Solution

Internal Assessment Test 1 – Sept.2019

Sub:	Automata Theory and Computability							Code:	17CS54
Date:	6/09/2019	Duration:	90mins	Max Marks:	50	Sem:	V	Branch:	ISE

1 (a) Explain with example : i) Language ii)String iii) Alphabet iv) Moore Machine

i) **Language**: A language is a (finite or infinite) set of strings over a finite alphabet Σ. Let Σ = {a, b}. Σ* = {ε, a, b, aa, ab, ba, bb, aaa, aab, …}.
Some examples of languages over Σ are: ∅, {ε}, {a, b}, {ε, a, aa, aaa, aaaa, aaaaa}, {ε, a, aa, aaa, aaaa, aaaaa, …}

ii) **String:** A string is a finite sequence, possibly empty, of symbols drawn from some alphabet Σ.
Given any alphabet Σ, the shortest string that can be formed from Σ is the empty string, which we will write as ε. The set of all possible strings over an alphabet Σ is written Σ*.
Eg: ε, aabbcg, aaaaa

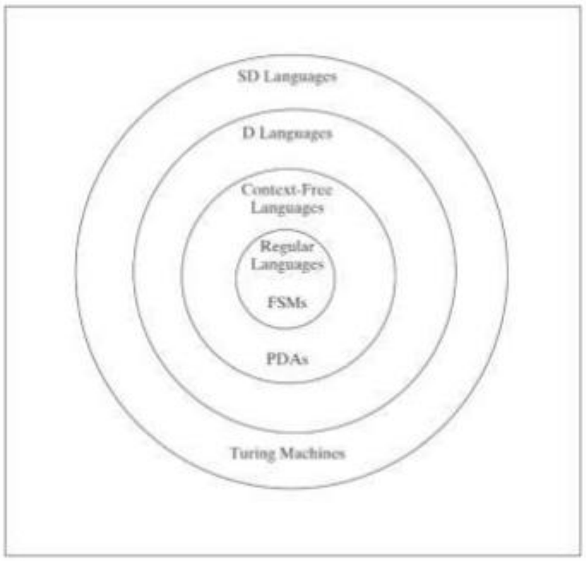
iii) **Alphabet:** An alphabet, often denoted Σ, is a finite set. We will call the members of Σ symbols or characters.
Eg: {a, b, c, …, z}
{0,1}

iv) **Moore Machine:**

A Moore machine M is a seven-tuple (K, Σ, O, δ, D, s, A), where:
• K is a finite set of states,
• Σ is an input alphabet,
• O is an output alphabet,
• s ∈ K is the start state,
• A ⊆ K is the set of accepting states (although for some applications this designation is not important), • δ is the transition function. It is function from (K × Σ) to (K), and

• D is the display or output function. It is a function from (K) to (O*).

1 (b) With a neat diagram, explain hierarchy of language classes in automata theory



The class of languages that can be accepted by some FSM regular. Many useful languages are regular, including binary strings with even parity, syntactically well-formed floating point numbers, and sequences of coins that are sufficient to buy a soda.

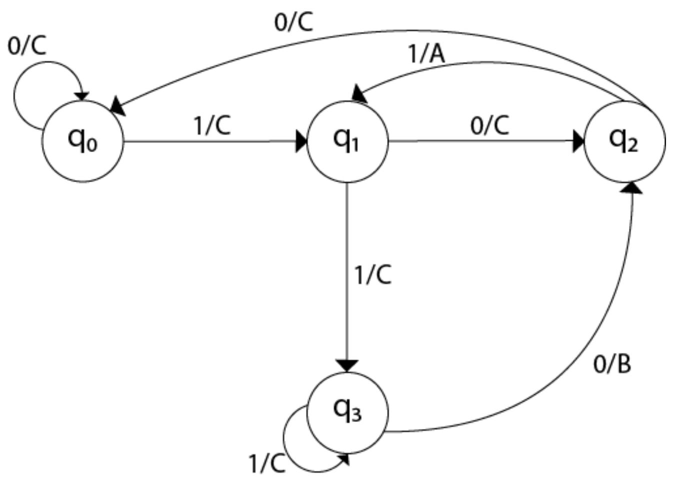
There are useful simple languages that are not regular. Consider, for example, Bal, the language of balanced parentheses. Bal contains strings like (()) and ()(); it does not contain strings like ()))(. Because it's hard to read strings of parentheses, let's consider instead the related language A^nB^n = $\{a^nb^n : n \geq 0\}$. In any string in A^nB^n, all the a's come first and the number of a's equals the number of b's. Such languages need extended memory. Such languages belong to family of context free languages. To accept CFLs we add one thing, a single stack to the FSM. We will call any machine that consists of an FSM, plus a single stack, a pushdown automaton or PDA.

