1/8/2017

MODULE - 1

Terminologies

1. Alphabet - finite non empty set of symbols is known as alphabet denoted by symbol Σ.

eg: English alphabets $\Sigma = \{A, B, \ldots Z, a, b, \ldots z\}$

Malayalm alphabets $\Sigma = \{$ ക, ഖ, ... ഹ $\}$

Decimal alphabets $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

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

2. Strings - a string is a finite sequence of symbols chosen from some alphabet.

eg: 0010 is a string from binary alphabet

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

: 123 is a string from decimal alphabet

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

3. Length of a string - length of a string is the no: of positions for symbols in the string. Denoted by $|w|$.

eg: $w = 1001$

$|w| = 4$

4. Empty string (ε Epsilon)

The empty string is the string with zero occurances of symbols. The empty string can be denoted by ε / λ.
length of (Epsilon) ε is always zero

$$|\varepsilon| = 0$$

5. Powers of an alphabet

If Σ is an alphabet we can express the set of all strings of a certain length from that alphabet using exponential notation we define Σ^k to be the set of strings of length k.
eg: if $\Sigma = \{0,1\}$
then $\Sigma^1 = \{0,1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$
$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
$\Sigma^0 = \{\varepsilon\}$

2/9/17.

→ Kleen's Closure / Star ~~Colse~~ Closure (Σ^*)

$\Sigma = \{0,1\}$ $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

Let Σ be the alphabet then Kleen's closure is denoted by Σ^* which represent set of all string (including empty string '$\{\varepsilon\}$') over the alphabet 'Σ'

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$$
$$= \{\varepsilon\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \ldots$$
$$= \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$$

→ positive closure: Σ^+

Let Σ be an alphabet the positive closure represented by Σ^+ which represents set of all string (Excluding ε). which mean

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}$$

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$$
$$= \{0,1\} \cup \{00, 01, 10, 11\} \cup \ldots$$
$$= \{0, 1, 00, 01, 10, 11 \ldots\}$$

→ Concatenation of strings:-

~~x = 0101~~ The concatenation of strings x and y, is denoted by x.y or yx

$x \cdot y$ means all the symbols x followed by all the symbols of y. Suppose

$x = 01011$

$y = 10101$

$x \cdot y = 0101110101$

$y \cdot x = 1010101011$

→ Reverse of a string (w^R)

The reverse of a string is obtained by writing the symbols in reverse order

eg: $x = 01011$

$x^R = 11010$

LANGUAGE (L)

Let Σ be an alphabet and Σ^* be the set of all string on Σ. The subset of Σ^* is termed as language which is denoted by 'L'

$L \subseteq \Sigma^*$

Σ^* is the language for any alphabet 'Σ'

ϕ the empty language is the language over any alphabet

$\{\varepsilon\}$ the language consisting of only empty string and it is also a language over any alphabet $\phi \neq \{\varepsilon\}$

eg: Suppose $\Sigma = \{0, 1, 2, \ldots 9\}$

The language of odd number $L_1 = \{1, 3, 5, 7, 9\}$

~~The language of 3 consecutive a~~ $L_2 = \{a, b\}$

3/8/17

Type 3 - Formalization - Finite State Automata

A finite state Automata is an abstract model of a digital computer. There are mainly three types of finite automata

FSA

DFSA | NDFSA | E-NFA

Deterministic finite state (M)

formal definition deterministic finite state consist of 5 ~~arguments~~ tuple;

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

where,

Q = Set of states

Σ = alphabet

δ = transition function. $\delta: Q \times \Sigma \rightarrow Q$

q_0 = Initial state

F / A = Final state (more than 1 state) Set of final state or accepting state.

4/8/17

Transition function.

Here the transition funct^n δ receive two arguments current state ~~(q) and~~ $q \in Q$ and input alphabet $a \in \Sigma$ and returns ~~P~~ next state $P \in Q$

$$\delta(q, a) = P$$

(q: current state; a: alphabet; P: next state.)

Representation of finite automata

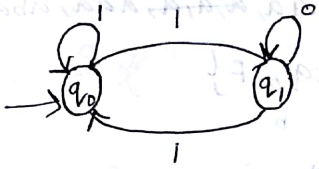
1) Transition Diagram:

Transition diagram is a directed graph having vertices and edges

- Single vertexed circle - State. ○
- Double ~~te~~ circle - final state ◎

→ : Initial arc - transit^n

eg: finite automata that accepts even no: of ones over the alphabet set $\{0, 1\}$. The above mentioned DFA $M = \{Q, \Sigma, \delta, q, F\}$



