TOC

Lec1, 98.

Theory of Computation

Module - I : Finite Automata

fundamentals : Alphabet, Strings, language, Operations.
Introduction to finite automata: The Central Concepts of automata theory deterministic finite automata, non-deterministic finite automata, an application of finite automata, finite automata with and without Epsilon transitions. Conversion of NFA to DFA, Moore and Mealy Machines.

---

why to study TOC :

* Computation in theoritical way
* Behind every practical there will be theory.
  Ex: atom bomb
  Newton Law $E=mc^2$

Basic pillars of TOC are

L A G → Grammar
↓ ↓
Language Automata

→ Before Language we will learn about symbol
↓
Smallest unit of language.
(a)
{Building block}

Example : {a, b, c, 0, 1, 2, 3, ...}

next is Alphabet denoted by Σ
Alphabet is nothing but finite set of Symbol
$\Sigma(a,b)$

next String → collection of alphabet. Sequence of alphabet

$\Sigma(a, b)$

a, b, aa, ab, ba, bb

Length of the string: Length of 2 {aa, ab, ba, bb}

Length of 3 {aaa, aab, aba, abb, baa, bab, bba, bbb}

Language: collection of strings.

Ex: write (String) a (String) 'c' program (String)

Language = { Prg1, Prg2 ... }

L_1 = string of length 3. {aaa, aab, aba, abb, baa, bab, bba, bbb} — 8

* Language can be infinite also

L_2 = string start with a End with a

{aa, aaa, aaaa .....}

↳ This language is infinite

* length of language is 0 i.e., $\Sigma^0 = \varepsilon$

Operations on Languages

① Union of two Languages: Let L_1 & L_2 be the two languages. then

$$L_1 \cup L_2 = \{x, y / x \in L_1 \text{ \& } y \in L_2\}$$

② Concatenation of two languages: Let L_1 & L_2 be two languages then

$$L_1 \cdot L_2 = \{x.y / x \in L_1 \text{ and } y \in L_2\}$$

③ closure of a language: Let L be language then

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Positive closure of a language: Let L be the language. then

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

Examples (or) Points to remember:

Union : No combination

Concatination : Combination

closure : zero (or) more repetations

Positive closure : one (or) more repetations

① $L_1 \cup L_2$ = $L_1 = \{00, 01\}$ $L_2 = \{00, 110, 11\}$

$= \{00, 01, 110, 11\}$ → No combination

② $L_1 \cdot L_2 = \{0000, 00110, 0011, 0100, 01110, 0111\}$

↳ combination

③ $L^* = \{ab\}$

$L^2 = L^1 \cdot L^1 = \{ab\ ab\}$

$L^3 = L^1 \cdot L^1 \cdot L^1 = \{ab\ ab\ ab\}$

$L^1 = \{ab\} = L$

$L^4 = \{ab\ ab\ ab\ ab\}$

$L^0 = \{\epsilon\}$

$L^* = \bigcup_{i=0}^{\infty} L^i$

$= L^0 \cup L^1 \cup L^2 \cup L^3 \cup \ldots L^\infty$

$= \{\epsilon, ab, abab, ababab \ldots\ldots \infty\}$

L^* is zero or more repetations.

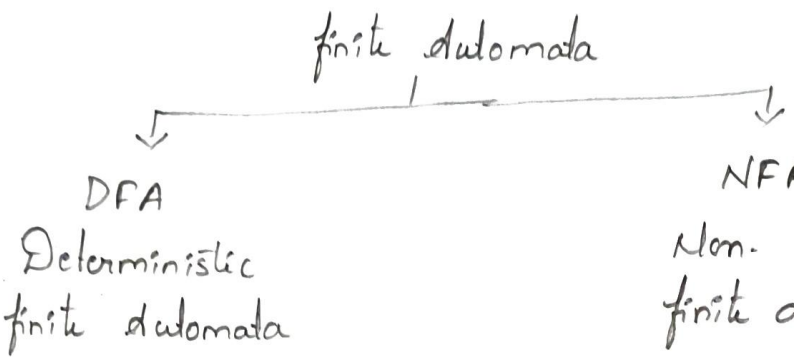
④ $L^+ = \{ab, abab, ababab, abababab \ldots\ldots\}$

↳ one or more repetatims.

Tuple

Introduction to Finite automata:

Mathematical model of abstract machine with finite no of states is called finite automata.



NB: Every FA is DFA.

* FA has five tuple notation.

State → denoted by circle (q_0)

Transition → Edge b/w two states. & it is a directed edge.

Initial state → State with an edge from no state

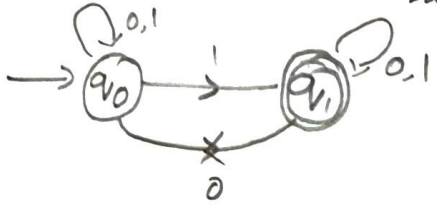
Final state → Double circled state (q_F)

NB: Finite automata will have one initial state and multiple final states.

Finite automata is Represented in two ways

↳ ① Directed Graph with states

② Transition table.



Tuple of 5 elements

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

$Q \Rightarrow$ set of all states

$\Sigma \Rightarrow$ input symbols

$\delta \Rightarrow$ Transition function

$q_0 \Rightarrow$ Initial state

$F \Rightarrow$ Final state

Some basic Operations on strings.

① Length of string : $|\omega|$ $\omega = 'abc'$

↳ No. of symbols, it can never be minus.

ϵ, ϕ $|\epsilon| = 0$

② Concatenation of string :

$\omega_1 = 'abc'$

$\omega_2 = 'ba'$

$\omega_1 \cdot \omega_2 = abcba$

$\omega_2 \cdot \omega_1 = baabc$

$\omega_1 \cdot \omega_2 \neq \omega_2 \cdot \omega_1$

$\omega_1 \cdot \epsilon = \epsilon \cdot \omega = \omega$

③ Reverse of a string : $\omega = 'abc'$

$\omega^R = cba$ $|\omega| = |\omega^R|$

④ Prefix & sufix : $\omega = abaa$

$P(\omega) = \{\epsilon, a, ab, aba, abaa\}$

$S(\omega) = \{\epsilon, a, aa, baa, abaa\}$

⑤ Sub-string : $\omega = abbcd$

abd ✗ bc ✓

abbc ✓

bcd ✓

cba ✗

Note : If length of string is 'n' and unique symbols then total number of sub string possible.

$1 + 2 + \cdots + n$

$= \frac{n(n+1)}{2} + 1$

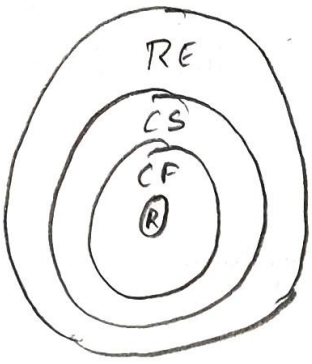
AAAA

GATE

0	1	2	3	4
∈	G A T E	GA AT TE	GAT ATG	GATG

Chomsky Hierarchy :

Grammar	Language	Automata
Type 0 → unrestricted Grammar	Recursive Enumerable Language	Turing Machine
Type 1 — Context Sensitive "	Context Sensitive Lang	Linear Bounded Au
Type 2 — Context free Grammar	Context Free Lang	Push Down Automata
Type 3 — Regular Grammar	Regular Lang	finite automata



Type 3 → Regular Language

$N \rightarrow T$

$N \rightarrow TN$

$X \rightarrow \epsilon$

$X \rightarrow a \mid aY$

T ↓ (NT)

Type 2 Grammar

$N \rightarrow (T \cup N)^*$

$S \rightarrow Xa$

NT T $X \rightarrow aX$

$X \rightarrow abc$

$X \rightarrow \epsilon$

Type 1 Grammar

$\alpha N \beta \rightarrow \alpha Y \beta$

$S \rightarrow \epsilon$

Type 0 Grammar $\alpha \rightarrow \beta$ (CB → DB, AD → Db)

↓ $S \rightarrow AcaB$

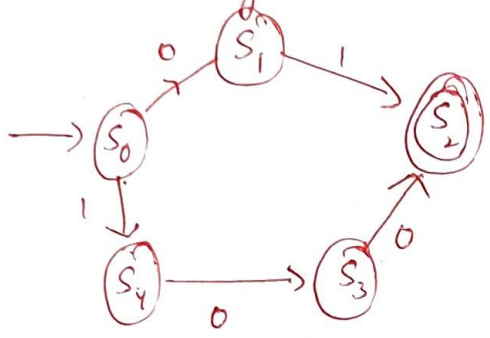
$BC \rightarrow acB$

Acceptance of strings & Languages:

The strings and languages can be accepted by a finite automata, when it reaches to final state.

Preferred notations for describing automata:

① Transition diagram:



2nd Example.

Transition diagram for input aabb.

