symbols

δ: Transition Function $Q \times \Sigma \to 2^Q$

q_0: initial state

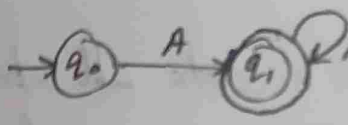
$F \leq$ set of final states.

- Construct NFA which accepts all the strings that starts with A $\Sigma\{A, B\}$

Ans $\Sigma = \{A, B\}$

$L = \{A, AB, ABA, ABAA \ldots\}$

state transition diagram.



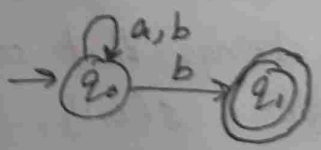
In NFA, Every state need not give O/P for all i/p's.

IN DFA we need O/P for all i/p's

- Construct NFA which accepts all the strings that end with B $\Sigma\{A, B\}$

Ans $\Sigma = \{A, B\}$

$L = \{B, AB, ABB, AABAB, \ldots\}$



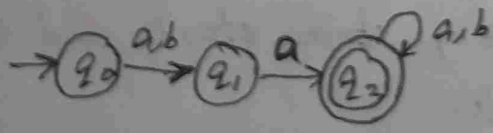
	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	ϕ	ϕ

In q_0 state for i/p b it can go to q_0 state or q_1 state this is NFA

- Construct NFA which accepts all the strings such that second symbol from LHS is a $\Sigma\{a, b\}$

Ans $\Sigma = \{a, b\}$

$L = \{aa, ba, aab, aaa, bab, baa, \ldots\}$

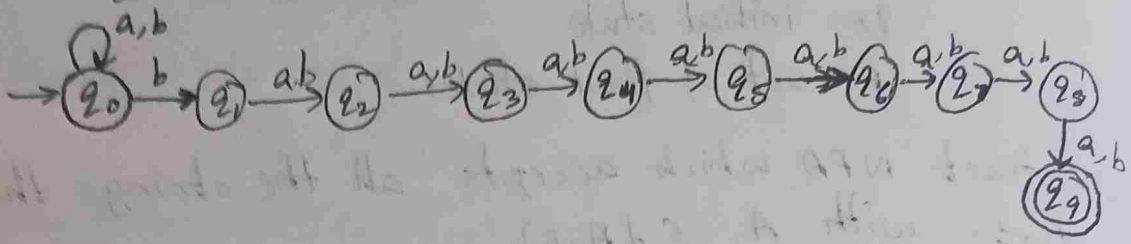


In NFA for state transition table we write as sets $\{\}$

- Construct NFA which accepts all the strings such that 9th symbol from RHS is b $\Sigma = \{a, b\}$

Ans $\Sigma = \{a, b\}$

$L = \{baaaaaaaa$



1. Construct Finite Automata which accepts all strings that contain odd no. of 1's and any no. of 0's $\Sigma = \{0, 1\}$
2. Construct FA which accepts all strings that ends with 00 $\Sigma = \{0, 1\}$
3. Construct FA which accepts all strings that does not contain exactly 2 a's. $\Sigma = \{0, 1\}$

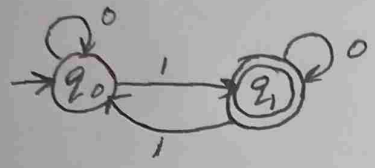
   (Construct for exactly 2 a's then change all Final states to non Final states and vice-versa)
4. Construct FA which accepts all strings that contain even no. of 0's followed by single 1 $\Sigma\{0, 1\}$.
5. Construct FA where every a is followed by single b $\Sigma\{a, b\}$
6. Construct FA where every strings starts with a and ends with b $\Sigma\{a, b\}$
7. Construct FA where every string starts and ends with different symbols.

(8) Construct FA where every string starts and ends with same symbols.

(9) Construct FA where length of string is ≤ 2.

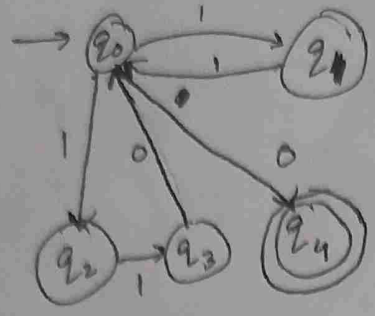
(1) $\Sigma = \{0, 1\}$

$L = \{1, 10, 100, 01, 001, 010, 111, 1011, 1101, 1110 \ldots\}$



Conversion of NFA to DFA

Ex:



NFA

	0	1
→ q_0	$\{q_4\}$	$\{q_1, q_2\}$
q_1	ϕ	$\{q_0\}$
q_2	ϕ	$\{q_3\}$
q_3	$\{q_0\}$	ϕ
* q_4	ϕ	ϕ

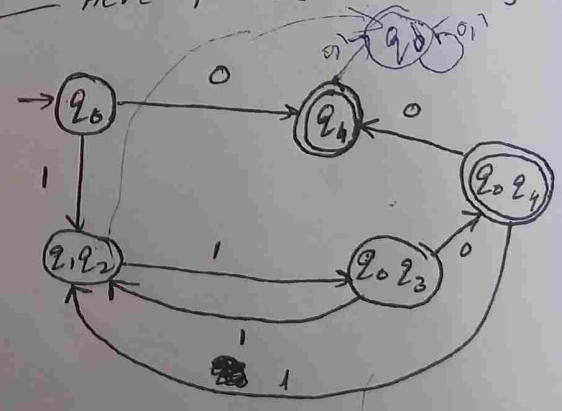
$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$2^Q =$ $\epsilon, q_0, q_1, q_2, q_3, q_4,$
$01, 02, 03, 04, 12, 13, 14,$
$23, 24, 34$
$012, 013, 014, 023, 024,$
$034, 123, 124, 134, 234,$
$0123, 0124, 0234, 1234, 0134$
$01234,$

Subset construction method

Keep the brackets correctly. Here put [ ] brackets only

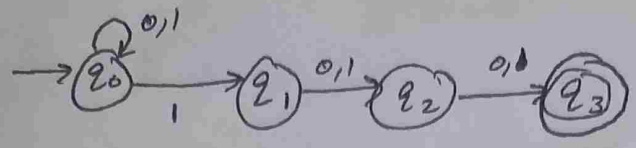
	0	1
→ q_0	$[q_4]$	$[q_1 q_2]$
q_4	ϕ	ϕ
$q_1 q_2$	ϕ	$[q_0 q_3]$
$q_0 q_3$	$[q_0 q_4]$	$[q_1 q_2]$
$q_0 q_4$	$[q_4]$	$[q_1 q_2]$



In original (NFA) the final state is q_4 so in this subset table in whichever state q_4 is there, it will be final states.