~~M =~~ $Q = \{q_0, q_1\}$

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

$\delta = \{(q_0, 0) = q_0,\ (q_0, 1) = q_1,\ (q_1, 0) = q_1,\ (q_1, 1) = q_0\}$

$q_0 = q_0$

$F = \{q_1\}$

Transition Table. (no: of alphabets)

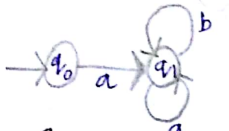
no: of States		0	1
	$\rightarrow q_0$	q_0	q_1
	$* q_1$	q_1	q_0

Q Design DFA that accept Strings Starting with A over the alphabet Set $\{a, b\}$

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

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

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



$Q = \{q_0, q_1\}$

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

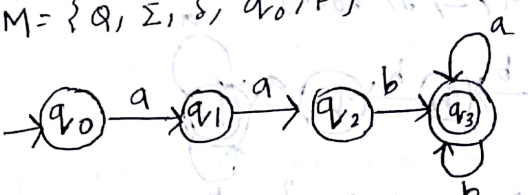
$\delta = \begin{cases} (q_0, a) = q_1 \\ (q_0, b) = - \\ (q_1, a) = q_1 \\ (q_1, b) = q_1 \end{cases}$

	a	b
q_0	q_1	-
q_1	q_1	q_1

Design DMA that accept String starting with aab

$L_1 = \{aab, aaba, aabb, aabab$

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



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

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

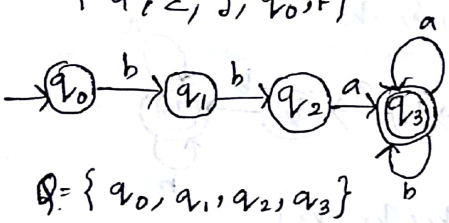
$\delta = \begin{cases} (q_0, a) = q_1 \\ (q_1, a) = q_2 \\ (q_2, b) = q_3 \\ (q_3, a) = q_3 \\ (q_3, b) = q_3 \end{cases}$

	a	b
q_0	q_1	-
q_1	q_2	-
q_2	q_3	-
q_3	q_3	q_3

q bba

$L_1 = \{ bba, bbab, bbaa, bbaba \ldots \}$

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



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

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

$\delta = \{ (q_0, b) = q_1$

$(q_1, b) = q_2$

$(q_2, a) = q_3$

$(q_3, a) = q_3$

$(q_3, b) = q_3 \}$

Transition table

	a	b
q_0	–	q_1
q_1	–	q_2
q_2	q_3	–
q_3	q_3	q_3

q Design FA that end with ab over the alphabet set

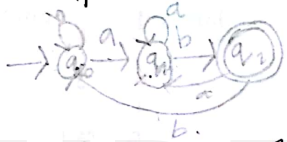
A) $\Sigma = \{a, b\}$

~~L =~~ { Strings correspond to above language.

$L = \{ ab, aab, bab, aaab, abab, baab, bbab \ldots \}$

Length

(min no: of states = ~~no: of alphabet~~ of Smallest String + 1)



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

~~Σ~~ $\delta = \{ (q_0, a) = q_1$

$(q_0, b) = q_0$

$(q_1, a) = q_1$

$(q_1, b) = q_2$

$(q_2, a) = q_1$

$(q_2, b) = q_0 \}$

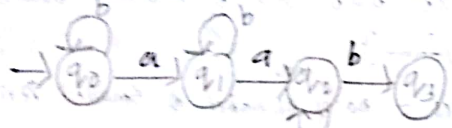
Transition table.

	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

Q Ends with aaab.

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

$L = \{$ aaab, aaaab, baab, abaaab, baaaab, aaaaaab, aabaab, abaaab, baaaab, bbbaaab $\}$



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

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

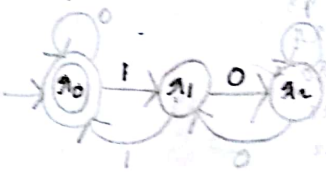
$\delta = \{ (q_0, a) = q_1$
$(q_0, b) = q_0$
$(q_1, a) = q_2$
$(q_1, b) = q_1$

Q Design FA that accept binary strings divisible by three.

N = No states

3 = 3 No states

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

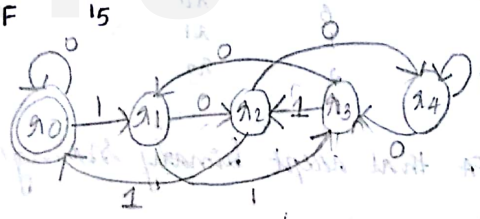


Binary	Decimal	Reminder	State
0	0	0	q_0
1	1	1	q_1
10	2	2	q_2
11	3	0	q_0
100	4	1	q_1
101	5	2	q_2
110	6	0	q_0
111	7	1	q_1
1000	8	2	q_2

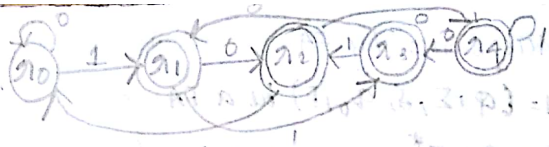
Q) Design FA that accept binary strings divisible by Five.