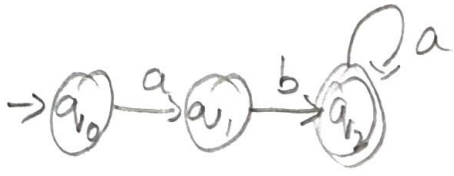
start state ⇒ S_0 $\Sigma = \{a, b\}$

final state ⇒ S_4 No. of states = S_0, S_1, S_2, S_3, S_4

↓ Intermediate states

② Transition table: Tabular representation of automata

	a	b
q_0	q_1	–
q_1	–	q_2
q_2	q_2	–



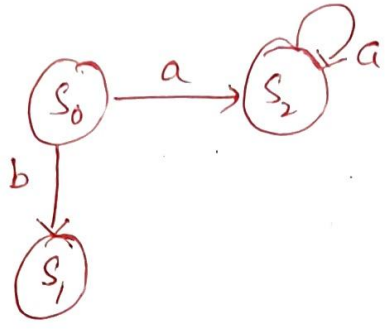
rows of table corresponds to states

columns of table " to inputs

Deterministic finite Automata (DFA):

The finite automata is called DFA if there is only one Path on a specific input from current state to next state.

for Example DFA can be shown as



Definition of DFA:

5 - tuple notation

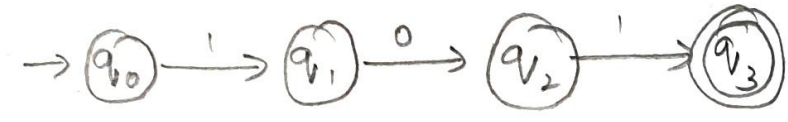
$\{Q, \Sigma, \delta, q_0, q_F\}$

for Example, $q_1 = \delta(q_0, a)$

Example Problems of FA (or) DFA

(1) Design a FA which accepts the only input 101 over the input $\Sigma = \{0, 1\}$.

Sol:

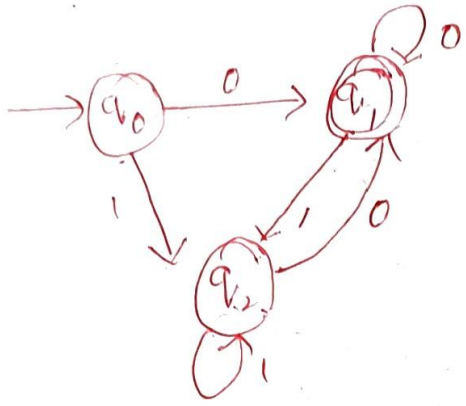


(2) Design a FA which checks whether the given binary number is Even.

Sol: binary number is made of '0' & '1'

if any binary number ends with 0 then it is Even

if any binary number ends with 1 then it is odd.

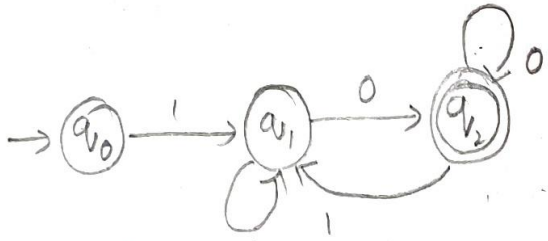


1000 → accept.

101 → Reject

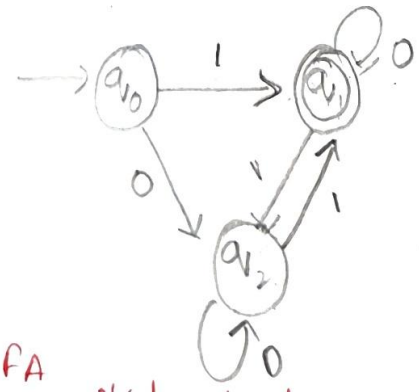
(3) Design FA which accepts only those strings which start with 1 and end with 0

Sol:



(4) Design FA which accepts odd no. of 1's and any no. of 0's

Sol:



1111

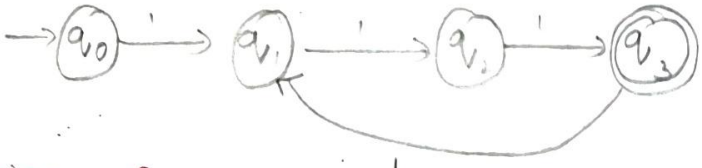
111

(5) Design FA which checks weather given unary number is divisible by 3

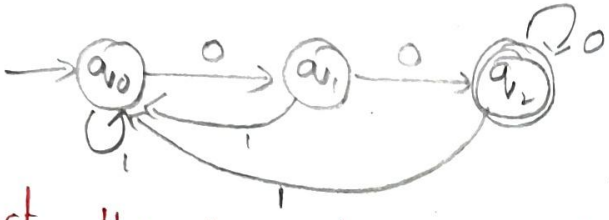
Sol: Unary number is made up of ones. unary number of 3 can be written as 111. Unnary number of 5 can be written as 11111

Unary number divisible by 3

1, 11, 111, 1111, 11111, ...



(6) Design FA to accept the string which always ends with 00

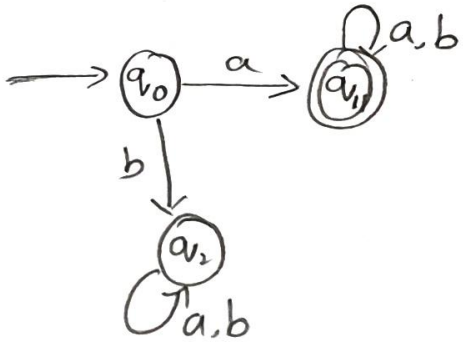


(7) Construct the transition graph for a FA which accepts a Language L over $\Sigma\{0,1\}$ in which every string start with 0 and Ends with 1

(8) L_1 = Set of all strings which starts with 'a' over alphabet $\Sigma(a,b)$

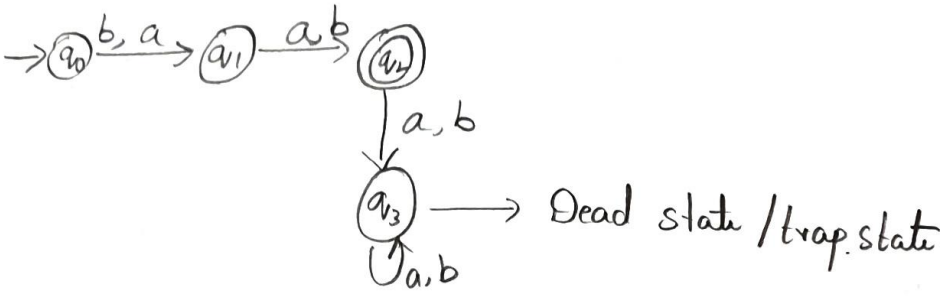
$L_1 = \{a, aa, aaa, ab \ldots\}$

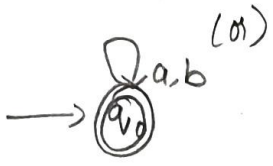
Sol:



(9) Construct DFA, that accepts set of all strings over $\{a, b\}$ of length 2

Sol: $L = \{aa, ab, ba, bb\}$





$L = \{aa, ab, ba, bb\}$

↳ finite automata accepts all the strings in the lang^ and also all the strings which are not in the language. So the finite automata is not correct.

(10) Construct a minimal DFA, which accepts set of all strings over $\{0, 1\}$, which when interpreted as binary number is divisible by 2

Sol:

1 2

$1 \times 10 + 2 = 12$

12 3

$12 \times 10 + 3 = 123$

1456

$1 \times 10 + 4 = 14$

$14 \times 10 + 5 = 145$

$145 \times 10 + 6 = 1456$

base 1 ← unary - 0

base 2 ← binary - 0, 1

base 3 ← trinary → 0, 1, 2

⋮

base 10 ← Decimal → 0, 1, 2 ... 9

⋮

base 16 ← Hexadecimal → 0, 1, 2 .. 9, A .. F

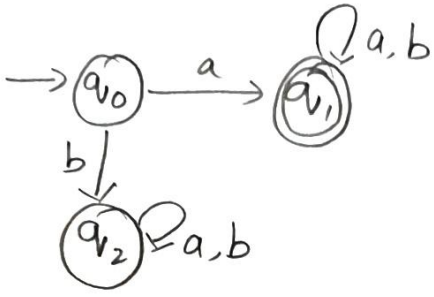
(11) L_1 = accepts string starting with a

L_2 = accepts string contains a

L_3 = accepts string Ending with a

Sol:

11.1

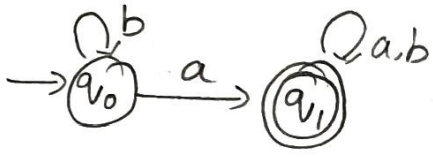


Dead state

$L_1 = \{a, ab, aab, aabb \ldots\}$

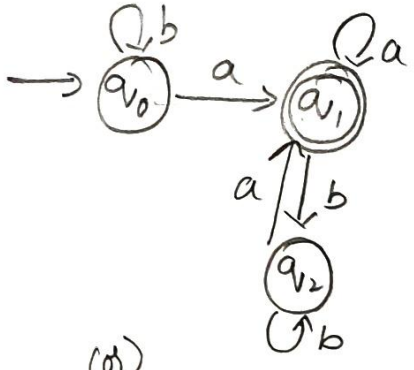
* start with something

11.2



$L_2 = \{bab, aab, aaa \ldots\}$

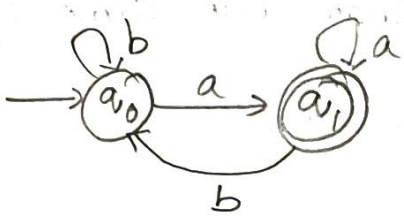
11.3



$L_3 = \{a, aba, aaa, babba \ldots\}$

aaaba

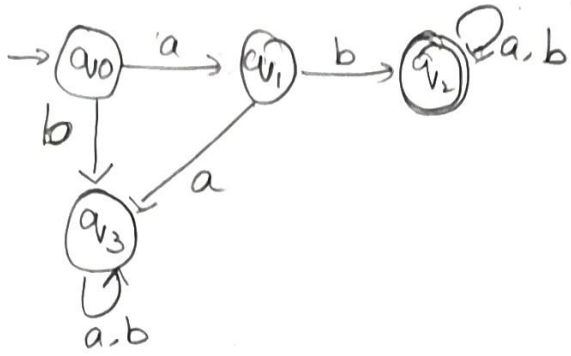
(or)



Come back if you find Ending with

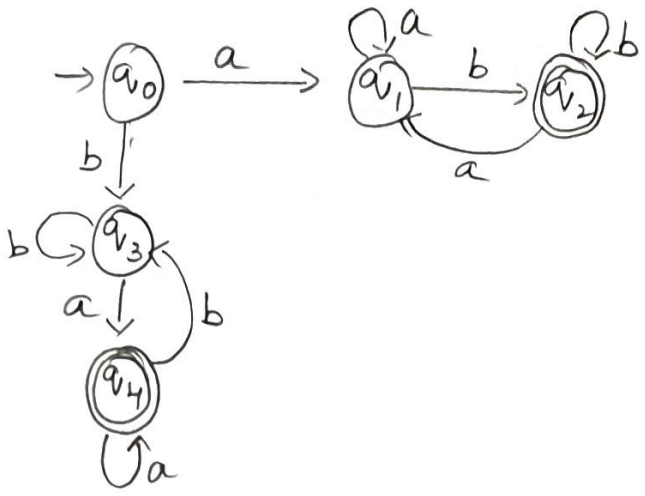
(12) Construct a DFA which accepts all string over {a, b} which start 'ab' → a followed by b (always)

$L = \{ab, abaabb \ldots\}$



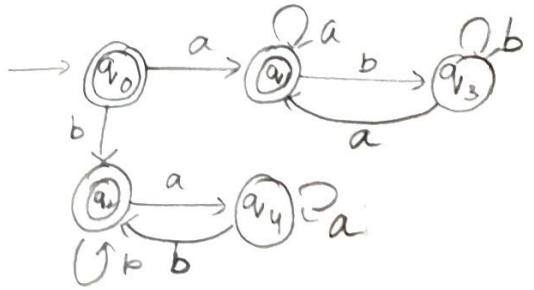
(13) Construct a DFA which accepts all string starts and End with a different symbols.

$L = \{ab, aabba, ba, baba \ldots\}$



(14) Construct a DFA which starts and Ends with same Symbol

$L = \{\epsilon, a, b, aa, bb, aaa, bbb, aaba, abbb \ldots\}$

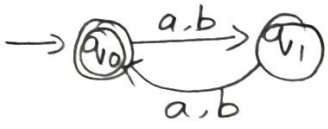


(15) Complement of DFA * which happens only in DFA.

(a) Set of all strings of Even length.

0, 2, 4, 6, 8, ----

L_1

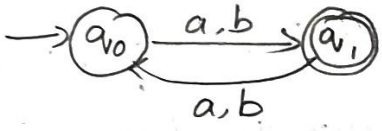


(b) Set of all strings of odd length

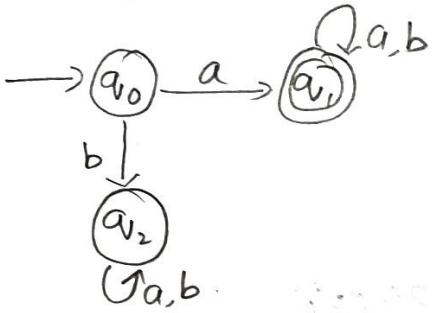
$L^+ = \overline{L_2}$

1, 3, 5, 7 ----

L_2

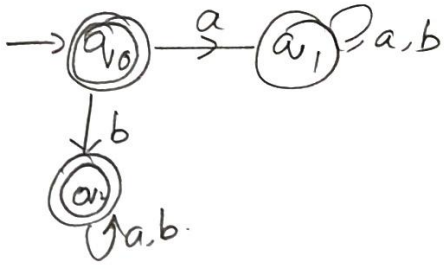


(c) Starting with 'a' $\Sigma(a,b)$



(d) not starting with a $\Sigma(a,b)$.

$L = \{\epsilon, b, ba, bb ---\}$



* Complementation is only for DFA not for NFA

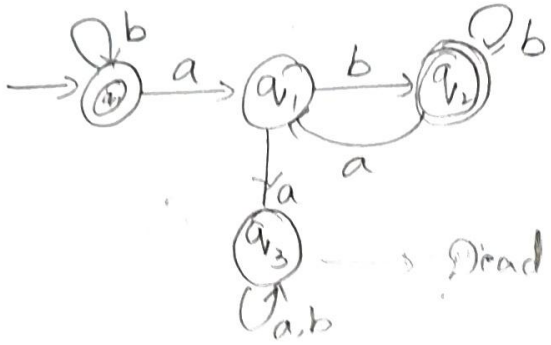
* $(Q, \Sigma, \delta, q_0, F)$

$\Downarrow$

$(Q, \Sigma, \delta, q_0, q_0 - F)$

(16) Construct a minimal DFA which accepts set of all strings over {a, b} in which every a followed by a b.

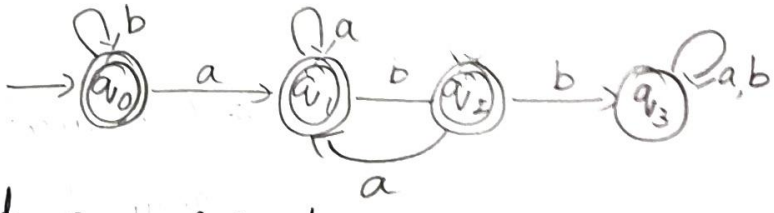
Sol: $L = \{\epsilon, ab, abab, bbb\ldots\}$ $\quad w \in \{a,b\}^*$



(17) Construct a minimal DFA which accepts set of all strings over {a, b} in which every 'a' should never be followed by bb

Sol: $L = \{\epsilon, b, bb, \ldots a, aa, aaa \ldots ab, aab \ldots\}$

abaaab

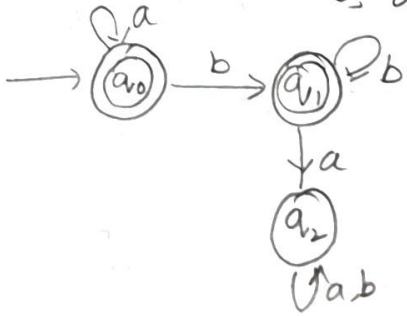


(18) Construct a minimal DFA which accepts

$L = \{a^n b^m / n, m >= 0\}$ if b's start no a should be seen.

$L = \{\epsilon, a, aa, aaa \ldots b, bb, bbb \ldots\}$

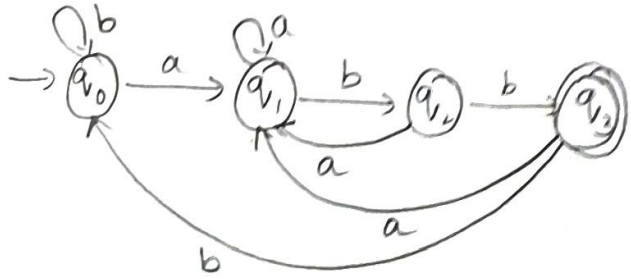
$ab, aab \ldots$



(19) Design a DFA to accept string of a's and b's ending with 'abb' over $\Sigma(a,b)$

Sol:

$L = \{abb, aabb, babb, aaabb, \ldots\}$



Divisibility

DFA which accepts all strings over an alphabet $\Sigma=(0,1)$ whose binary integers divisible by 3.