[ ] brackets represent single state only

(2)



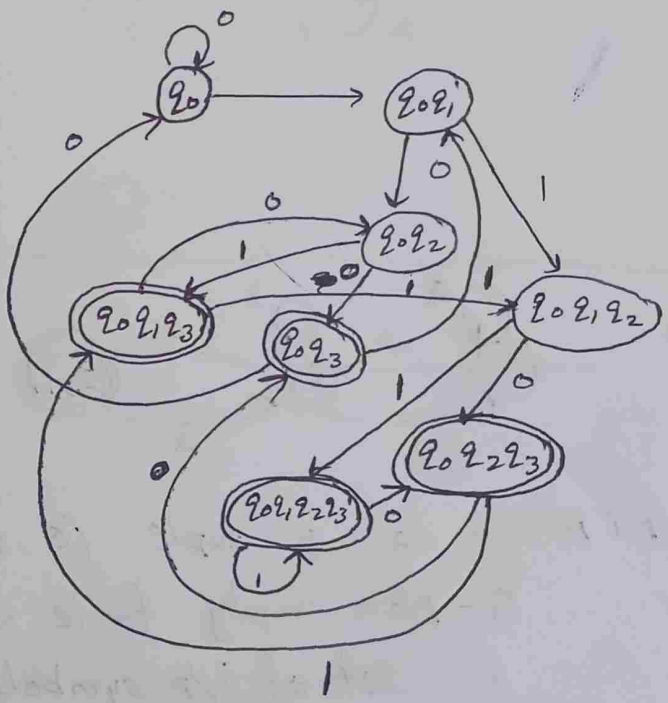
NFA

	0	1
→ q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
(q_3)	$\emptyset$	$\emptyset$

$Q = \{q_0, q_1, q_2, q_3\}$

Subset construction method

	0	1
→ q_0	$[q_0]$	$[q_0, q_1]$
q_0q_1	$[q_0q_2]$	$[q_0q_1q_2]$
q_0q_2	$[q_0q_3]$	$[q_0, q_1q_3]$
$q_0q_1q_2$	$[q_0q_2q_3]$	$[q_0q_1q_2q_3]$
(q_0q_3)	$[q_0]$	$[q_0q_1]$
($q_0q_1q_3$)	$[q_0q_2]$	$[q_0q_1q_2]$
($q_0q_2q_3$)	$[q_0q_3]$	$[q_0q_1q_3]$
($q_0q_1q_2q_3$)	$[q_0q_2q_3]$	$[q_0q_1q_2q_3]$



Formal definition of DFA

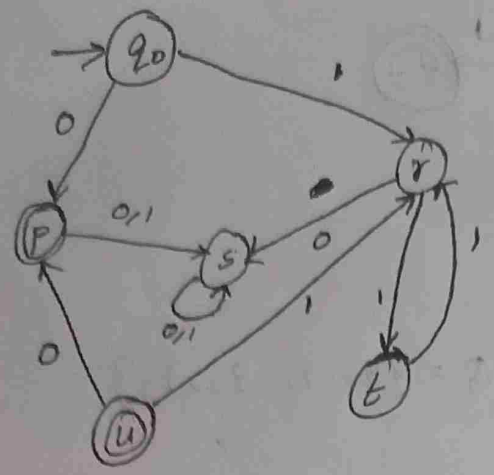
$Q = \{[q_0], [q_0q_1], [q_0q_2], [q_0q_1q_2], [q_0q_3], [q_0q_1q_3], [q_0q_2q_3], [q_0q_1q_2q_3]\}$

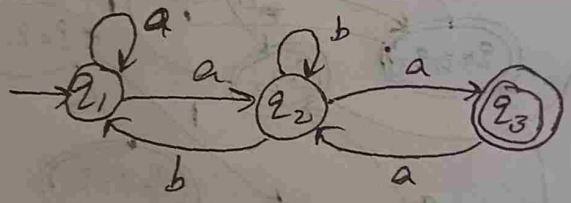
$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F = \{[q_0q_3], [q_0q_1q_3], [q_0q_2q_3], [q_0q_1q_2q_3]\}$

(3)





A) NFA is a 5 tuple $(Q, \Sigma, \delta, q_0, F)$

Q - Non empty finite set of states

Σ - Set of i/p symbols

δ - transition function $\Rightarrow Q \times \Sigma \to 2^Q$

q_0 - Initial state

F - set of final states.

$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$q_0 = q_1$

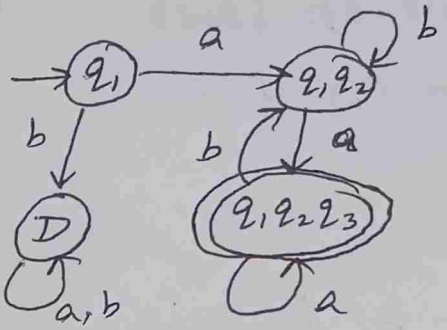
$F = \{q_3\}$

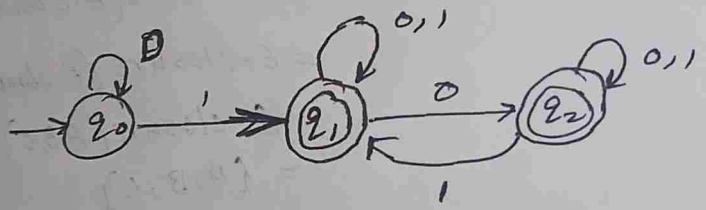
State transition table

	a	b
→ q_1	$\{q_1, q_2\}$	ϕ
q_2	$\{q_3\}$	$\{q_1, q_2\}$
* q_3	$\{q_2\}$	ϕ

subset construction method.