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

Binary	Decimal	Reminder	State
0	0	0	q_0 ✓
1	1	1	q_1
10	2	2	q_2
11	3	3	q_3
0100	4	4	q_4
0101	5	0	q_0 ✓
0110	6	1	q_1
0111	7	2	q_2
1000	8	3	q_3
1001	9	4	q_4
1010	10	0	q_0 ✓
1011	B	11	
1100	C	12	
1101	D	13	
1110	E	14	
1111	F	15	



Develop automata that accept binary strings that are not divisible by 5.



9/8/17

Extended Transition funtion.

Let $M=(Q,\Sigma,\delta,q_0,F)$ being an FA.

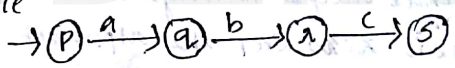
We define Extended transition fn $\delta: Q\times\Sigma^* \to Q$ as follows

(1) for any $q\in Q$, $\delta^*(q,\varepsilon)=q$

(2) for any $q\in Q$, $a\in\Sigma$, $y\in\Sigma^*$

$$\delta^*(q,ya)=\delta(\delta^*(q,y),a)$$

Example



$$\delta^*(p,abc) = \delta((\delta^*(p,ab)),c)$$

($y=ab$, $a=c$)

$$= \delta((\delta(\delta^*(p,a),b)),c$$

$$= \delta(\delta(\delta(\delta^*(p,\varepsilon,a)\,b),c)$$

$$= \delta(\delta(\delta(p,a),b),c)$$

$$= \delta(\delta(q,b),c)$$

$$= \delta(r,c)$$

$$= s$$

Acceptance by FA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a FA

A string $x \in \Sigma^*$ is accepted by M.

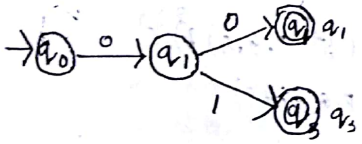
iff $\delta^*(q_0, x) \in F$

If a string is not accepted we say it is rejected by M. Language accepted recognized by FA. ~~The lang recognized~~ is knowns as regular languages. Regular Lang is a set of all strings accepted by FA. ie $L(M) = \{x$

$L(M) = \{x \mid x \in \Sigma^* \text{ and } \delta^*(q, x) \in F\}$

Q Design FA that accepts strings where second last elements is zero over the alphabet set $\Sigma = \{0, 1\}$

A) $L = \{00, 01, 001, 101, 100, 000 \ldots$



Non-Deterministic Finite State Automata (NFA)

Formally a NFA can be defined as a five tuple $N = \{Q, \Sigma, \delta, q_0, F\}$ where,

Q = Finite set of states

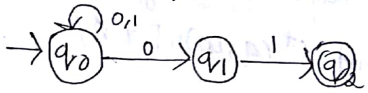
Σ = Finite set of Symbols as Alphabet

q_0 = initial state.

F = set of final states.

$\delta = Q \times \Sigma \to 2^Q$ is a transition funct^n that takes a state in Q. And an i/p alphabet Symbol in Σ as argument and returns a subset of Q

eg: NFA accept strings ends with (0,1) over the alphabet set $\Sigma = \{0, 1\}$ is shown in below fig



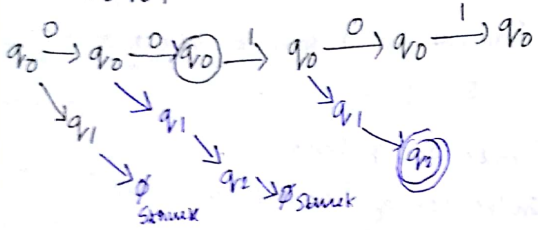
$N = \{Q, \Sigma, \delta, q_0, F\}$

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

$\delta = \begin{cases} (q_0, 0) \to \{q_0, q_1\} \\ (q_0, 1) \to \{q_0\} \\ (q_1, 1) \to \{q_2\} \\ (q_2, 0) \to \\ (q_2, 1) \to \end{cases}$

	0	1
$\to\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	ϕ	q_2
$\{q_2\}$	ϕ	ϕ

I/P = 00101



The above fig shows transition Sequence of NFA by processing I/p signal ~~Sym~~ 00101

The languages accepted by NFA are known as Regular lang. it a set of all string accepted by NFA over the alphabet set Σ.