We will use the Turing machine to define two new classes of languages:

• A language L is decidable iff there exists a Turing machine M that halts on all inputs, accepts all strings that are in L, and rejects all strings that are not in L. In other words, M can always say yes or no, as appropriate.

• A language L is semidecidable iff there exists a Turing machine M that halts on all inputs, accepts all strings that are in L, and fails to accept every string that is not in L. Given a string that is not in L, M may reject or it may loop forever. In other words, M can recognize a solution and then say yes, but it may not know when it should give up looking for a solution and say no.

2 (a) Design a mealy machine for a binary input sequence such that, if it has substring 101, the machine outputs A. If input has substring 110, the machine outputs B. otherwise it outputs C.



b) Design DFSM to accept the following languages

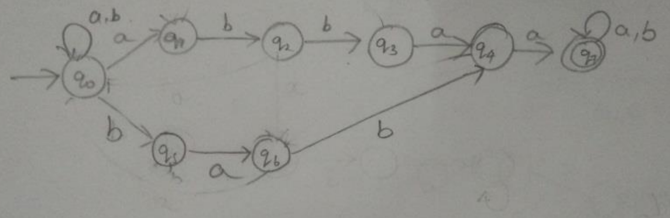
i) L={w ∈ {a, b}*, ∀ x,y ∈ {a,b}* ((w=xabbaay) or (w=xbabay))}

ii) L={w: n_a (w)≥1, n_b (w)=2}

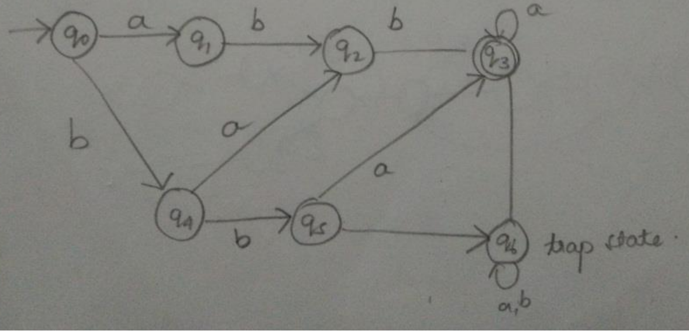
DFSM.

i) $L = \{w \in \{a,b\}^* ,\ \forall\ x,y \in \{a,b\}^* ((w = x\,abbaa\,y)$ or $(w = x\,baba\,y))\}$

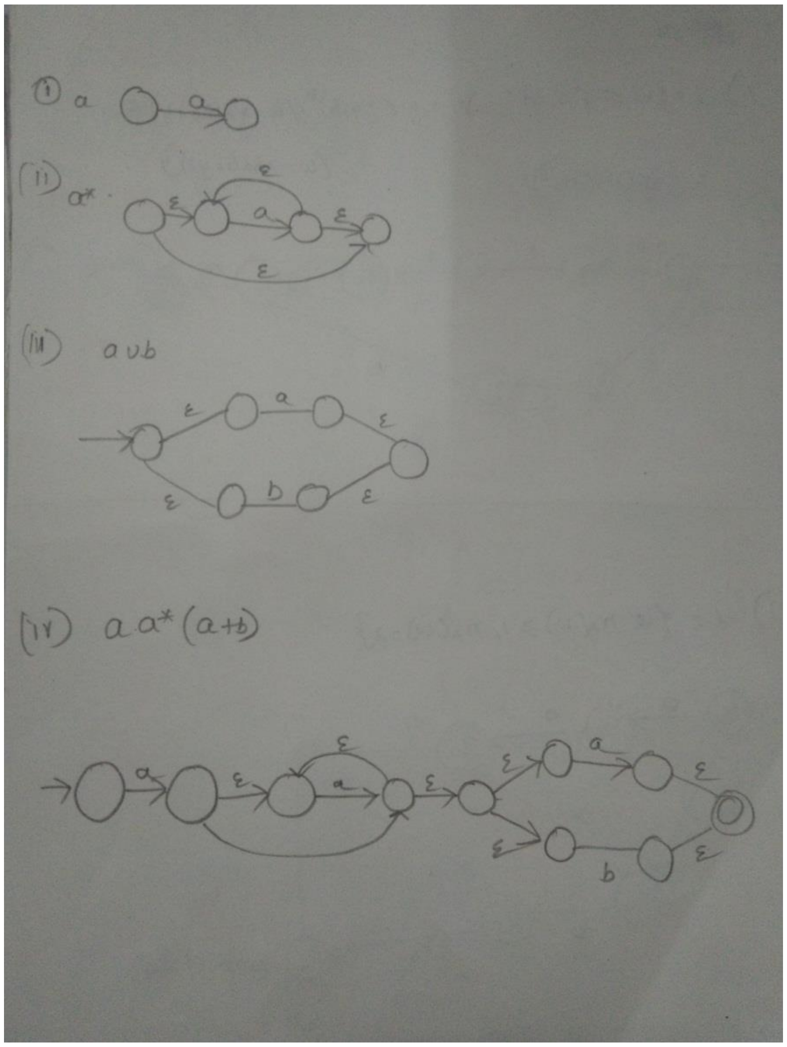
(NDFSM)