	a	b
→ q_1	$[q_1, q_2]$	ϕ D
$[q_1, q_2]$	$[q_1, q_2, q_3]$	$[q_1, q_2]$
* $[q_1, q_2, q_3]$	$[q_1, q_2, q_3]$	$[q_1, q_2]$
D	D	D



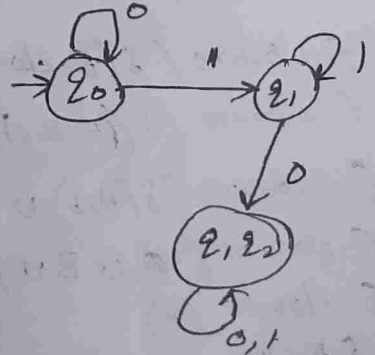


Any

	0	1
→ q_0	$\{q_0\}$	$\{q_1\}$
* q_1	$\{q_1, q_2\}$	$\{q_1\}$
* q_2	$\{q_2\}$	$\{q_1, q_2\}$

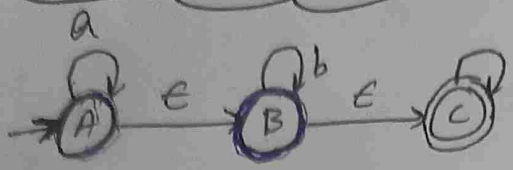
Subset construction method

	0	1
→ q_0	$[q_0]$	$[q_1]$
$[q_1]$	$[q_1, q_2]$	$[q_1]$
$[q_1, q_2]$	$[q_1, q_2]$	$[q_1, q_2]$



• NFA with ϵ $\Rightarrow$ $\delta \Rightarrow Q \times \Sigma \cup \{\epsilon\} = 2^Q$

If no final state is given take last state as final or first state as final



(Here all states are final states.)

Computation of ϵ closure. (Every state as its own state in the set) (To which states it can go from that state using ϵ as transition is in the set)

ϵ-closure(A) = {A, B, C}

ϵ-closure(B) = {B, C}

ϵ-closure(C) = {C}

$a^*b^*c^*$ only in order of a b c
abc, aabbcc, abbcc, bc, ...

$(a+b+c)^*$
ϵ, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc
any combination of a's, b's, c's

δ	ϕ	a	b	c
→ A	{A,B,C}	{B,C}	{C}	
B	ϕ	{B,C}	{C}	
C	ϕ	ϕ	{C}	

$\delta(A, a)$ = ϵ-closure(ϵ-closure $\delta(A,a)$)
= ϵ-closure (ϵ-closure(A))
= ϵ-closure(A,B,C)
= {A, B, C}

$\delta(A, a)$ = ϵ-closure (δ(ϵ-closure(A), a))
= ϵ-closure (δ(A,B,C), a)
= ϵ-closure ($\delta(A,a) \cup \delta(B,a) \cup \delta(C,a)$)
= ϵ-closure ($A \cup \phi \cup \phi$)
= ϵ-closure (A)
$\delta(A,a)$ = {A, B, C}

$\delta(A,b)$ = ϵ-closure (δ(ϵ-closure(A), b))
= ϵ-closure (δ((A,B,C), b))
= ϵ-closure ($\delta(A,b) \cup \delta(B,b) \cup \delta(C,b)$)
= ϵ-closure ($\phi \cup B \cup \phi$)
= ϵ-closure (B)
= {B, C}

$\delta(A,c) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A),b))$

$= \epsilon\text{-closure}(\delta((A,B,C),c))$

$= \epsilon\text{-closure}(\delta(A,c) \cup \delta(B,c) \cup \delta(C,c))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup C)$

$= \epsilon\text{-closure}(C)$

$= \{C\}$

$\delta(B,a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(B),a))$

$= \epsilon\text{-closure}(\delta((B,C),a))$

$= \epsilon\text{-closure}(\delta(B,a) \cup \delta(C,a))$

$= \epsilon\text{-closure}(\phi \cup \phi)$

$= \epsilon\text{-closure}(\phi)$

$\delta(B,a) = \phi$

$\delta(B,b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(B),b))$

$= \epsilon\text{-closure}(\delta((B,C),b))$

$= \epsilon\text{-closure}(\delta(B,b) \cup \delta(C,b))$

$= \epsilon\text{-closure}(B \cup \phi)$

$= \epsilon\text{-closure}(B)$

$\delta(B,b) = \{B,C\}$

$\delta(B,c) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(B),c))$

$= \epsilon\text{-closure}(\delta((B,C),c))$

$= \epsilon\text{-closure}(\phi \cup C)$

$\delta(B,c) = \{C\}$

$\delta(C,a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(C),a))$

$= \epsilon\text{-closure}(\delta(C,a))$

$= \epsilon\text{-closure}(\phi)$

$\delta(C,a) = \phi$

$\delta(C,b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(C),b))$

$= \epsilon\text{-closure}(\delta(C,b))$

$\delta(C,b) = \phi$

$\delta(C,c) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(C),c))$

$= \epsilon\text{-closure}(\delta(C,c))$

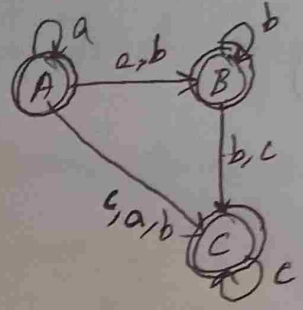
$\delta(C,c) = \{C\}$

NFA without Null.