Formally the language accepted by NFA $N = \{Q, \Sigma, \delta, q_0, F\}$ is defined along

$$L(N) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \cap F \neq \emptyset\}$$

Equivalace of NFA and DFA

Equivalance of two finite state automata. Two finite state automata M, N are said to be equivalant if and only if lang. accepted by M = lang. accepted by N.

$$L(M) = L(N)$$

Every language that can be described by some NFA can als be described by DFA.

proof that DFA's can do whatever NFA's can do whatever involves an important construction called the SUBSET CONSTRUCTION. Because, it involves constructing all subset of the set of states of the NFA. The SUBSET CONSTRUCTION starts from an NFA $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ its goal is the description of DFA $M = \{Q_M, \Sigma, \delta_M, \{q_0\}, F_M\}$ such that $L(N) = L(M)$

1: Input alphabet of N = M = Σ.

2: Initial state of M is the set contain only the ~~state~~ Start State $q_0 = \{q_0\}$

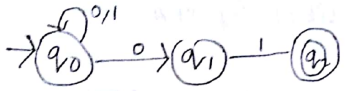
3: Q_M is the set of subset of Q_N ie Q_M = powerset of Q_N.

4: F_M is the ~~set~~ set of N state that include atleast one accepting state in N

5: For each subset $S \subseteq Q_N$ and $a \in \Sigma$

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

Q Convert the below NFA to DFA



Suppose $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ represents above NFA that contains

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

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

$q = \{q_0\}$

$F_N = q_2$

NFA - transition table

NFA	0	1
q_0	q_1, q_0	q_0
q_1	ϕ	q_2
q_2	ϕ	ϕ

To convert above NFA $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ into $M = \{Q_M, \Sigma, \delta_N, q_M, F_M\}$

we use subset construction

Subset construction with

Step 1: Input alphabet $\Sigma = \{0,1\}$

2: Initial state of DFA ~~$M = \{$~~ $q_0 = \{q_0\}$.

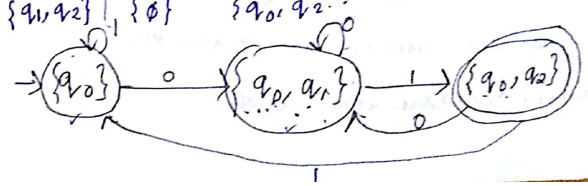
3: $Q_M = 2^{Q_N}$ (power set of Q_N)

4: $F_M = \{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}$

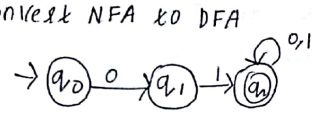
5: δ_M can be defined as

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

unreachable states



Q Convert NFA to DFA



A)

Suppose $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$

Represents above NFA that contains.

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

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

$q_0 = \{q_0\}$

$F_N = \{q_2\}$

NFA transition

NFA	0	1
q_0	q_1	$\emptyset$
q_1	$\emptyset$	q_2
q_2	q_2	q_2

To convert above NFA $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ into $M = \{Q_M, \Sigma, \delta_M, F_M\}$

we use subset construction,

Subset construction with.

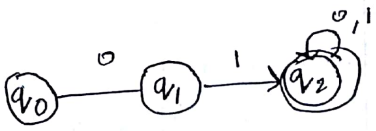
Step1: Input alphabet $\Sigma = \{0, 1\}$

2: Initial state of DFA $q_0 = \{q_0\}$

3: $Q_M = 2^{Q_N}$

4: $F_M = \{\{q_2\}, \{q_0, q_2\}\ \{q_0, q_1, q_2\}\}$

DFA	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_0, q_1\}$	$\{q_1\}$	$\{q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	q_2	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$



Epsilon NFA: (ε-NFA)

ε-NFA is formally defined as ~~a~~ five tuple $N = \{Q, \Sigma, \delta, q_0, F\}$. Where

Q = Set of states

Σ = Set of alphabets.

δ = Transition function. $\delta: Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$.

q_0 = Initial state.

F = final state.

eg: Below fig shows a sample ε-NFA.

ε-NFA Transition diagram.

	ε	0	1
→A	{B}	{A}	∅
B	{D}	{C}	∅
C	∅	∅	{B}
*D	∅	{D}	∅

Closure of ε

closure on ε is the set of states that can be reached from a state ~~q~~ q upon applying ~~to~~ zero or more epsilon transition.

ε-closure(A) = {A, B, D}

ε-closure(B) = {B, D}

(C) = {C}

(D) = {D}

∅ = {∅}

Equivalance of ε-NFA & NFA

Step-2 $\delta(A, \Sigma^*, 0, \Sigma^*)$

{A, B, D}

{A, C, D}

{A B D} ∪ {C} ∪ {D} = ABCD.