$L = \{0, 11, 110, 1001, 1100, 1111 \ldots\}$

0 3 6 9 12 15

$\rightarrow (q_0)$

$0/3 = 0$

$L = \{w \mid w$ has even no. of a's and even no. of b's$\}$

Sol: $L = \{\underset{0}{\in}, aa, bb, abab, aabb, bbaa, baba, abba, baab \cdots\}$

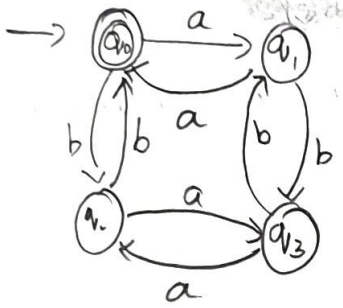


abb a

abba

Non-Deterministic finite automata.

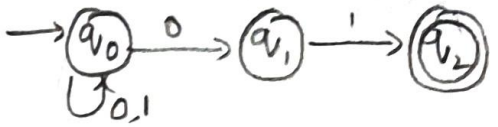
① string starts with '0,1' $\Sigma = \{0, 1\}$ min string + 1 = 3 states

sol: $L = \{01, 010, 011, 0100, 0101, 0110 \ldots\}$

$\delta - Q \times \Sigma = 2^Q$

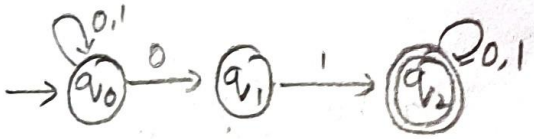
② String ending with 01 $\Sigma = \{0, 1\}$

sol: $L = \{01, 001, 101 \ldots\}$



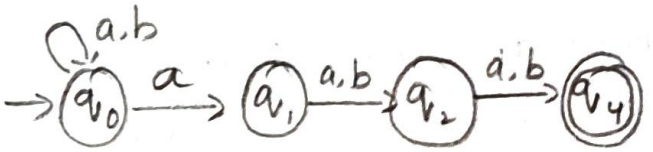
③ String contains 01 $\Sigma = \{0, 1\}$

sol: $L = \{01, 0011, 1010 \ldots\}$



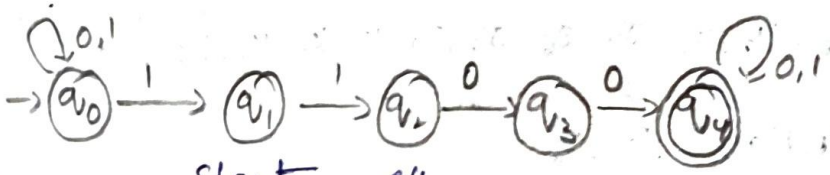
④ Third symbol from right End is 'a' $\Sigma = \{a, b\}$

sol: $L = \{\_\_a\_\_$



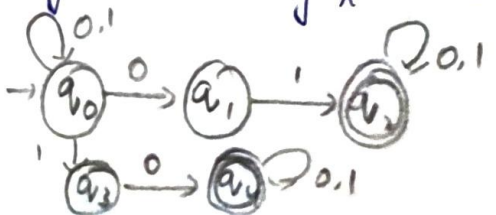
⑤ Double '1' is followed by double '0' $\Sigma = \{0, 1\}$

sol: $L = \{1100,$



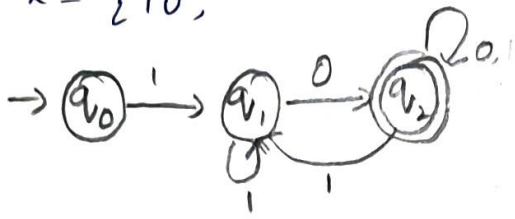
⑥ String Starts with either 01 (or) 10. $\Sigma = \{0, 1\}$

sol:



(7) Starting with '1' Ends with '0'

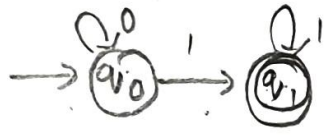
Sol: $L = \{10,$ 101010



(8)

$L = \{0^m 1^n / m \geq 0 \text{ and } n \geq 1\}$ ✓

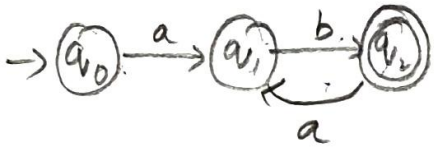
Sol: Let $m=0, n=1$; $m=1, n=2 \Rightarrow 011$; $m=2; n=3 \Rightarrow 00111$



$L = \{1, 011, 00111 \ldots\}$

(9) $L = (ab)^n$ with $n \geq 1$ ✓ $(ab)^0 = 1$ $(ab)^1 = ab$

Sol: Let $n=1 \Rightarrow ab$; Let $n=2 \Rightarrow abab$; Let $n=3 \Rightarrow ab\ ab\ ab$

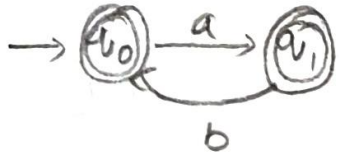


(10) $L = (ab)^n$ with $n \geq 0$

$L = \{ab, abab, ababab \ldots\}$ abba Reject ababab accept

Sol: Let $n=0 \Rightarrow \phi$; $n=1 \Rightarrow ab$; $n=2$ ab ab

$(ab)^0 = 1$



$L = \{\phi, ab, abab\ ab\}$

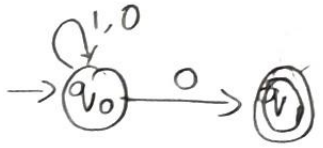
$G_{10}, G_{13}, G_{17}, G_{18}, H_2, H_3, H_8, H_9, J_2, J_4, J_6, J_7, J_9, K_1, K_2, K_3, K_6, K_8, K_9, L_0, L_1, L_4, L_5, L_6, L_7, M_0$

$M_3, M_4, M_6, N_1, N_4, N_6, N_7, N_8, N_9, P_1, P_2, P_3, P_5, P_6$

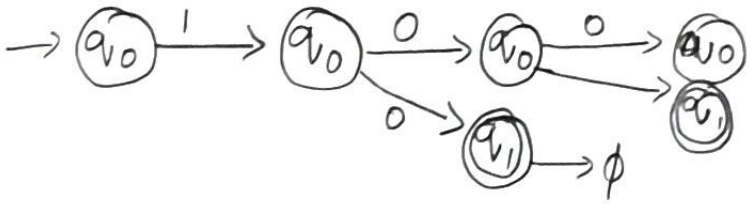
$L_{12}, L_{14}, L_{15}, L_{17}, L_{18}, L_{19}$

(11) Set of all strings that End with 0 ✓

Sol:-



$L = \{100$ it should be accepted



Transition table.

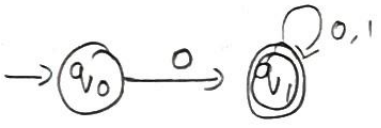
Eg: 01 Rejected

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	ϕ

If there is any way to run the machine that Ends in any set of states out of which @least one state is a final state, then the NFA accepts

(12) $L = \{$ set of all strings that starts with 0 $\}$

Sol:



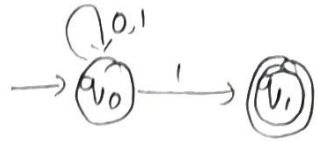
Examples: 001

(13) $L = \{$ Construct an NFA that accepts set of all strings over $\{0,1\}$ of length 2

Sol: $L = \{00, 01, 10, 11\}$

(14) L = { Set of all strings that Ends with '1' }

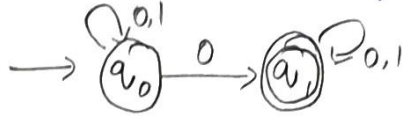
Sol:



0^*1

(15) L_2 = { Set of all strings that contain '0' }

Sol:

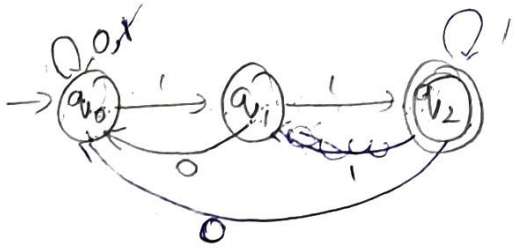


(16) L_3 = { set of all strings that starts with '10' }

(16) L_3 = { set of all string that Ends with '11' }

*

Sol:



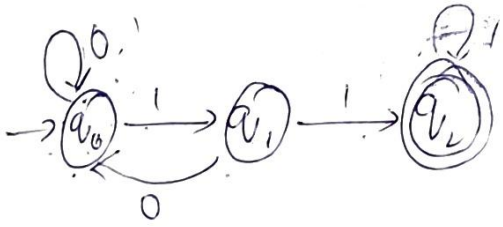
(17)