δ	a	b	c
→ A	{A,B,C} ~~{B,C}~~ ~~{C}~~		
B	∅	{B,C}	{C}
C	∅	∅	{C}

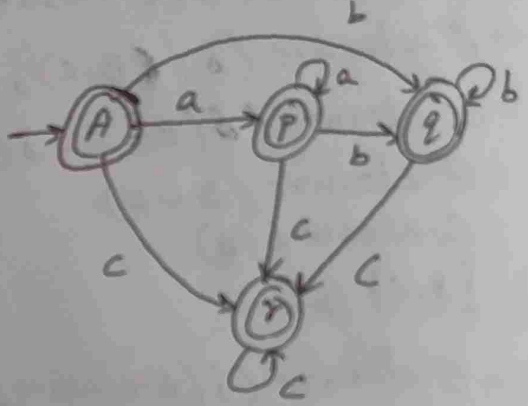
Wherever final state is there it will also become final state.
Ex: A closure also has ϵ so A is also final state.
Similarly B also final state

State transition diagram

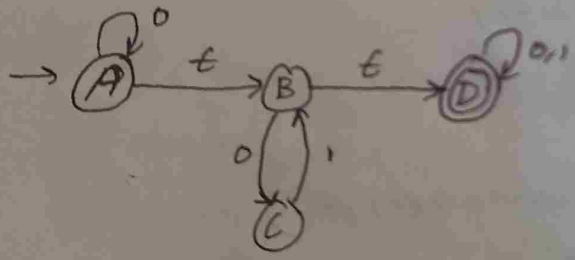


DFA

	a	b	c
→ A	[ABC]	[BC]	[C]
P [ABC]	[ABC]	[BC]	[C]
Q [BC]	∅	[BC]	[C]
R [C]	∅	∅	[C]



⇒



ϵ-closure(A) = {A, B, D} (F)

ϵ-closure(B) = {B, D} (F)

ϵ-closure(C) = {C}

ϵ-closure(D) = {D} (F)

$\delta(A,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A),0))$

$= \epsilon\text{-closure}(\delta((A,B,D),0))$

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(D,0))$

$= \epsilon\text{-closure}(A \cup C \cup D)$

$\delta(A,0) = \{A, B, C, D\}$

$\delta(A,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A),1))$

$= \epsilon\text{-closure}(\delta((A,B,D),1))$

$= \epsilon\text{-closure}(\delta(A,1) \cup \delta(B,1) \cup \delta(D,1))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup D)$

$= \epsilon\text{-closure}(D)$

$\delta(A,1) = \{D\}$

$\delta(B,D) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(B),0))$

$= \epsilon\text{-closure}(\delta((B,D),0))$

$= \epsilon\text{-closure}(\delta(B,0) \cup \delta(D,0))$

$= \epsilon\text{-closure}(C \cup D)$

$\delta(B,0) = \{C,D\}$

$\delta(B,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(B),1))$

$= \epsilon\text{-closure}(\delta((B,D),1))$

$= \epsilon\text{-closure}(\delta(B,1) \cup \delta(D,1))$

$= \epsilon\text{-closure}(\phi \cup D)$

$\delta(B,1) = \{D\}$

$\delta(C,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(C),0))$

$= \epsilon\text{-closure}(\delta(C,0))$

$\delta(C,0) = \phi$

$\delta(C,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(C),1))$

$= \epsilon\text{-closure}(\delta(C,1))$

$= \epsilon\text{-closure}(B)$

$\delta(C,1) = \{B,D\}$

$\delta(D,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(D),0))$

$= \epsilon\text{-closure}(\delta(D,0))$

$\delta(D,0) = \{D\}$

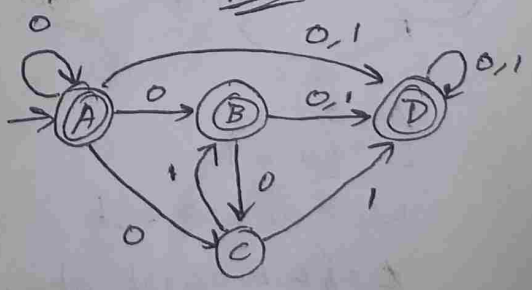
$\delta(D,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(D),1))$

$= \epsilon\text{-closure}(\delta(D,1))$

$\delta(D,1) = \{D\}$

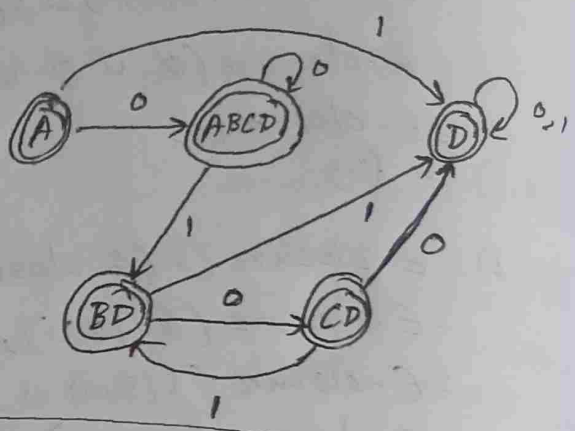
δ	0	1
→ (A)	{A,B,C,D}	{D}
(B)	{C,D}	{D}
C	ϕ	{B,D}
(D)	{D}	{D}

NFA



DFA

	0	1
→ A	[ABCD]	[D]
[ABCD]	[ABCD]	[BD]
[D]	[D]	[D]
[BD]	[CD]	[D]
[CD]	[D]	[BD]



Finite Automata
- without o/p → DFA, NFA
- with o/p → Melay, ~~Moore~~ Moore.

Formal definition of Melay machine and Moore machine

It is a 6-tuple $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Q - Non-Empty finite set of states

Σ - set of i/p alphabet

δ - transition function $\rightarrow Q \times \Sigma \rightarrow Q$

q_0 - initial state

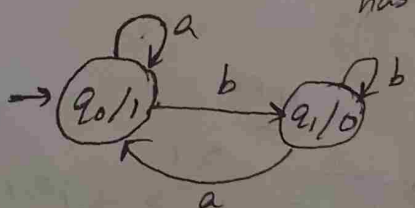
Δ - output alphabet.