$\delta(A, \varepsilon^* | \varepsilon^*)$

{A B D}

∅ ∪ ∅ ∪ ∅ = ∅

$\delta(B, \varepsilon^* 0 \varepsilon^*)$

{B D}

{C} {D} $\varepsilon^* = \{C, D\}$

$\delta(B, \varepsilon^* | \varepsilon^*)$

{B, D}

{∅}, {∅}

∅

$\{C, \Sigma^* 0 \Sigma^*\}$

ε-closure(C) = {C}

Apply 0 = $\{\phi\}$

ε^* = $\{\phi\}$

$\{C, \Sigma^* 1 \Sigma^*\}$

ε-closure(C) = {C}

Apply ϕ = {B}

ε^* = {B, D}.

$\{D, \Sigma^* 0 \Sigma^*\}$

ε-closure(D) = {D}

Apply 0 = {D}

Σ^* = {D}.

$\{D, \Sigma^* 1 \Sigma^*\}$.

ε-closure(D) = {D}

Apply 1 = ϕ

Σ^* = $\{\phi\}$

eg:

NFA	0	1
→ * A	{A, B, C, D}	ϕ
B	{C, D}	ϕ
C	ϕ	{B, D}
* D	{D}	{ϕ}

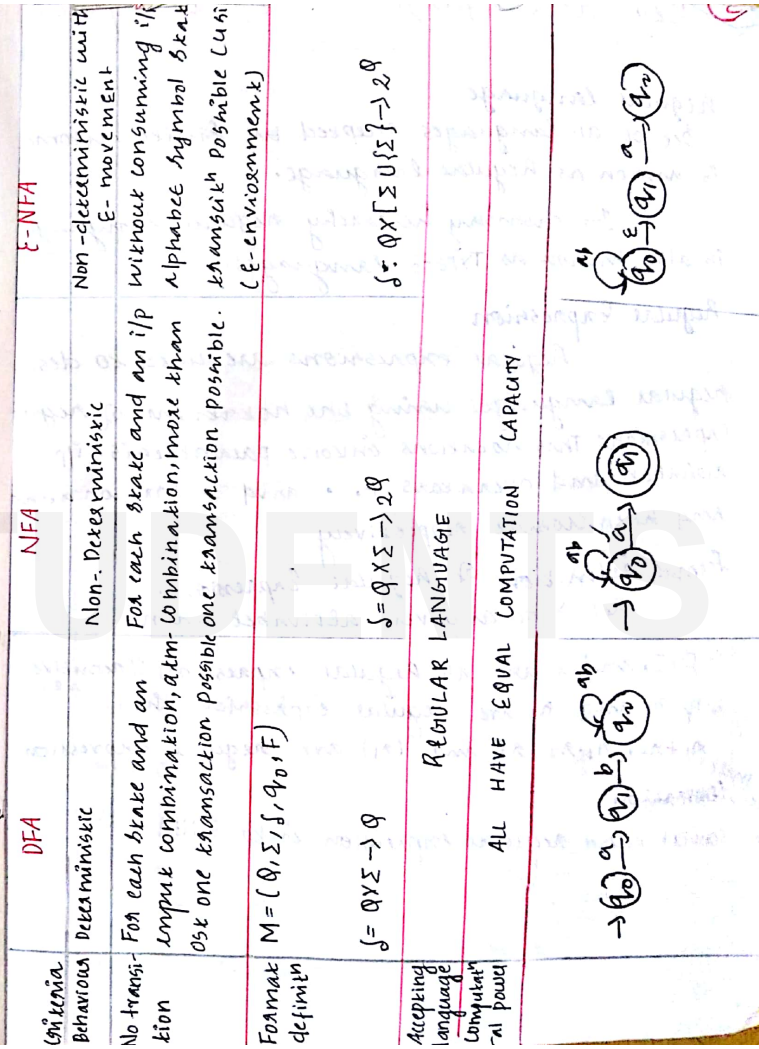
Step1: Check the final state and initial states are equal.

Final State = Initial State

D = A

{D} = {A, B, D}

Criteria	DFA	NFA	ε-NFA
Behaviour	Deterministic	Non-Deterministic	Non-deterministic with ε-movement
Transition	For each state and an input combination, atmost one transaction possible	For each state and an i/p combination, more than one transaction possible	without consuming i/p alphabet symbol state transition possible (with ε-environment)
Formal definition	$M = (Q, \Sigma, \delta, q_0, F)$		
	$\delta = Q \times \Sigma \to Q$	$\delta = Q \times \Sigma \to 2^Q$	$\delta : Q \times [\Sigma \cup \{\Sigma\}] \to 2^Q$
Accepting language	REGULAR LANGUAGE		
Computational power	ALL HAVE EQUAL COMPUTATION CAPACITY		



(2) Regular language and Regular expression

Regular language.