(ii) $L = \{w : n_a(w) \geq 1,\ n_b(w) = 2\}$



3 a) Design a NDFSM that accepts the language L(aa*(a+b))



3 b) Give regular expressions for the following: ∑ ={a,b}

i) L={a^nb^m | m+n is even }
(aa)*(bb)* + (aa)a*(bb)b*

ii) Strings of a's and b's having strings without ending with ab.
ε + a + b + (a + b) ∗ (aa + ba + bb).

iii) All strings that contain atleast one occurrence of each symbol.
(a+b)*a(a+b)*b(a+b)*+(a+b)*b(a+b)*a(a+b)*

4 a) Write a note on applications of regular expressions.

Because patterns are everywhere, applications of regular expressions are everywhere.

Many programming languages and scripting systems provide support for regular expression matching. Each of them has its own syntax. They all have the basic operators union, concatenation, and Kleene star.

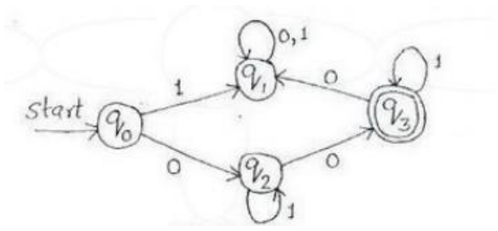
Regular expressions can be matched against the subject fields of emails to find at least some of the ones that are likely to be spam

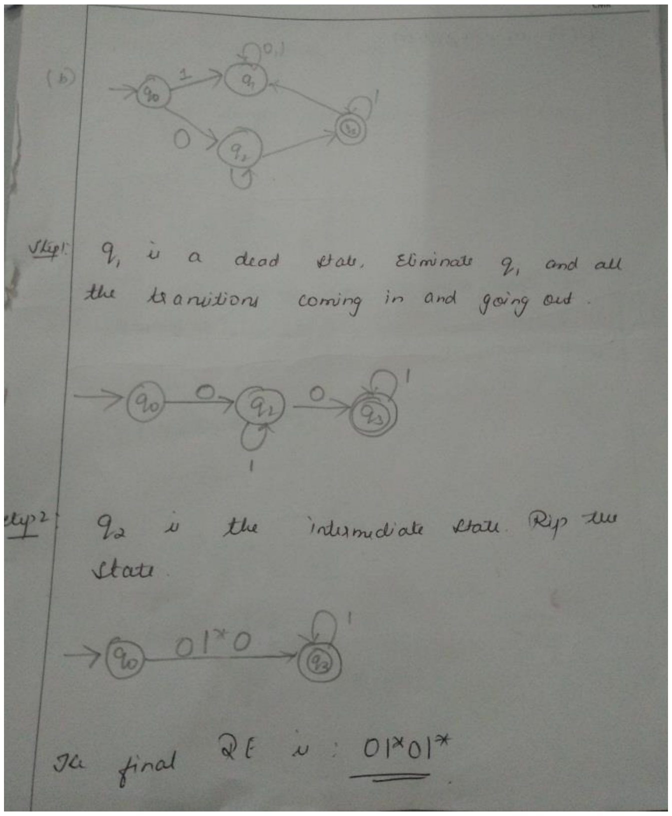
The programming language Perl, for example, supports regular expression matching.

Meaningful “words” in protein sequences are called motifs. They can be described with regular expressions

In XML, regular expressions are one way to define parts of new document types

4 b) Obtain the regular expressions for the following DFSM :




q_0
q_1
q_2
q_3
0,1
1
0
1
1
01^*0

(b)

Step1: q_1 is a dead state, Eliminate q_1 and all the transitions coming in and going out.