λ - output function $Q \times \Sigma \rightarrow \Delta$

Examples of Moore machine and ~~Mealy~~ Melay machine

Moore: only state has o/p

Melay: along with i/p, o/p is there



Moore

$\Delta = 0, 1$

$\Sigma = \{a, b\}$

$\omega_1 = ab, ba, a, b$

$L = \{a, b, aa, bb, aba \ldots\}$

aba → 1101

initially q_0 so 1

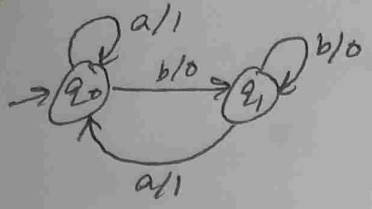
$q_0 \xrightarrow{a} q_0$ o/p 1

$q_0 \xrightarrow{b} q_1$ o/p 0

$q_1 \xrightarrow{a} q_0$ o/p 1

ababa → 110101

Length of o/p string is 1 + length of i/p



Melay

$\Sigma = \{a, b\}$	ab	ababab
$\Delta = \{0, 1\}$	10	101010

→ Construct moore machine $\Sigma = \{a, b\}$ and prints 1 as o/p for every occurence of ab as substring.

Ans Moore machine is a 6 tuple $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

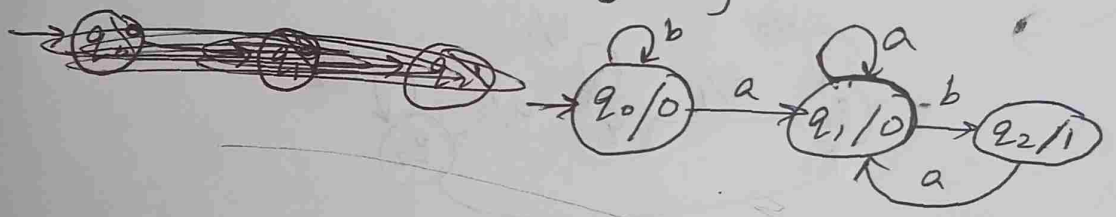
Q - ϕNon-Empty finite set of states.

$\Sigma = \{a, b\}$

$q_0 = q_0$

$\Delta = \{1, 0\}$

$L = \{ab, ab, aab, abab \ldots\}$

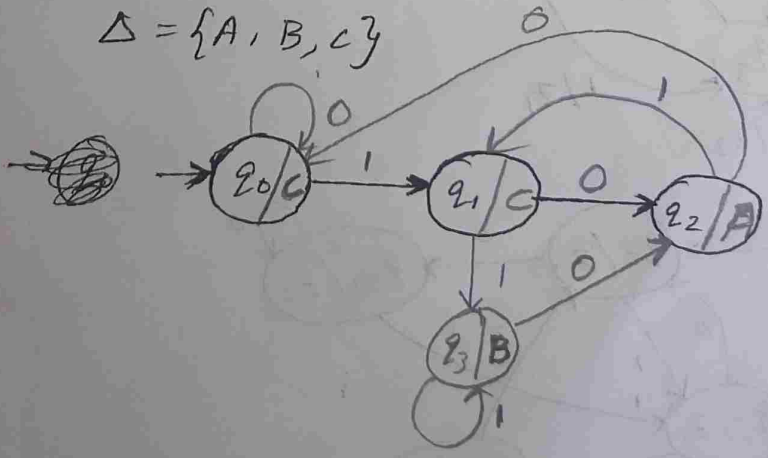


→ Construct moore machine $\Sigma = \{0, 1\}$ and print A whenever there is an occurence of 10. Print B whenever there is an occurence of 11 and print C otherwise

Ans Moore machine is a 6 tuple $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

$\Sigma = \{0, 1\}$

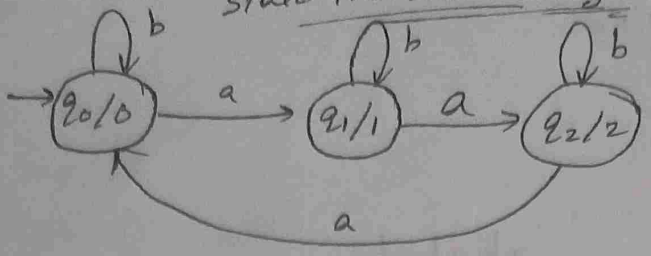
$\Delta = \{A, B, C\}$



101100
CCACBAC

9.10.23

state transition diagram



state transition table

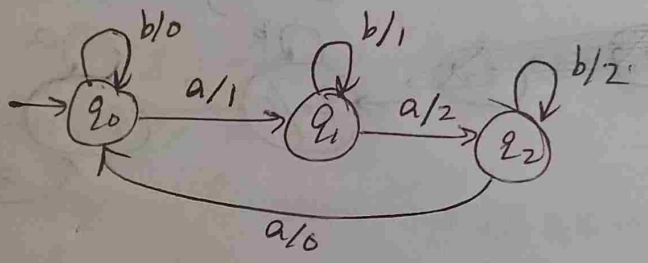
Moore machine

	a	b	Δ
→ q_0	q_1	q_0	0
q_1	q_2	q_2	1
q_2	q_0	q_2	2

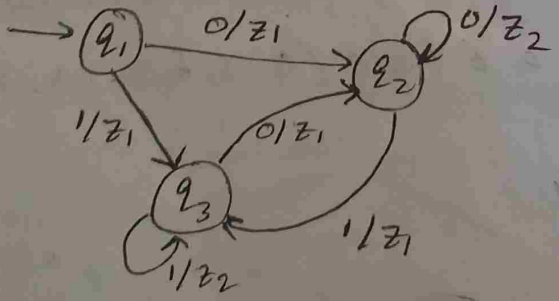
Melay machine

	a	b
→ q_0	$(q_1, 1)$	$(q_0, 0)$
q_1	$(q_2, 2)$	$(q_1, 1)$
q_2	$(q_0, 0)$	$(q_2, 2)$

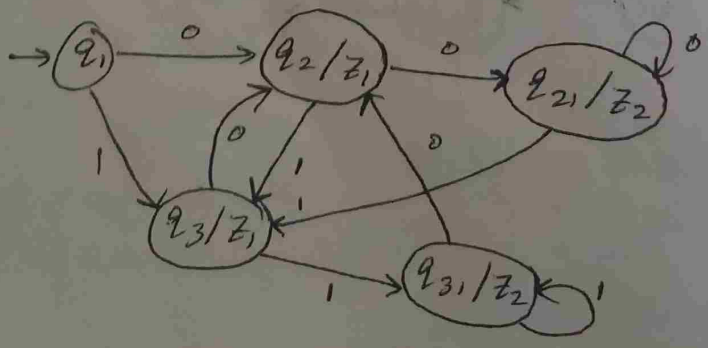
Construct moore machine which accepts no. of a's divisible by 3 which accepts $\Sigma = \{a, b\}$



Convert given Melay machine to moore machine.



