USN

Internal Assessment Test 2 – Dec 2022

Sub:	Automata Theory and Computability				Sub Code:	18CS54	Branch:	CSE
Date:	2/12/2022	Duration:	90 mins	Max Marks: 50	Sem / Sec:	5 A,B,C		OBE
Answer any FIVE FULL Questions						MARKS	CO	RBT

1 (a) Define a Regular Expression. Obtain the regular expression for the following languages. [05] CO2 L2

i) $L=\{a^{2n}b^{2m+1} \mid n\geq 0, m\geq 0\}$

(aa)*(bb)*b

ii) All strings containing no more than 3 a's over Σ = {a,b}.

$$b^*(a \cup \varepsilon)b^*(a \cup \varepsilon)b^*(a \cup \varepsilon)b^*$$

(b) State pumping lemma for regular languages. And show that L= $\{ww^R \mid w \in (0+1)^*\}$ is not regular. [06] CO1 L3

For the language to be proved that it is regular, for any string of form w = xyz, 3 conditions must hold.

|xy|≤ k, i.e. k-1 characters can be read without revisiting any states, but kth character must take DFSM M to a state it has visited before.

y ≠ ε : Since M is deterministic, no transitions on ε

$\forall q \geq 0$ $(xy^qz \in L)$: y can be pumped (q = 0 or q>1). The resulting string should be in L

Let us take s = w^kw_r^k, so the |s| = 2k and k=|s| / 2

Let us take a string of length 6.

Let s = aabbaa

k = 3

Let's split the string as per the rules.

|xy|≤ 3 and y ≠ ε

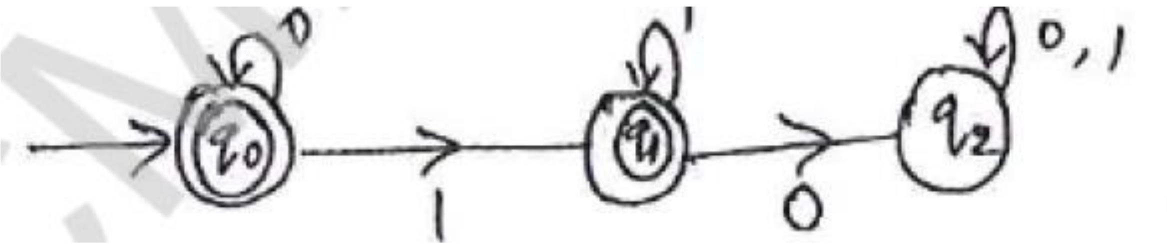
a	ab	baa
x	y	z

Now lets pump y 2times

The resulting string is aaba bbaa $\notin L = ww^R$

Hence we have proved that the language is not regular.

2 (a) Obtain the regular expression from the following FSM using state elimination method. 04] CO1 L1

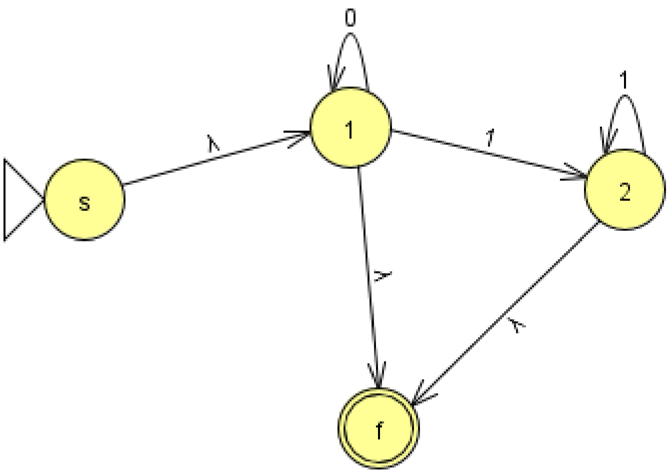


Create a new start state as initial state has incoming transitions. Connect new start state to existing start state via $\varepsilon - transitions$

Create a new final state as there are 2 final states and also there are outgoing transitions from final states. Connect existing final states to new final state via $\varepsilon - transitions$. Make existing final states as non-final.

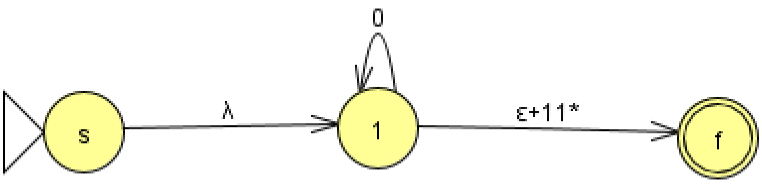
We'll rename q0 to 1 and q1 to 2.