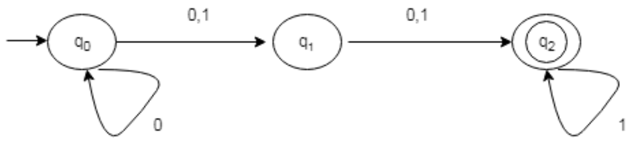
Step2: q_2 is the intermediate state. Rip the state

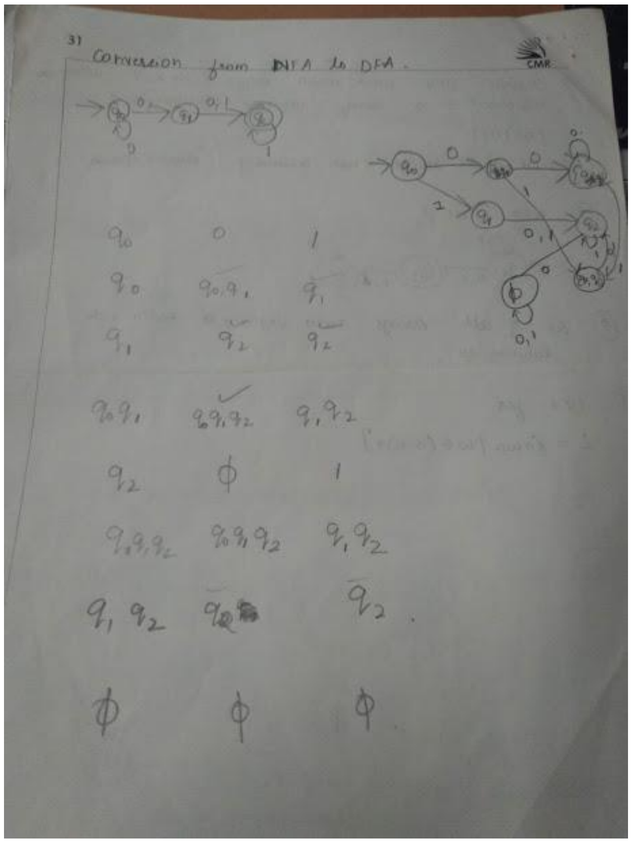
The final RE is : $\underline{01^*01^*}$

5 a) Define NDFSM. Convert the following NDFSM to its equivalent DFSM

A nondeterministic FSM (or NDFSM) M is a quintuple (K, Σ, Δ, s, A), where:

- K is a finite set of states,
- Σ is an alphabet,
- s ∈ K is the start state,
- A ⊆ K is the set of final states, and
- Δ is the transition relation. It is a finite subset of: (K × (Σ ∪ {ε})) × K.





31

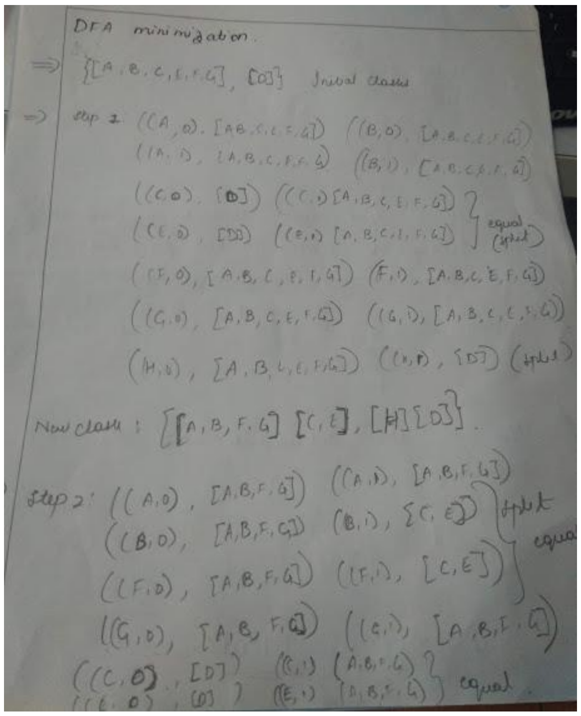
Conversion from NFA to DFA.

q_0	0	1
q_0	q_0, q_1	q_1
q_1	q_2	q_2
$q_0 q_1$	$q_0 q_1 q_2$	q_1, q_2
q_2	ϕ	1
$q_0 q_1 q_2$	$q_0 q_1 q_2$	q_1, q_2
q_1, q_2	q_2	q_2.
ϕ	ϕ	ϕ.