Ans



State transition table Moore

	0	1	Δ
→ q_1	q_2	q_3	0
q_2	q_{21}	q_3	z_1
q_{21}	q_{21}	q_3	z_2
q_3	q_2	q_{31}	z_1
q_{31}	q_2	q_{31}	z_2

Melay

	0	1
→ q_1	(q_2, z_1)	(q_3, z_1)
q_2	(q_2, z_2)	(q_3, z_1)
q_3	(q_2, z_1)	(q_3, z_2)

- Construct Melay machine to construct 2's complement for given binary number.

10100

Regular expressions

operations

Union: $L1 = (1+0)\cdot(1+0) = \{00, 01, 11, 10\}$

$L2 = \{\epsilon, 100\}$ $L1 \cup L2 = \{\epsilon, 00, 01, 11, 10, 100\}$

Concatenation:

Kleene closure:

Properties

Closure

$r1$ & $r2$ are Regular Expressions (RE) then

$r1^*$ is RE

$r1 + r2$ is RE

$r1 \cdot r2$ is RE

Closure Laws

$r^+ = r \cdot r^* = r^* r$

$r^* = \epsilon + r + rr + rrr + \ldots$

$r \cdot r^* = r + rr + rrr + rrrr + \ldots$

$r^* = r^+ + \epsilon$

Associativity

$r1 + (r2 + r3) = (r1 + r2) + r3$

$r1 \cdot (r2 \cdot r3) = (r1 \cdot r2) \cdot r3$

* Associativity does not hold for kleene closure(*) because it is unary operator.

Identity

$r + x = r$ $\quad x = \phi$

$r \cup \phi = r$ $\quad \phi$ is identity for '+'

$r \cdot x = r$ $\quad x = \epsilon$

$r \cdot \epsilon = r$ $\quad \epsilon$ is identity for '.'

Annihilator

$r \cdot x = x \qquad r \cdot \phi = \phi$

Commutative

$r1 + r2 = r2 + r1$

$r1 \cdot r2 \neq r2 \cdot r1$

Identities

$\phi + r = r$

$\phi \cdot r = r \cdot \phi = \phi$

$\epsilon \cdot r = r \cdot \epsilon = r$

$\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

$r + r = r$

$r^* \cdot r^* = r^*$

$r \cdot r^* = r^* \cdot r = r+$

$(r^*)^* = r^*$

$\epsilon + r \cdot r^* = r^* = \epsilon + r^* \cdot r$

$(p \cdot q)^* \cdot p = p \cdot (q \cdot p)^*$

$(p+q)^* = p^* \cdot q^* = (p^* + q^*)^*$

$(p+q) \cdot r = p \cdot r + q \cdot r$

$r \cdot (p+q) = r \cdot p + r \cdot q$

- Construct a regular expression which accepts all the strings ~~which accept~~ $\Sigma = \{a, b\}$ of even length.

Ans $\Sigma = \{a, b\}$

$L = \{\epsilon, aa, ab, ba, bb, aaaa, aabb, bbaa, bbbb$

$a(a+b) + b(a+b)$ Regular

$(a+b)(a+b)$ ⇒ for length 2

$((a+b)(a+b))^*$

- write regular expression which accepts all strings $\Sigma=\{a,b\}$ of odd length.

Ans

$\Sigma=\{a,b\}$

$L=\{$

$(a+b)((a+b)(a+b))^*$

- Construction RE which accepts exactly 2 a's and any no-of b's.

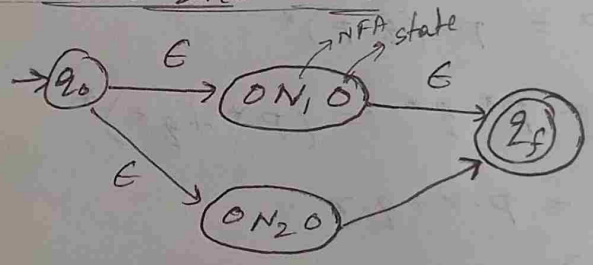
$\Sigma=\{a,b\}$

$L=\{aa, baa, aab, \ldots\}$

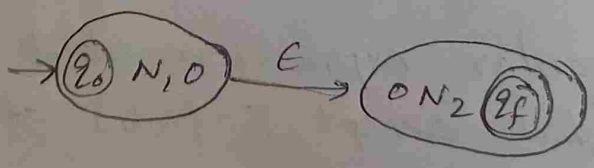
$b^*ab^*ab^*$

Regular expressions and NFA

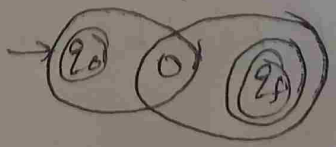
1) r_1+r_2 (union) (either r_1 or r_2)



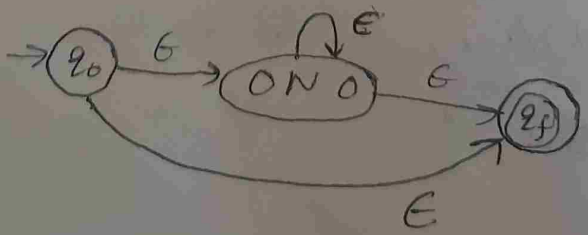
2) $r_1.r_2$ Concatenation

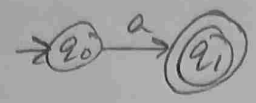


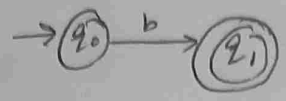
(or)



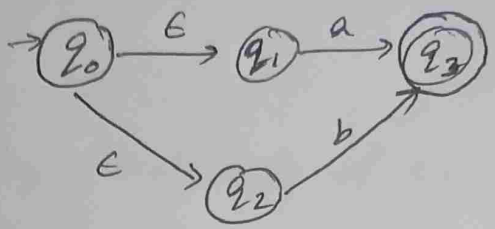
3) r_1^*



a → 

b → 

ab → 

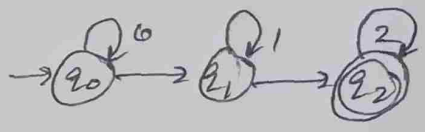
a+b → 

Regular expressions and NFA.