Remove q2 as it is a dead state.



Rip 2

1-2-f



Rip 1

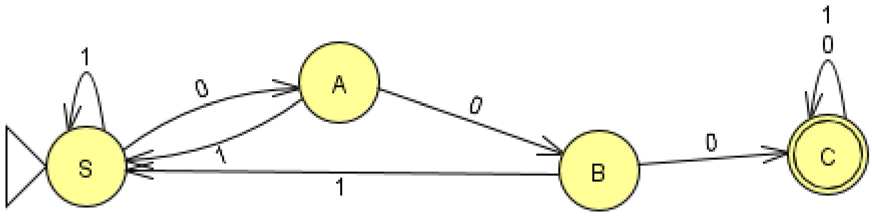
R(s,f) = R(s,f) ∪ R(s,rip)R(rip,rip)*R(rip,f)

= φ ∪ ε0*(ε∪11*) **= 0* ∪ 011***

[06] CO1 L3

(b) Write a Regular Grammar for the given language.

i) Strings of 0's and 1's having three consecutive 0's.

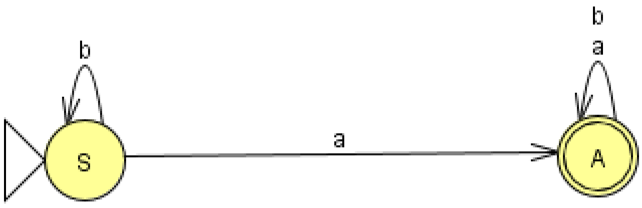


$$S \longrightarrow 1S \mid 0A$$
$$A \longrightarrow 1S \mid 0B$$
$$B \longrightarrow 1S \mid 0C$$
$$C \longrightarrow 0C \mid 1C \mid \varepsilon$$
$$G = (V, \Sigma, R, S)$$
$$G = (\{S, A, B, C\}, \{0,1\}, R, S)$$

w = 10001

$$S \Longrightarrow 1S \Longrightarrow 10A \Longrightarrow 100B \Longrightarrow 1000C \Longrightarrow 10001C \Longrightarrow 10001$$

ii) Strings of a’s and b’s with at least one a.



$$S \longrightarrow aA \mid bS$$
$$A \longrightarrow aA \mid bB \mid \varepsilon$$
$$G = (\{S, A, B\}, \{a, b\}, R, S)$$

w = baab

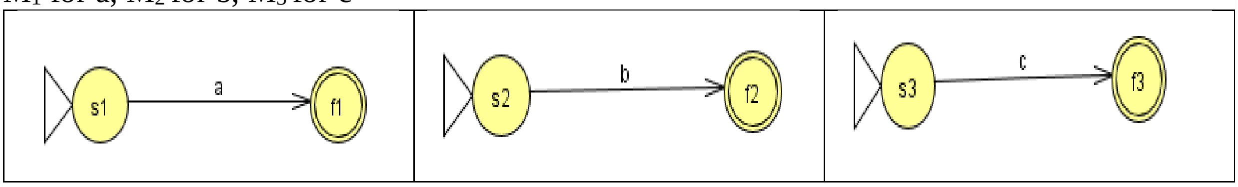
$$S \Longrightarrow bS \Longrightarrow baA \Longrightarrow baaA \Longrightarrow baabA \Longrightarrow baab$$

3 Convert the regular expression a* +b*+c* to NDFSM. [05] CO2 L3

(a) Using Kleene’s theorem

Let’s design machine for primitive types.

M_1 for a, M_2 for b, M_3 for c

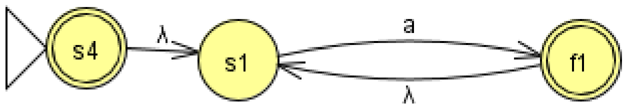


Let’s design M_4 for a*, M_5 for b*, M_6 for c*

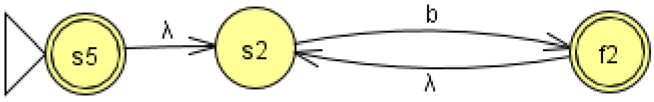
According to the theorem for Kleene closure, we create a new start state, make it accepting and connect the new start state to existing start state via $\varepsilon - transition$.