Set of all languages accepted by finite automata is known as Regular language.

In chomsky heirarchy regular languages is also known as TYPE-3 languages.

Regular Expression

Regular expressions are used to describe regular languages. using the notations of regular expressions. This notations involve paranthesis, i/p alphabet and operators +, · and * (or, concatenat^n and kleen closure respectively

Formal definition of Regular Expression.

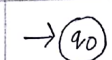
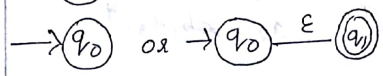
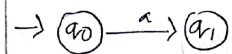
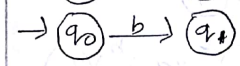
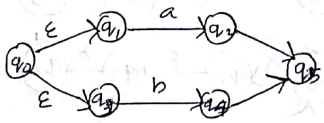
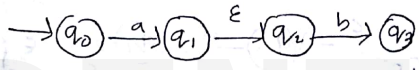
Let Σ be an given alphabet then

1. $\emptyset, \Sigma$, and a are all regular expressions. (Premitive $\lambda\varepsilon$)
2. If r_1 and r_2 are regular expression then $r_1 + r_2$, $r_1 \cdot r_2$, r_1^* and (r_1) are regular expression.

10 mark

Q Convert below regular expression into ε-NFA.

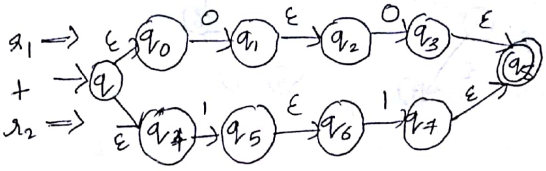
A)

Regular Language	ε-NFA
$r = \emptyset$	
$r = \Sigma$	
$r_1 = a$	
$r_2 = b$	
$r_1 + r_2 = a + b$	
$r_1 \cdot r_2 = ab$	
a^*	

Q Convert regular expression $\begin{matrix}00\\11\end{matrix}^+$ into NFA

$r_1 = 00$

$r_2 = 11$



8 Convert below regular Expression into NFA

$a + (ab)^*$

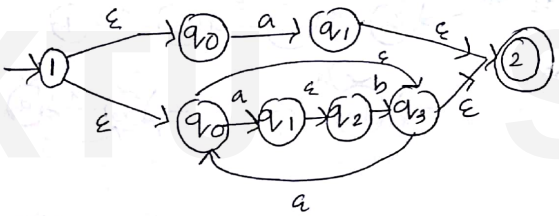
$r_1 = a$ $\quad r_3 = ab$

$r_2 = (ab)^*$ $\quad r_2 = (r_3)^*$

$r_1 = a$ → $q_0 \xrightarrow{a} q_1$

$r_3 = ab$ → $q_0 \xrightarrow{a} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_3$

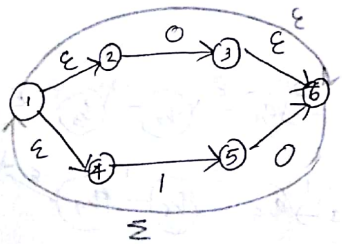
$r_2 = (r_3)^*$ → $q_0 \xrightarrow{a} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_3$, $q_3 \xrightarrow{\varepsilon} q_0$, $q_0 \xrightarrow{\varepsilon}$ final, $q_3 \xrightarrow{\varepsilon}$ final



9 Convert $(0+1)^*000(0+1)^*$ to NFA.

$(0+1)^*$

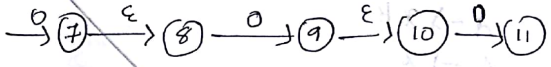
$r_1 = (0+1)^*$



$\{D\}$ on $\{A, B, D\} \neq \phi$

$= \{D\}$

$r_2 = 000$



DFA to regular language :-

Arden's theorem.

Let r and s are two regular expressions and X is a state then

$X = r + Xs$ has an unique solution.

$X = rs^*$

23/9/17

REGULAR GRAMMAR.

A grammar $G=(V,T,P,S)$ is said to be Right-linear if all productions of the form

$A \to xB$

$A \to x$

when $A,B \in V$ and $x \in T^*$

A grammer is regular is one that is right-linear or left linear.

A grammar $G=(V,T,P,S)$ is said to be ~~Right~~ Left linear if all productions of the form

DFA minimization.

→ Distinguishable and indistinguishable State.