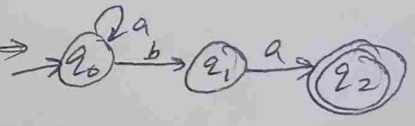
→ If r_1 & r_2 are 2 RE then to construct NFA for RE $r_1 + r_2$

→ If r_1 & r_2 are 2 RE then to construct NFA for $r_1 . r_2$

→ If r_1 & r_2 are 2 RE then to construct NFA for r_1^*

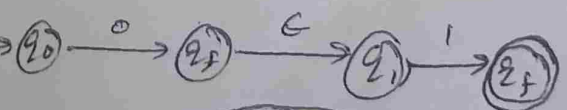
Construct NFA for ① $0^* 1^* 2^*$ 

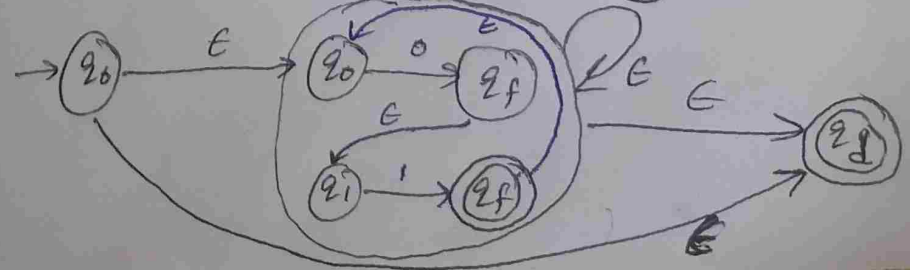
② $(0+1)^*$

③ $a^* b a$ ⇒ 

→ Construct NFA with ϵ for ① $(0^*1 + 1^*01)10$

② $(1^*0 + 0^*101)^* 01$

01 ⇒ 

$(01)^*$ 

Construct regular expression from finite Automata

(Arden's Theorem)

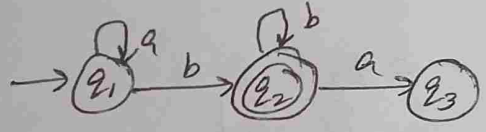
If P, Q, R are three RE and P is not null 'ε' then if $R = Q + RP$ then we can write it as

$R = QP^*$

Rules:

1. For every initial state of RE add (null) ε.
2. Final state always gives you R.E.
3. If more than one final state add all R.E of final states

Construct R.E for given FA



Sol As q_1 is initial state, add ε

$q_1 = q_1 a + \epsilon$ $\Rightarrow R = RP + Q$, $q_1 = q_1 a + \epsilon$ $\Rightarrow R = QP^*$, $q_1 = \epsilon \cdot a^*$, $q_1 = a^*$

$q_2 = q_2 b + q_1 b$

$q_3 = q_2 a$

$R = Q + RP$

$q_2 = q_1 b + q_2(b)$ (R = q_2, Q = $q_1 b$, R = q_2, P = b)

$\Rightarrow q_2 = (q_1 a + \epsilon) b + q_2 b$

$q_2 = q_1 b\, b^*$ (R = q_2, Q = $q_1 b$, $P^* = b^*$)