Next we connect ever accepting state in the machine to the old start state of the machine via $\varepsilon - transitions$

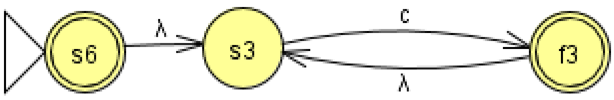
M_4



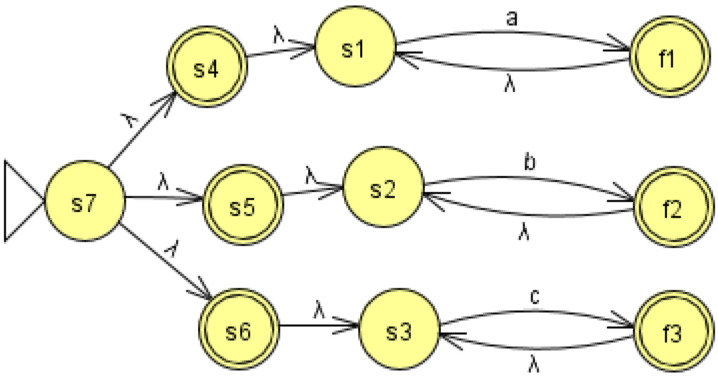
M_5



M_6



Next we design M_7 for a* +b*+c*
According to the theorem, we create a new start state, connect to each of the machines via $\varepsilon - transition$. The final states in the existing machines will also be the final state of the resulting machine

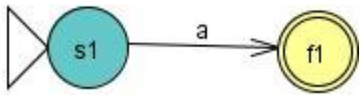


(b) Prove that regular languages are closed under Union and Intersection. [05] CO2 L2

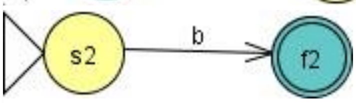
Proof that regular languages are closed under union

The proof for regular languages are closed under union is by construction. Let's take a regular expression a ∪ b. We first construct FSM M1 and M2 to accept the primitives a and b.

The proof for regular languages are closed under union is by construction. Let's take a regular expression a ∪ b. We first construct FSM M1 and M2 to accept the primitives a and b.



M_1 that accepts RE a

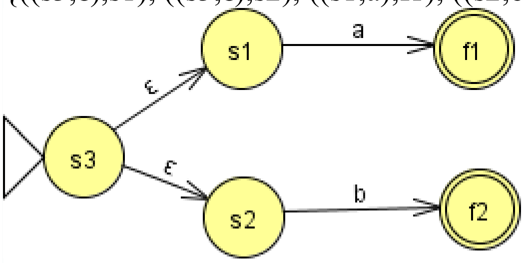


M_2 that accepts RE b

According to Kleene′s theorem, for union of two languages that are regular, M1 = (k1, Σ, δ1, s1, A1) and M2 = (k2, Σ, δ2, s2, A2), we construct a new FSM, M3 = (k3, Σ, δ3, s3, A3) such that L(M3) = L(M1) ∪ L(M2). We rename states of M1 and M2 such that k1 ∩ k2 = Φ.

Create a new start state s3 and connect the start states of M1 and M2 via ε- transitions. So M3 = ({s3} ∪ k1 ∪ k2, Σ, δ3, s3. A1 ∪ A2) where δ3 = δ1 ∪ δ2 ∪ {(s3, ε), s1) ∪ {(s3, ε), s2).

So for L(M3) = a ∪ b, we get the machine where M3 = ({s3, s1, s2, f1,f2}, {a,b}, {((s3,ε),s1), ((s3,ε),s2), ((s1,a),f1), ((s2,b),f2)}, s3, {f1,f2}.



Proof that regular languages are closed under Intersection

Given languages L and M, in order to prove that L∩M is regular, we need to prove that it is closed under complement and union. We have proved that the language is closed under union by construction.

Using DeMorgan's Laws we write intersection in terms of complement and union.

L ∩ M = ¬ ¬ (L ∩ M) = ¬ (¬ L ∪ ¬ M)

Below is proof that regular languages are closed under complement by construction.