0 01110 1 1011

00101 1 (Accept)

01010 0 (Reject

6 Symbols



3 { 011

1 1 1 (Reject)

0 0 1 1 Accept

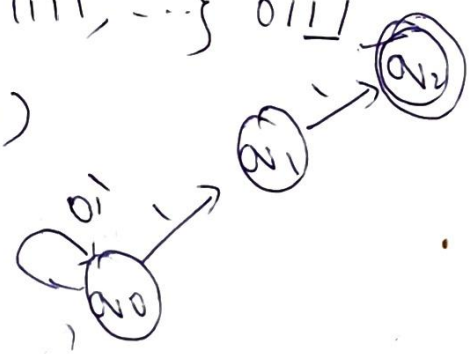
1 1 1 1

L : { 11, 011, 111, 0011, 1111, ... }

0111

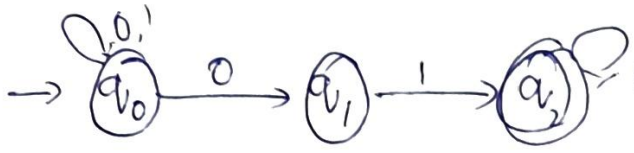
0101 (Reject)

010101

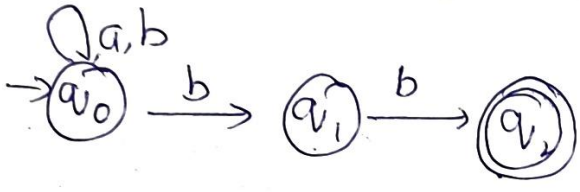


(18) construct NFA for $\omega = \{ x\,01\,y \mid \text{where } x, y = \{0,1\} \}$

sol:



NFA to DFA



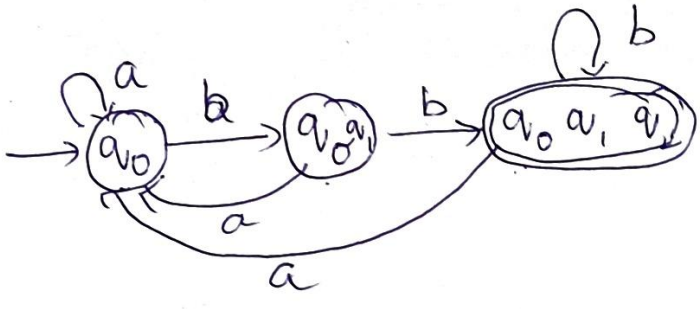
NFA Transition table.

	a	b
→ q_0	q_0	$\{q_0, q_1\}$
q_1	–	q_2
* q_2	–	–

step1: Transition table.

DFA

	a	b → new state
→ q_0	q_0	$\{q_0, q_1\}$
$\{q_0\ q_1\}$	q_0	$\{q_0\ q_1 q_2\}$
$\{q_0\ q_1\ q_2\}$	$\{q_0\}$	$\{q_0, q_2\}$



$G_0, G_1, G_2, G_3, G_4, G_5, G_7, G_8, H_0, H_1, H_2, H_6, H_7, H_8, H_9, J_0, J_2, J_3, J_4, J_6$
$J_8, J_9, K_1, K_2, K_4, K_6, K_7, L_0, L_1, L_3, L_5, L_7, L_8, M_0, M_1, M_2, M_3, M_4, M_5, M_6$
$M_8, M_9, N_0, N_1, N_4, N_6, N_7, N_8, N_9, P_1, P_2, P_5, P_6, P_7, P_8,$
$L_e, 13, 14, 15, 16, 17, 18, 19,$

Application of finite Automata

(1) s/w for designing and checking the behaviour of digital circuit

(2) Lexical analyzer of a typical compiler.

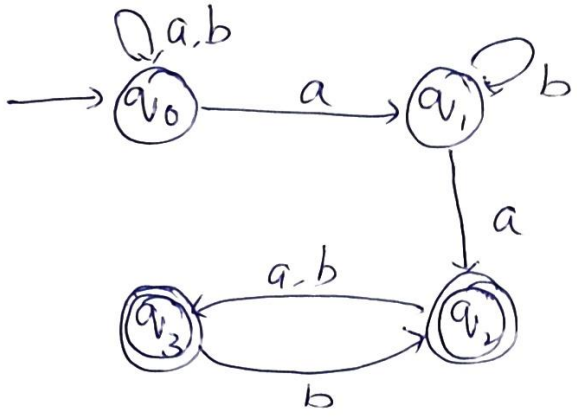
(3) s/w for scanning large bodies of text Example web pages, pattern matching

(4) s/w for verifying system of all types that have finite no. of states

Eg: stock market.

(5) Computer graphics, modeling of grammar, Game theory Theoritical biology (L-systms as a model of organism growth)

(Q)

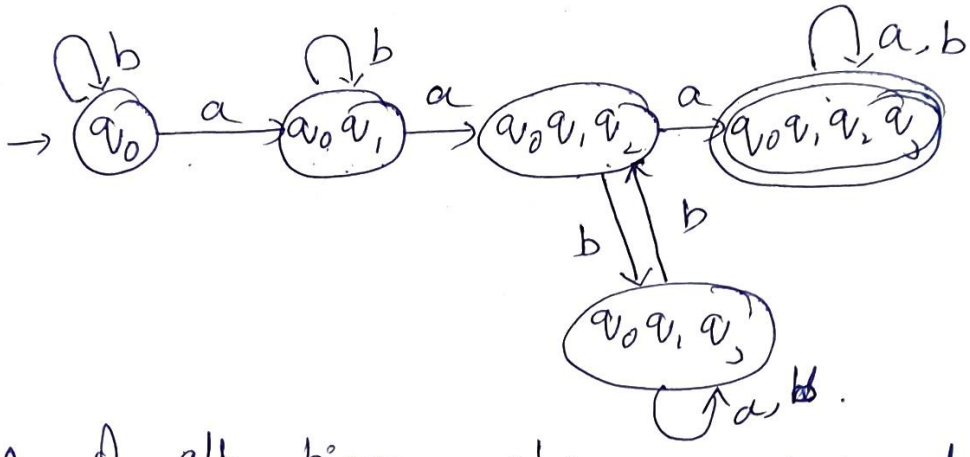


Sol:

δ	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3	-	q_2

Now

δ	a	b
q_0	$\{q_0, q_1\}$	q_0
$\{q_0 q_1\}$	$\{q_0 q_1 q_2\}$	$\{q_0 q_1\}$
$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_3\}$
$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_3 q_2\}$
$\{q_0 q_1 q_3\}$	$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$



(Q) NFA of all binary strings in which 2nd Last bit is '1'

0010 01010 10101

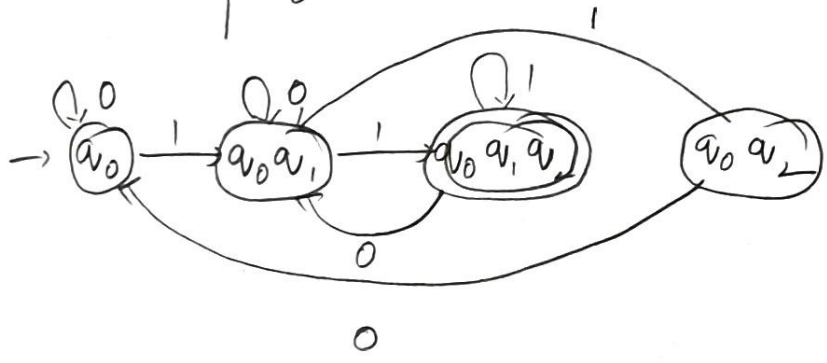
Sol:

δ	0	1
q_0	$\{q_0\}$	$\{q_0 q_1\}$
q_1	q_2	q_2
q_2	-	-

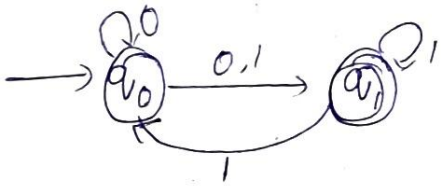
δ	0	1
q_0	$\{q_0\}$	$\{q_0 q_1\}$
$\{q_0 q_1\}$	$\{q_0 q_2\}$	$\{q_0 q_1 q_2\}$
$\{q_0 q_2\}$	q_0	$\{q_0 q_1 q_2\}$

δ	0	1
→ q_0	q_0	$q_0 q_1$
$q_0 q_1$	$q_0 q_2$	$q_0 q_1 q_2$
* $q_0 q_1 q_2$	$q_0 q_2$	$q_0 q_1 q_2$
$q_0 q_2$	q_0	$q_0 q_1$

NFA → NO dead state



(4)