$= (q_1 a + \epsilon)$

$q_2 = (a^* a + \epsilon) b + q_2 b$

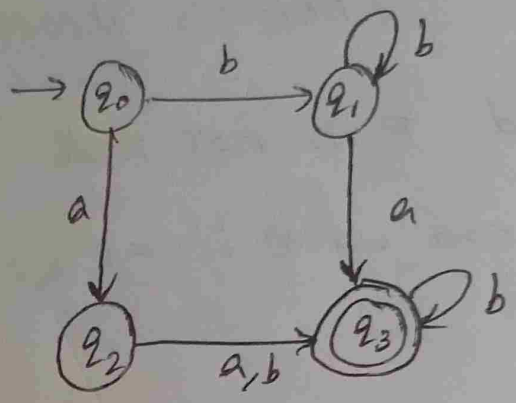
$q_2 = (a^+ + \epsilon) b + q_2 b$

$q_2 = a^* b + q_2 b$

$R = Q + RP \Rightarrow R = QP^*$

$\Rightarrow q_2 = a^* b\, b^*$

. Convert give FA to RE



Ans q_0 is initial state. add ϵ

$$q_0 = \epsilon$$

$$q_1 = q_0 b + q_1 b$$

$$q_2 = q_0 a$$

$$q_3 = q_2(a+b) + q_1 a + q_3 b$$

$$\Rightarrow q_1 = \epsilon \cdot b + q_1 b$$

$$q_1 = b + q_1 b$$

$R = Q + RP \rightarrow R = QP^*$

$$q_1 = b b^*$$

$$\Rightarrow q_2 = \epsilon \cdot a$$

$$q_2 = a$$

$$\Rightarrow q_3 = a(a+b) + bb^*a + q_3 b$$

$$q_3 = aa + ab + bb^*a + q_3 b$$

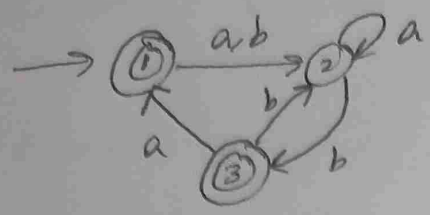
$R = Q + RP$

$R = QP^*$

$$q_3 = (aa + ab + bb^*a)\, b^*$$

$$q_3 = aab^* + abb^* + bb^*ab^*$$

• Convert given FA to RE



Any 1 is initial state, add ϵ

$\Rightarrow 1 = 3a + \epsilon$

$2 = 1a + 1b + 2a + 3b$

$\Rightarrow 2 = 1(a+b) + 2a + 3b$

$\Rightarrow 3 = 2b$

$\Rightarrow 1 = 2ba + \epsilon$

$\Rightarrow 2 = 1(a+b) + 2a + 2bb$

$2 = (2ba+\epsilon)(a+b) + 2a + 2bb$

$2 = 2baa + 2bab + (a+b) + 2a + 2bb$

$2 = 2(baa + bab + a + bb) + (a+b)$

$R = RP + Q$

$R = QP^*$

$2 = (a+b)(baa + bab + a + bb)^*$

$2 = (a+b)(ba(a+b) + (a+bb))^*$

$2 = (a+b) \cdot ba(a+b)^* \cdot (a+bb)^*$

$3 = 2b$

$3 = ((a+b)\,ba\,(a+b)^*(a+bb)^*)\,b$

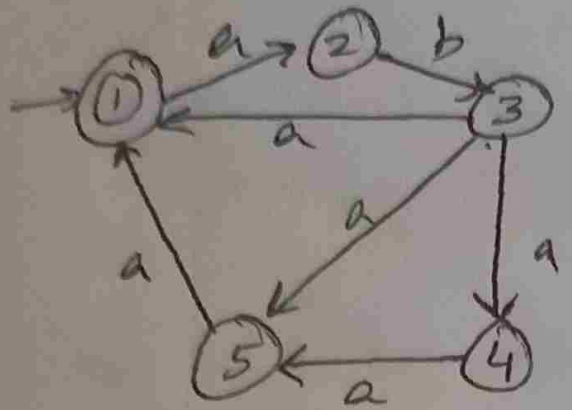
$1 = ((a+b)\,ba\,(a+b)^*(a+bb)^*)\,ba + \epsilon$

$1 \cup 3 = ((a+b)ba(a+b)^*(a+bb)^*)\,ba + \epsilon + ((a+b)\,ba(a+b)^*(a+bb)^*)\,b$

1)

2)

3)

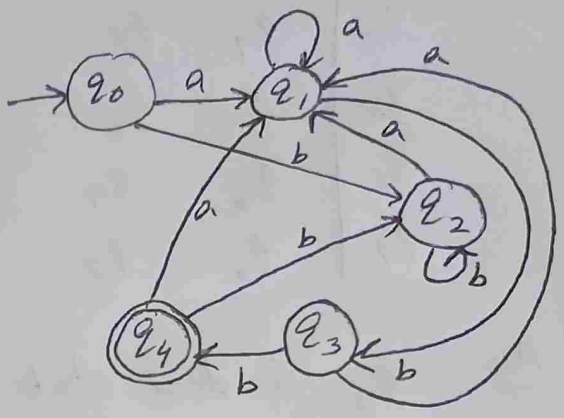


Minimisation of FSM (Finite state machine) (only applicable for DFA)

(Everyone will get some FSM)

- Convert the given DFA to its minimised form.

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
(*q_4)	q_1	q_2



Can be done in 2 methods

Table filling method (Hard)

Partitioning method. (Easy)

Ans Done by finding ~~equitent~~ equivalent sets/states.

$\pi_0 = \{\{q_0, q_1, q_2, q_3\}$ (Non Final set) $\{q_4\}\}$ Final set.

$\pi_1 =$

	a	b
q_0	q_1	q_2
q_1	q_1	q_3

q_0, q_1 → Non final set; q_2, q_3 → both are non-final set

	a	b
q_0	q_1	q_2
q_2	q_1	q_2

q_0 same set, q_2 same set

X	a	b
q_0	q_1	q_2
q_3	q_1	q_4

q_0 same, q_3 not same

q_2 – NFS

q_4 – FS

So q_0, q_3 separate.

$\pi_1 = \{\{q_0, q_1, q_2\}\ \{q_4\}\ \{q_3\}\}$

X	a	b
q_0	q_1	q_2
q_1	q_1	q_3

→ diff. so separate q_1

$\pi_2 = \{\{q_0, q_2\}\ \{q_4\}\ \{q_3\}\ \{q_1\}\}$

	a	b
q_0	q_1	q_2
q_2	q_1	q_2

→ same

$\pi_3 = \{\{q_0, q_2\}, \{q_4\}, \{q_3\}, \{q_1\}\}$

(2).

	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

q_3 becomes dead state bcuz no state has o/p to q_3.

Ans q_3 is dead state so we can remove it

$\pi_0 = \{\{q_0, q_1, q_4, q_5, q_6, q_7\}, \{q_2\}\}$

X	0	1
q_0	q_1	q_5
q_1	q_6	q_2

→ diff set

✓	0	1
q_0	q_1	q_5
q_4	q_7	q_5

X	0	1
q_0	q_1	q_5
q_5	q_2	q_6

diff

✓	0	1
q_0	q_1	q_5
q_6	q_6	q_4

same set

X	0	1
q_0	q_1	q_5
q_7	q_6	q_2

→ diff set

q_1 q_5 q_7

	0	1
q_1	q_6	q_2
q_5	q_2	q_6

→ diff bcuz separated

	0	1
q_1	q_6	q_2
q_7	q_6	q_2

same same

$\pi_1 = \{\{q_0, q_4, q_6\}\{q_1, q_7\}\{q_5\}\{q_2\}\}$

✓	0	1
q_0	q_1	q_5
q_4	q_7	q_5 ✓

X	0	1
q_0	q_1	q_5
q_6	q_6	q_4

✓	0	1
q_1	q_6	q_2
q_7	q_6	q_2 ✓

$\pi_2 = \{\{q_0 q_4\}\{q_6\}\{q_1, q_7\}\{q_5\}\{q_2\}\}$

✓	0	1
q_0	q_1	q_5
q_4	q_7	q_5 ✓

$\pi_3 = \{\{q_0 q_4\}\{q_6\}\{q_1 q_7\}\{q_5\}\{q_2\}\}$