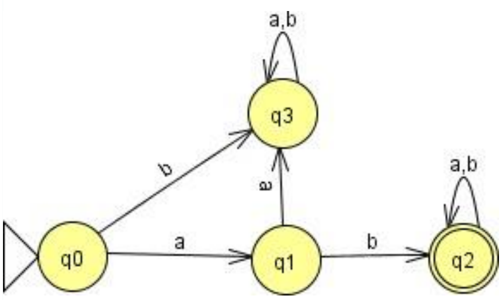
As we have proved by construction that union and complement are closed under construction, we infer using the above equation that regular languages are also closed under intersection.

Proof that Regular Languages are closed under complement

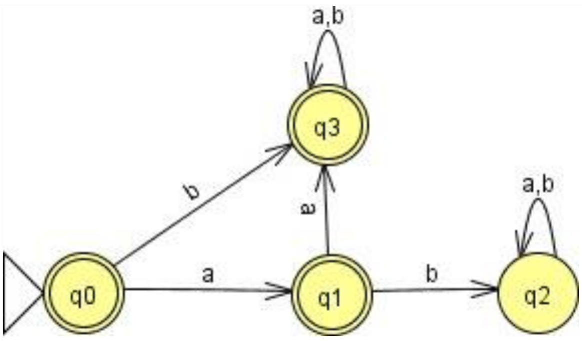
If L is a regular language, there exists a DFSM M1 = (k, Σ, δ, s, A) that accepts L. The complement of L, ¬L will be accepted by M2 = (k, Σ, δ, s, k-A).

Any NDFSM has to be converted to an equivalent DFSM, then the accepting states have to be swapped with the non-accepting states.

For example, consider that language L that accepts strings that begin with 'ab' over the alphabet, Σ={a,b}. The following DFSM, M = ({q0, q1, q2, q3}, {a,b}, {((q0,a),q1), ((q0,b),q3), ((q1,a),q3), ((q1,b),q2), ((q2,a),q2), ((q2,b),q2), ((q3,a),q3), ((q3,b),q3) }, q0, {q2}



The following DFSM accepts the complement of L, ¬L(M)



Hence, we proved the regular languages are closed under complement. TheAccepting states are k-A.
The following DFSM, not_M = ({q0, q1, q2, q3}, {a,b}, {((q0,a),q1), ((q0,b),q3),((q1,a),q3), ((q1,b),q2), ((q2,a),q2), ((q2,b),q2), ((q3,a),q3), ((q3,b),q3) }, q0,
{q0,q1,q3}

4 (a) Prove that language L={a^n| n is prime} is not regular. [04] CO1 L2

For the language to be proved that it is regular, for any string of form w = xyz, 3 conditions must hold.
|xy|≤ k, i.e. k-1 characters can be read without revisiting any states, but kth character must take DFSM M to a state it has visited before.

y ≠ ε : Since M is deterministic, no transitions on ε

∀ q ≥ 0 (xyqz ∈ L) : y can be pumped (q = 0 or q>1). The resulting string should be in L

Let us take w= a^k, so the |w| = k and k=|w|
Let us take a string of length 5.
Let w = aaaaa
k = 5
Let's split the string as per the rules.

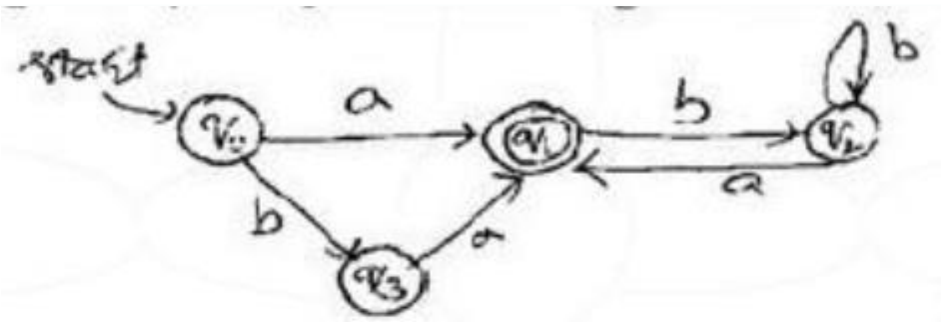
|xy|≤ 5 and y ≠ ε

a	**aaa**	**a**
x	y	z

Now lets pump y 2 times
The resulting string is a a aaa aaa a = $a^8 \notin L = a^p$
Hence we have proved that the language is not regular.