Two states P and q of a DFA are called indistinguishable if

$\delta^*(p,w) \in F \Rightarrow \delta^*(q,w) \in F$

and $\delta^*(p,w) \notin F \Rightarrow \delta^*(q,w) \notin F$

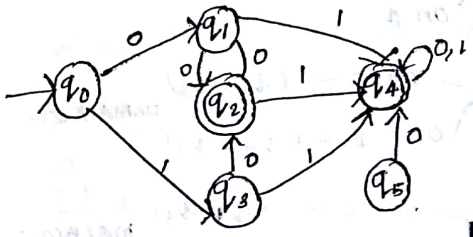
For all $w \in \Sigma^*$.

on the other hand there exist some string $w \in \Sigma^*$ such that

$\delta^*(p,w) \in F$ and $\delta^*(q,w) \notin F$ or vice versa

Then the states p and q are distinguishable

minimise below DFA Automata.

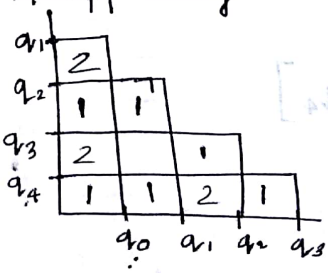


Procedure

1: Remove all inaccessible states.

Remove q_5 it is not accessible.

2 Identify distinguishable states.



Final and non final states.

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

✓ (q_0, q_1) — on 0 — (q_1, q_2) marked
on 1

✓ (q_0, q_3) — on 0 — (q_1, q_2) marked
on 1

(q_1, q_3) — on 0 — (q_2, q_2) unmarked.
on 1 — (q_4, q_2)

✓ (q_2, q_4) — on 0 — (q_1, q_4) marked.
on 1

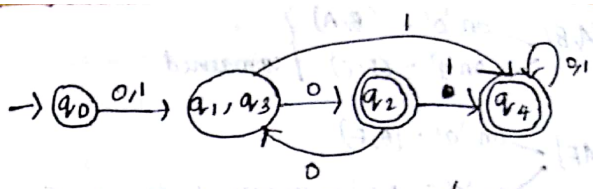
Indistinguishable states - (q_1, q_3)

$(q_0, q_1, q_2, q_3, q_4)$

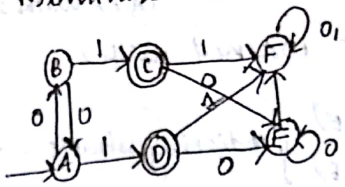
↓

$[q_0, \{q_1, q_3\}, q_2, q_4]$

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



Q Minimise below automata / TABULATION METHOD



MyHILL - Nerode Theorem.

1) Step 1: Here all state are accessible.

Step 2: partition

Non final states = {A, B, F}

final states = {C, D, E}

	A	B	C	D	E
B					
C	1	1			
D	1	1			
E	1	1			
F	2		1	1	1

(A,B) — on '0' = (B,A) } unmarked
\ on '1' = (D,C) }

✓(AF) — on '0' = (B,F)
\ on '1' = (D,F) — marked state so mark (A,F) with 2

✓(B,F) — on '0' = (A,F) — marked with 2.

(C,D) — on '0' = (E,E) } Indistinguishable.
\ on '1' = (F,F) }

(C,E) — on '0' = (E,E) } Indistinguishable
\ on '1' = (F,F) }

(D,E) — on '0' = (E,E) } Indistinguishable.
\ on '1' = (FF) }

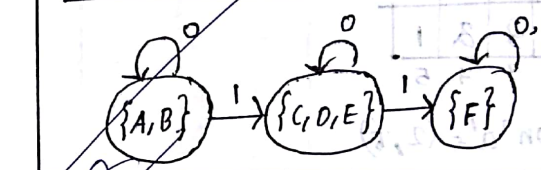
Indistinguishable states are:

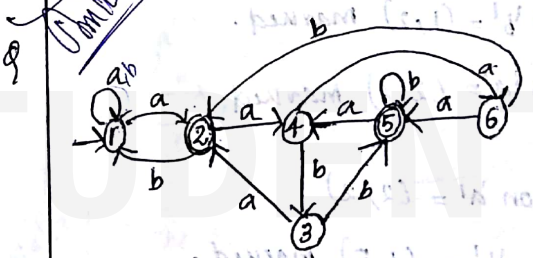
(A,B) (C,D) (C,E) (D,E)

Now form new states.

(A,B) (C,D,E) (F)

	0	1
→{A,B}	{A,B}	{C,D,E}
*{CDE}	{C,D,E}	{F}
*{F}	{F}	{F}





A) 1: All states are accessible.

2: Non final states: {1, 4, 6, 3}

final states: {2, 5}

2	1				
3	2	1			
4	2	1	2		
5	1	2	1	1	
6	2	1	2	2	1
	1	2	3	4	5

(1,6) → on 'a' = (2,5)
↳ on 'b' = (1,2) marked.