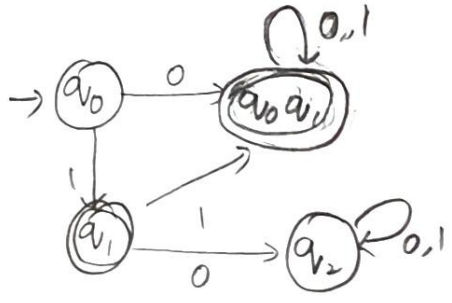
NFA to DFA

Sol

δ	0	1
q_0	$\{q_0, q_1\}$	q_1
q_1	$\{q_0, q_1\}$	$q_1 q_0$

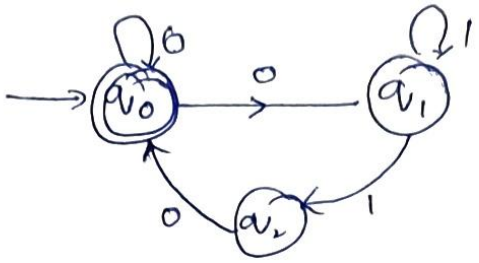
Step 2:

δ	0	1
q_0	$\{q_0 q_1\}$	q_1
$\{q_0 q_1\}$	$\{q_0 q_1\}$	$\{q_1 q_0\}$
q_1	-	$\{q_0 q_1\}$



final state is q_1
so Every state which has q_1 becomes final state

(5)

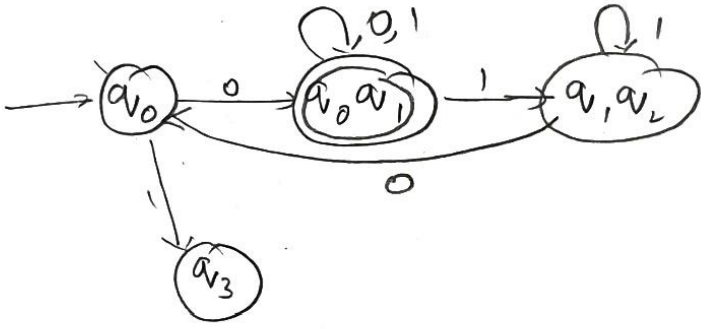


Sol:

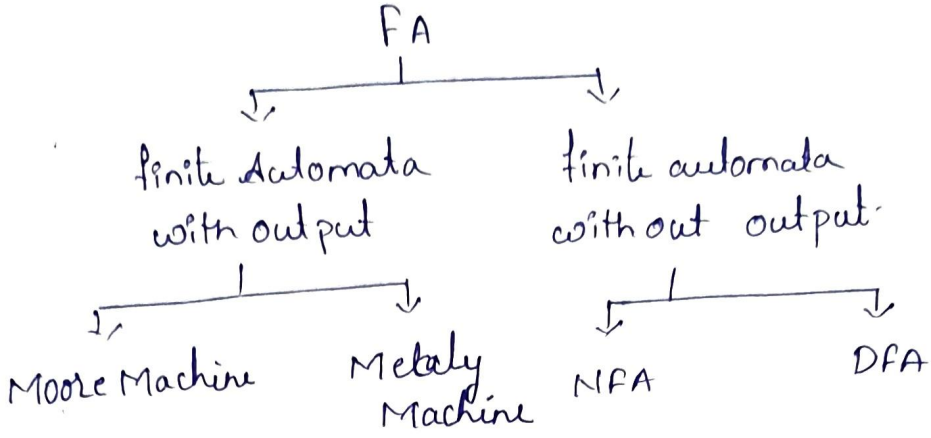
δ	0	1
$\rightarrow q_0$	$\{q_0 q_1\}$	ϕ
q_1	ϕ	$\{q_1, q_2\}$
q_2	q_0	ϕ

Step 2:

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



Finite Automata with output



Moore Machine : 6 - Tuple notation

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$Q \rightarrow$ No. of states

$\Sigma \rightarrow$ No. of i/p symbols

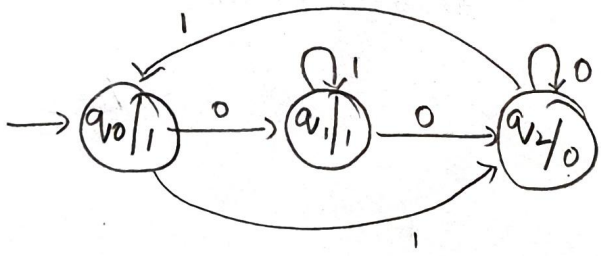
$\Delta \rightarrow$ No. of o/p symbols

$\delta \rightarrow$ Transition $Q \times \Sigma \rightarrow Q$

$\lambda \rightarrow$ O/p function $Q \rightarrow \Delta$

$q_0 \rightarrow$ initial state

* output always depends on current state
* Moore & Mealy Machine doesn't have any o/p



Transition table

current State	next State 0	next State 1	output (λ)
→ q_0	q_0	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

i/p string : 0110

* In Moore Machine if size of i/p string is 'n' size of o/p string is n+1

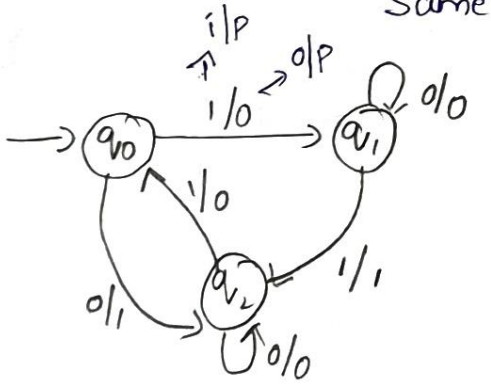
$\delta(q_0, 0110) = 1$ ✓
$\delta(q_0, 0110) = 1$ ✓
$\delta(q_1, 110) = 1$ ✓
$\delta(q_1, 10) = 1$ ✓
$\delta(q_1, 0) = 1$ ✓

} 5 input symbols.

q_2

Mealy Machine : 6 tuple notation

Same as Moore Machine.



* output function depends on both Q & Σ

$Q \times \Sigma \rightarrow \Delta$

current state	0 ns	0/p	1 ns	0/p
q_0	q_2	1	q_1	0
q_1	q_1	0	q_2	1
q_2	q_2	0	q_0	0

i/p string 1001

* In Mealy Machine i/p string is n o/p string will also be n

1001

$\delta(q_0, 1001) = 0$ o/p string : 0001

$\delta(q_1, 001) = 0$

$\delta(q_1, 01) \Rightarrow 0$

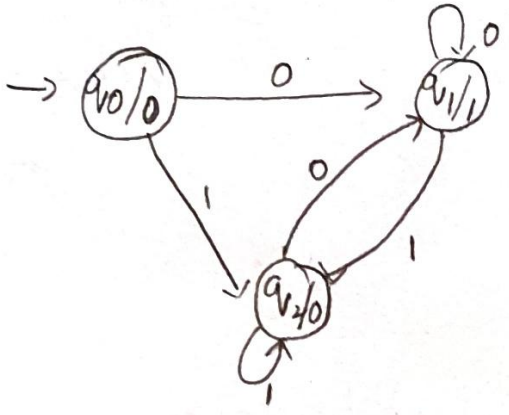
$\delta(q_1, 1) \Rightarrow 1$

$\delta(q_0, \varepsilon) \Rightarrow q_2$

(Q) Design a Mode Machine to find the 1's complement of a given binary number.

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

Method 1 (3 states)



1011 = 1's ⇒ 0100

1 0 1 1

q_0 q_2 q_1 q_2 q_2

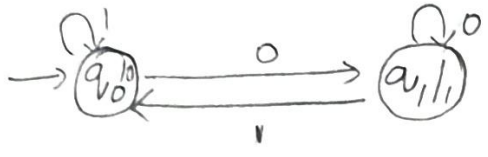
0 0 1 0 0

0 — ignore

0100 — 1's complement of given string

∴ Input string is 4 symbols & o/p string has 5 symbols and also 1's complement of given string

Method 2 (2 states).



i/p string 1011

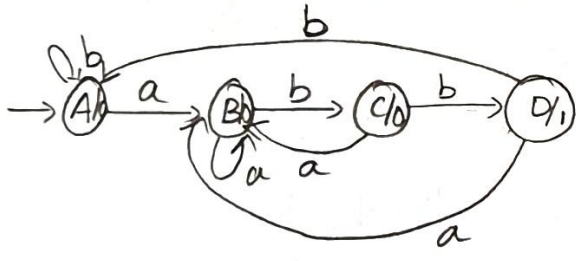
1 0 1 1
q_0 q_0 q_1 q_0 q_0
0 0 1 0 0
↓
ignore.

difference b/w Moore & Mealy in the way o/p is generated.
Application Electrical circuits.