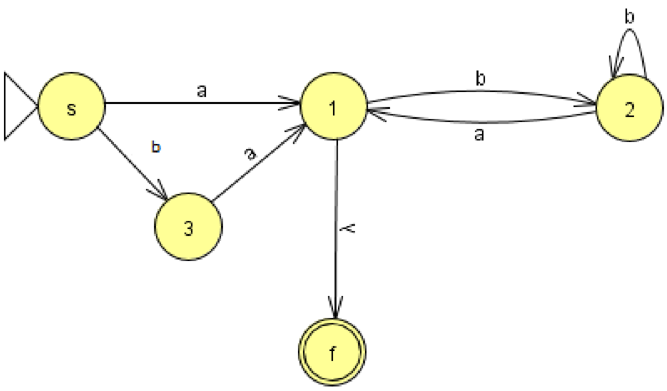
(b) Let L be the language accepted by the finite state machine. [06] CO2 L3

Indicate, for each of the following regular expressions, whether it correctly describes L:
L (a U ba)bb*a.
b. (Ɛ U b)a(bb*a)*.
c. ba U ab*a.
d. (a U ba)(bb*a)*.

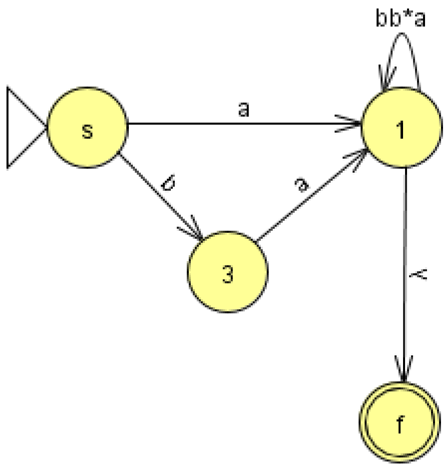


Let’s find RE based on state elimination.

Create new final state as existing final state has outgoing transitions.
Let’s rename states q0 – s, q1-1, q2-2, q3-3



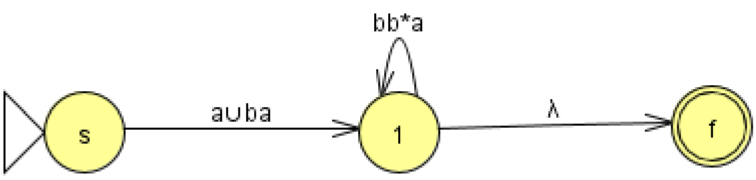
Now let’s rip 2
1-2-1



Now let’s rip 3
s-3-1

$R(s,1) = R(s,1) \cup R(s,3)R(3,3)^*R(3,1)$

$\qquad = a \cup ba$



Now Rip 1

R(s,f) = R(s,f) ∪ R(s,1)R(1,1)*R(1,f)

= φ ∪ (a ∪ ba) (bb*a)*

= (a ∪ ba)(bb*a)*

5 (a) Consider a Grammar G with production. [06] CO3 L2

S→AbB

A→aA|Ɛ

B→aB|bB|Ɛ .

Obtain the left most Derivation ,rightmost Derivation and parse tree for the string aaabab

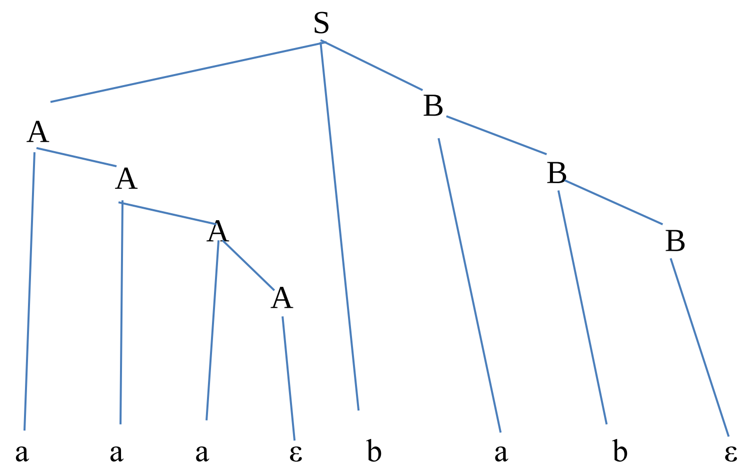
Left most derivation

$$S \Longrightarrow AbB \Longrightarrow aAbB \Longrightarrow aaAbB \Longrightarrow aaaAbB \Longrightarrow aaabB \Longrightarrow aaabaB \Longrightarrow aaababB \Longrightarrow aaabab$$