(1,4) → on 'a' = (2,6) marked
↳ 0

(1,3) → on 'a' = (2,2)
↳ on 'b' = (1,5) marked.

(2,5) → on 'a' = (4,4) unmarked
↳ on 'b' = (1,5) marked.

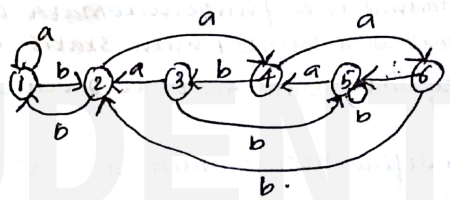
(3,6) → on 'a' = (2,5) marked with 2.

(3,4) → on 'a' = (2,6) - marked

(6,4) → on 'a' = (6,5) - marked with 2.

{1, 2, 3, 4, 5, 6}

	a	b
→ 1	1	2
2	4	1
3	2	5
4	6	3
5	4	5
6	5	2



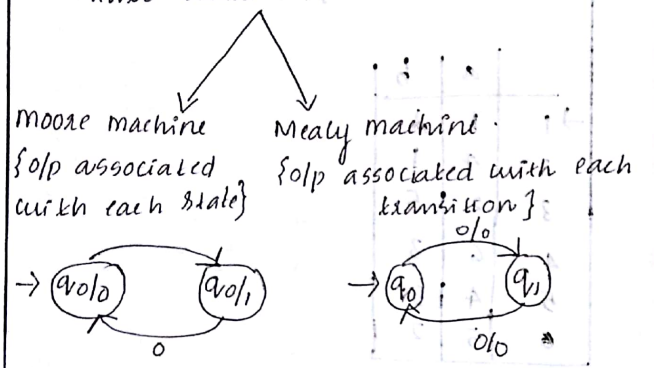
25/8/17

Finite state machine with output.

The automata that is capable for processing i/p string and generating some o/p strings is called automata with outputs or transducer

There are two models of such automata the first model is called moore machine and second model is called mealy machine.

Finite state automata



Moore machine is a finite automata in which the output is associated with states. i.e after reaching on to some state it produce some output.

We can define moore machine.

M_{moore} as 6-tuple

$M_{moore} = (Q, \Sigma, q_0, q_0, \Theta, \Delta)$

Q = set of finite states

Σ = set of alphabets

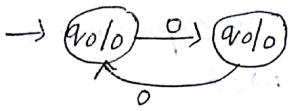
δ = transition $Q \times \Sigma \to \Theta$

$q_0 = q_0 \in Q$

Θ = finite set of output symbol

Δ = o/p mapping function $f: Q \to \Theta$

Moore machine representation.



$Q = \{q_0, q_1\}$

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

$\delta = \{(q_0, 0) = \Sigma_1$
$(q_1, 0) = q_0\}$

$q_0 = q_0$

$\Theta = \{0, 1\}$ $\Delta = F(q_0) = 0$.
$\Delta = F(q_1) = 1$

Δ transition table

M_0	0	1	o/p
$\to \varepsilon_0$	ε_1	ϕ	0
ε_1	q_0	ϕ	1

Page no: 50, 286

30/8/2017

closure properties of Regular language back side of the note book.

18/9/17

Regular Expression.

Develop regular expression for the language $L = \{a^n \mid n \geq 0\}$

$L = \{a^0, a^1, a^2, \ldots\}$

$= \{\Sigma, a, aa, aaa, \ldots\}$

Reg expression corresponds to $L = a^*$

Q Develop regular expression for the language $L = \{bbba^n \mid n \geq 0\}$.

A) $L = \{bbba^0, bbba^1, bbba^2, \ldots\}$

$\{\Sigma, bbba, bbbaa, \ldots\}$.

Regular expression corresponds to $L = bbba^*$

Q $L = \{w \mid w$ accept string starting with $ab\}$.

$L = \{ab, aba, abb \ldots\}$

$RE = L = ab(a+b)^*$

Q $L = \{w \mid w$ contains 3 consecutive zero$\}$

$\Sigma = 0, 1$

A) $(0+1)^* 000 (0+1)^*$

Q $L = \{w \mid w$ contains atleast 3 zeros$\}$

A) $(0+1)^* 0 (0+1)^* 0 (0+1)^* 0 (0+1)^*$

Regular language.

module 3

pumping ~~lemmaph~~ lemma for Regular language

n = length of strings

m = no: of states.

5 mark

Theorem:

Let L be an infinite regular language. Then there exist some +ve integer m such that any w element of L with length of $w \geq m$ can be decomposed into $w = xyz$ such that

a) length of $xy \leq m$.

b) length of ~~y ≠ ε~~ $|y| > 0$

~~c) length of $y > 0$~~

c) for all $i \geq 0$ $xy^iz \in L$