* Construct a Moore Machine that counts the occurences of the sequence 'abb' in any i/p strings over $\Sigma(a,b)$

Sol: $\Sigma = \{a,b\}$ $\Delta = \{0,1\}$

abb ⇒ 1



Example

	a	b	b
	A	B	C	D
	0	0	0	1

Single occurance of abb

	a	b	a	b
	A	B	C	D	B	C	D
	0	0	0	1	0	0	1

2 no. of 1's

2 occurances of abb

(Q) for the following Moore Machine the i/p alphabet is $\Sigma = \{a, b\}$ and o/p alphabet is $\Delta = \{0, 1\}$. Run the following i/p sequences and find the respective o/ps

(i) aabab (ii) abbb (iii) ababb

Sol:

States	a	b	o/p
→ q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

① a a b a b

q_0 q_1 q_2 q_4 q_0 q_2

0 0 1 0 0 1

② a b b b

q_0 q_1 q_3 q_4 q_0

0 0 0 0 0

③ a b a b b

q_0 q_1 q_3 q_4 q_0 q_2

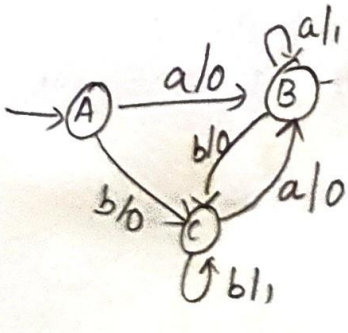
0 0 0 0 0 1

Mealy Machine Example:

(Q) Design a Mealy Machine accepting the language consisting of strings from Σ^*, where $\Sigma = \{a, b\}$ and the string should End with either aa or bb.

Sol:

aa → 1 bb → 1



K_1 → Present

i/p string ba aa aaa

00 01 011

↓

we should understand that is correct

(Q) Construct a Mealy Machine that gives 2's complement of any binary i/p (Assume that the last carry bit is neglected)

Sol:

2's complement → 1's complement + 1

Eg: MSB ← LSB

10100

01011 → 1's complement

complemented

+ 1

01100 → 2's complement

1111

0000

complemented

+ 1

0001 → 2's complement

11100

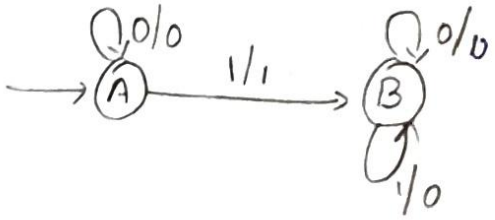
00011

complemented

+ 1

00100 → 2's complement

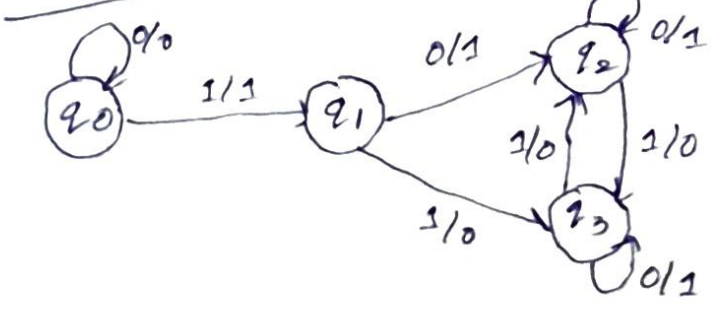
* In 2's complement first 1 which you is left as same. and from that one to msb it is complemented



10100

BBBAA

00000



10100

BBBAA

00000

10100

BBBAA

00

Construction of Mealy Machine

* Construct a Mealy Machine that gives 2's complement of any binary input (Assume that the last carry bit is neglected)

Sol: 2's complement = 1's complement + 1

10100 → 1's 01011 + 1 = 01100 → 2's

11100 → 1's 00011 + 1 = 00100

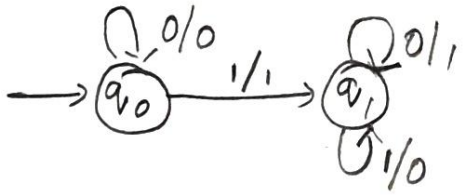
1111 → 0000 + 1 = 0001 → 2's complement

0 is left as 0, 0 is left as so

when ever we find first one it will be same.

(from LSB to MSB) Least segnificant bit

there after bits get complimented.



10100

$q_1 q_1 q_1 q_0 q_0$

1 0 0

10100

$q_1 q_1 q_1 q_0 q_0$

0 1 1 0 0 ✓

11100

$q_1 q_1 q_1 q_0 q_0$

0 0 1 0 0 ✓

11111

$q_1 q_1 q_1 q_1 q_0$

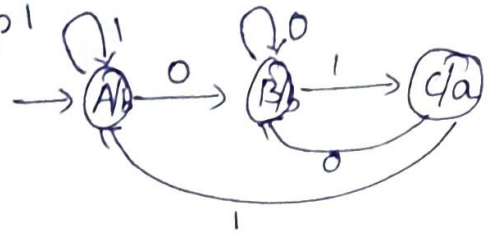
0 0 0 1 0

* on

Conversion of Moore Machine to Mealy Machine

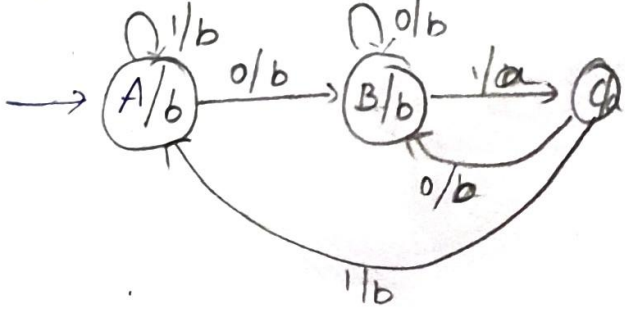
Construct a Moore Machine that prints 'a' whenever the sequence '01' is Encountered in any input binary string and then Convert to its Equivalent Mealy Machine

Sol: Step 1

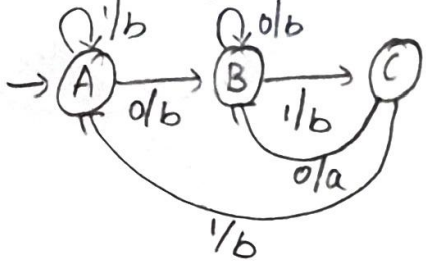


State	0	1	λ
→ A	B	A	b
B	B	C	b
C	B	A	a

Step 2



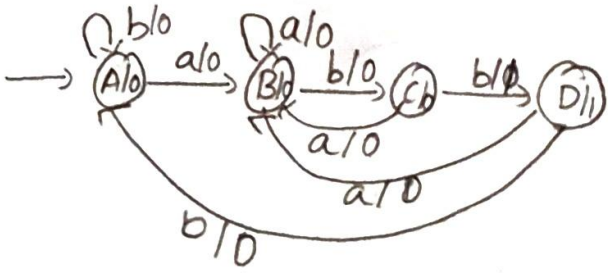
Step 3: Remove the o/p's for states.



Step 4: Convert the transition table.

State	0	1	λ
A	B,b	A,b	b
B	B,b	C,a	b
C	B,b	A,b	a

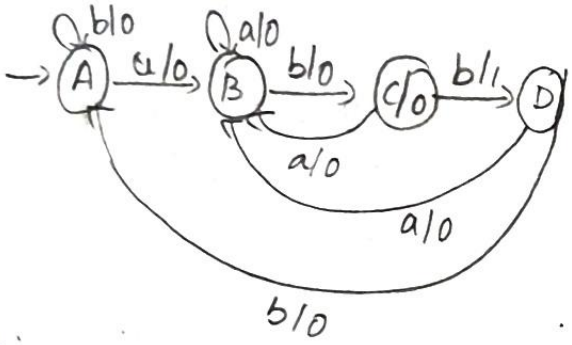
→ The given Moore Machine counts the occurances of the sequence 'abb' in any input binary strings over {a,b} convert it to its Equivalent Mealy Machine



Sol: Step 1: Transition table

current state	a	b	λ
A	B	A	0
B	B	C	0
C	B	D	0
D	B	A	1

Step 2: Remove o/p in states

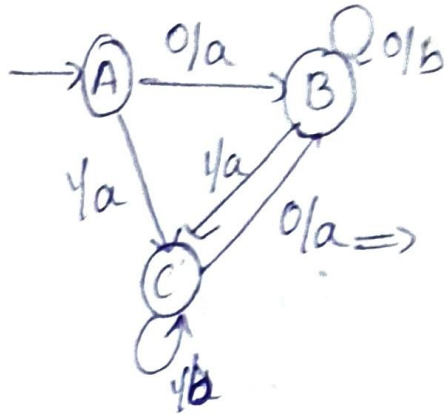


Step 3: Convert the transition table.