Define distinguishable and indistinguishable states. Minimize the following DFSM and construct minimum state equivalent of automata

S	0	1
→A	B	A

B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D



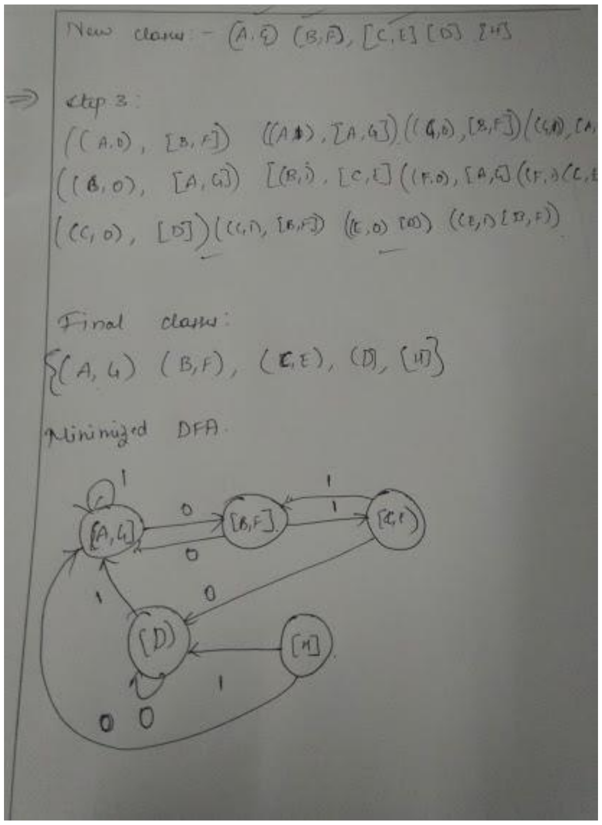
DFA minimization.

⟹ {[A, B, C, E, F, G], [D]} Initial class

⟹ step 1. ((A, 0), [A, B, C, E, F, G]) ((B, 0), [A, B, C, E, F, G])
((A, 1), [A, B, C, E, F, G]) ((B, 1), [A, B, C, E, F, G])
((C, 0), [D]) ((C, 1) [A, B, C, E, F, G]) }
((E, 0), [D]) ((E, 1) [A, B, C, E, F, G]) } equal (split)
((F, 0), [A, B, C, E, F, G]) ((F, 1), [A, B, C, E, F, G])
((G, 0), [A, B, C, E, F, G]) ((G, 1), [A, B, C, E, F, G])
((H, 0), [A, B, C, E, F, G]) ((H, 1), [D]) (split)

New class : [[A, B, F, G] [C, E], [H] [D]].

step 2: ((A, 0), [A, B, F, G]) ((A, 1), [A, B, F, G])
((B, 0), [A, B, F, G]) ((B, 1), {C, E}) } split
((F, 0), [A, B, F, G]) ((F, 1), [C, E]) } equal
((G, 0), [A, B, F, G]) ((G, 1), [A, B, F, G])
((C, 0), [D]) ((C, 1) (A, B, F, G) }
((E, 0), [D]) ((E, 1) (A, B, F, G) } equal.



New class: - (A, G) (B, F), [C, E] [D] [H]

step 3:

((A, 0), [B, F]) ((A, 1), [A, G]) ((G, 0), [B, F]) ((G, 1), [A,

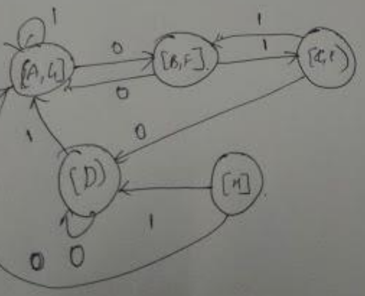
((B, 0), [A, G]) [(B, 1), [C, E] ((F, 0), [A, G] ((F, 1) (C, E

((C, 0), [D]) ((C, 1), [B, F]) ((E, 0) [D]) ((E, 1) [B, F])

Final class:

{(A, G) (B, F), (C, E), (D), (H)}

Minimized DFA.



$$\begin{tikzcd}
{[A,G]} \arrow[loop above, "1"] \arrow[r, bend left, "0"] & {[B,F]} \arrow[l, bend left, "0"] \arrow[r, bend right, "1"'] & {[C,E]} \arrow[l, bend right, "1"'] \arrow[dll, "0"] \\
{[D]} \arrow[u, "1"] \arrow[loop below, "0"] & {[H]} \arrow[l, "1"] \arrow[ul, bend left=80, "0"] &
\end{tikzcd}$$