current state	a	b	λ
A	B,0	A,0	0
B	B,0	C,0	0
C	B,0	D,1	0
D	B,0	A,0	1

Conversion of Mealy Machine to Moore machine

(Q)

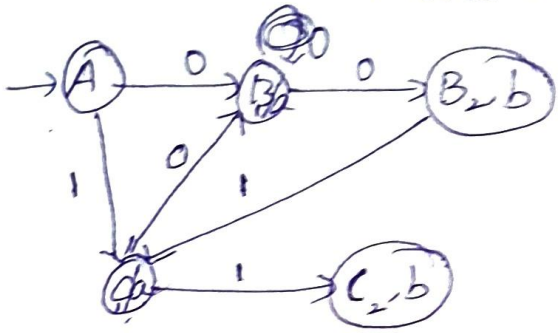


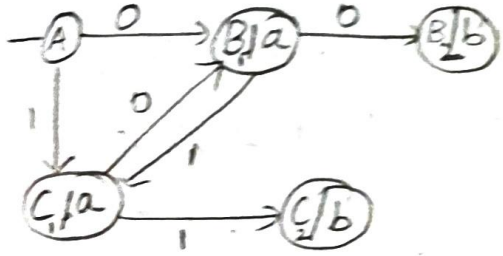
$\Sigma = \{0, 1\}$ $\Delta = \{a, b\}$

=> Mealy Machine

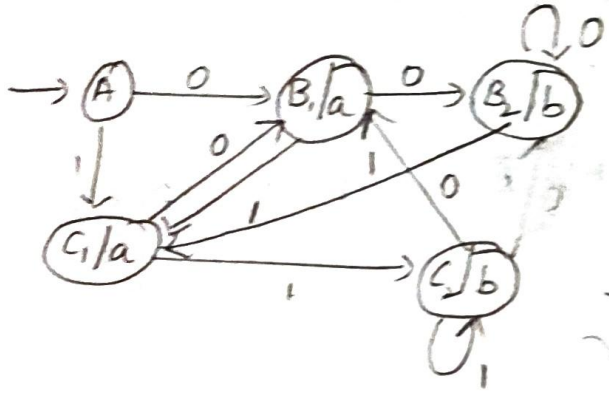
Sol:

Moore Machine

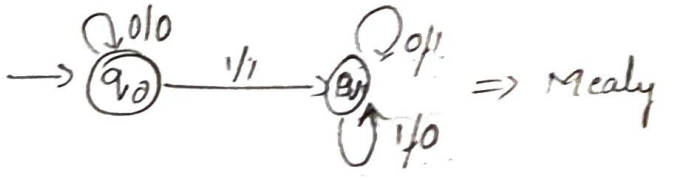




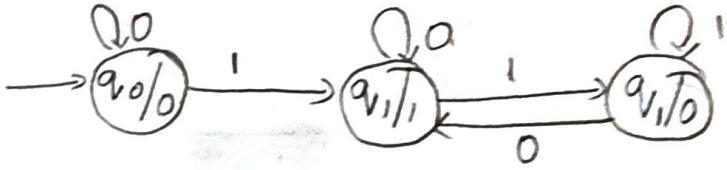
Step 2:



(Q) 2's complement



=> Mealy machine



1 0 1 0 0

$q_1\ q_1\ q_0\ q_1\ q_0$

0 0 1 0 0

Finite automata

- with output
  - Mealy Machine
  - Moore Machine
- Finite automata without output
  - DFA
  - NFA
  - ϵ-NFA.

ϵ-NFA (or) NFA with ϵ transition

↳ Empty symbols

Regular NFA has 5-tuple Notation

$(Q, \Sigma, \delta, q_0, q_f)$

$\delta = Q \times \Sigma = 2^Q$

ϵ-NFA $\delta = Q \times \Sigma \cup \epsilon \rightarrow 2^Q$ $(Q, \Sigma, \delta, q_0, q_f)$

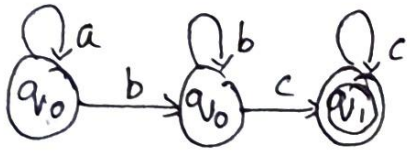
↓ alphabet with ϵ

Example

* Every state on ϵ goes to it self

Example: Construct NFA with ϵ which accept all strings over $\Sigma = \{a, b, c\}$ where string contains multiple a's followed by multiple b's followed by c's

Sol:



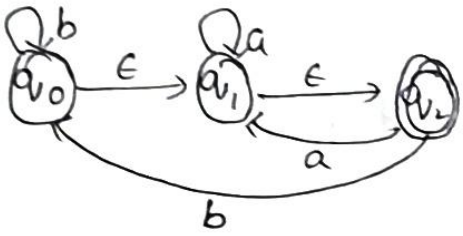
$\Sigma = \{a, b, c, \epsilon\}$

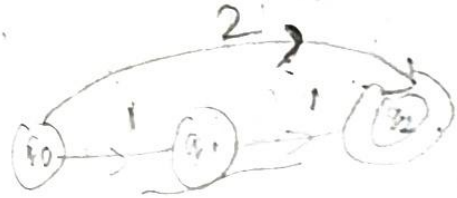
(or)

	a	b	c	ϵ
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

Conversion of NFA with ϵ moves to without ϵ moves

(Q)





Sol: Step 1: ϵ-closure$(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure$(q_1) = \{q_1, q_2\}$

ϵ-closure$(q_2) = \{q_2\}$

Step 2:

$\delta'(q_0, a) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_0), a))$

$= \epsilon$-closure$(\delta(\{q_0, q_1, q_2\}), a)$

$= \epsilon$-closure$(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a))$

$= \epsilon$-closure$(\phi \cup q_1 \cup q_1)$

$= \epsilon$-closure(q_1)

$= \{q_1, q_2\}$

$\therefore \quad \delta'(q_0, a) = \{q_1, q_2\}$

$\delta'(q_0, b) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_0), b))$

$= \epsilon$-closure$(\delta(\{q_0, q_1, q_2\}, b))$

$= \epsilon$-closure$(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b))$

$= \epsilon$-closure$(q_0 \cup \phi \cup q_0)$

$= \epsilon\text{-closure}(q_0 \cup \phi \cup q_0)$

$= \epsilon\text{-closure}(q_0)$

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

$\delta'(q_1, a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), a))$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}), a)$

$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a))$

$= \epsilon\text{-closure}(q_1 \cup q_1)$

$= \epsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\therefore \delta'(q_1, a) = \{q_1, q_2\}$

$\delta'(q_1, b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), b))$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2\}), b)$

$= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b))$

$= \epsilon\text{-closure}(\phi_0 \cup q_0)$

$= \epsilon\text{-closure}(q_0)$

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

$\therefore \delta'(q_1, b) = \{q_0, q_1, q_2\}$

*** $\delta'(q_2, a) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2), a))$

$= \epsilon\text{-closure}(\delta(q_2), a)$

$= \epsilon\text{-closure}(\delta(q_2, a))$

$= \epsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\therefore \delta'(q_2, a) = \{q_1, q_2\}$

$$\delta'(q_2, b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2), b))$$

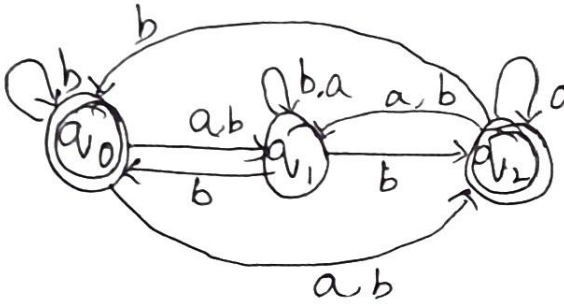
$$= \epsilon\text{-closure}(\delta(q_2), b)$$

$$= \epsilon\text{-closure}(\delta(q_2, b))$$

$$= \epsilon\text{-closure}(q_0)$$

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

$$\therefore \delta'(q_2, b) = \{q_0, q_1, q_2\}$$

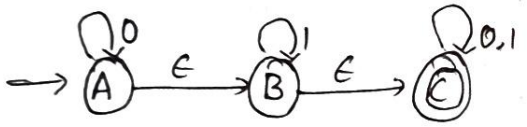


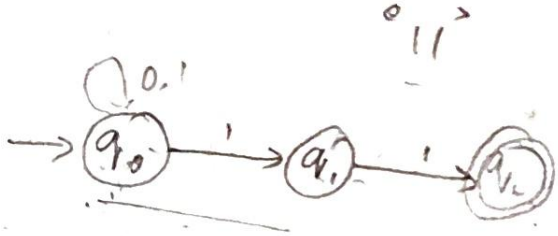
NFA without Epsilon

Now, $F \cup \{q_0\}$ if ϵ-closure of q_0 has F

$q_2 \cup q_0$

(2)





NFA to DFA

"11"

01 00 11

011