Rightmost derivation

$$S \Longrightarrow AbB \Longrightarrow AbaB \Longrightarrow AbabB \Longrightarrow Abab \Longrightarrow aAbab \Longrightarrow aaAbab \Longrightarrow aaaAbab \Longrightarrow aaabab$$

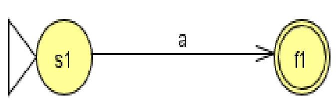
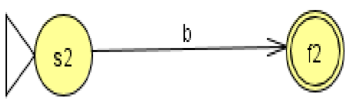
Parse Tree



(b) Convert the regular expression a* (b+a) to NDFSM [04] CO2 L3

According to Kleene’s theorem,

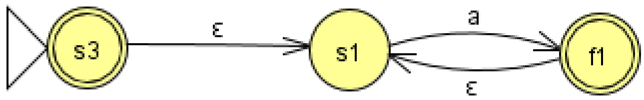
First we give FSM for primitive types a and b

 M$_1$ for accepting RE a	 M$_2$ for accepting RE b

M_3 for RE a*

According to the theorem for Kleene closure, we create a new start state, make it accepting and connect the new start state to existing start state via $\varepsilon - transition$.

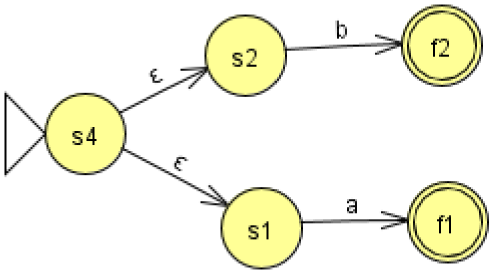
Next we connect ever accepting state in the machine to the old start state of the machine via $\varepsilon - transitions$



M_3 for accepting RE a*

M_4 for RE b+a

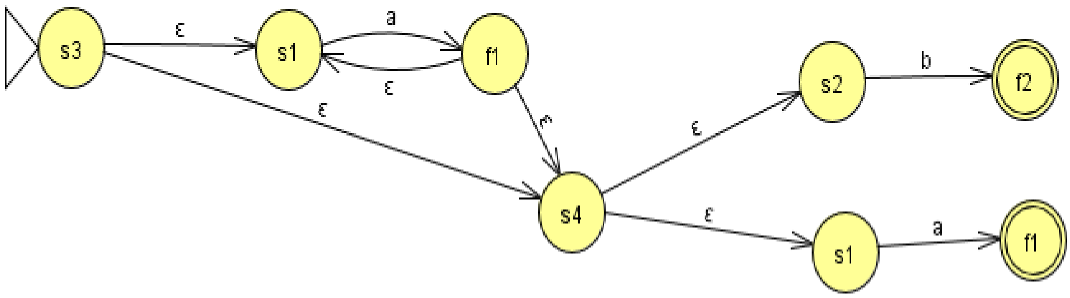
According to the theorem, for union, we create a new start state, connect to each of the machines via $\varepsilon - transition$. The final states in the existing machines will also be the final state of the resulting machine



M_4 for accepting RE b+a

M_5 for accepting a*(b+a)

According to the theorem, for concatenation, we connect every accepting state of the first machine to the initial state of the second machine via $\varepsilon - transitions$. Then we make accepting states of first machine as non-accepting. The accepting state of the final machine will be the accepting states of the second machine.



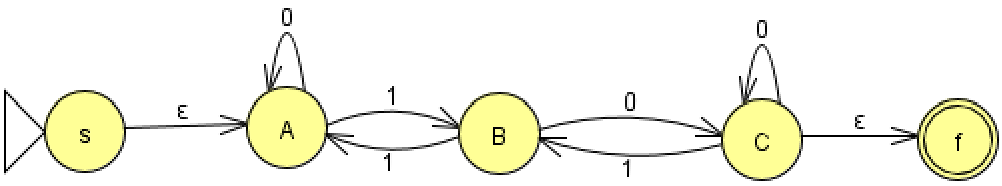
6 (a) Write the regular expression for the DFSM given in Figure 1 using state elimination method. [06] CO2 L2

(b) [05] CO2 L3

Fig. 1

As initial state has incoming transitions, we create new initial state, connect new start state to existing initial state via ε-transition.

As final state has outgoing transition, we create new final state, connect existing final state to new final state via ε-transition. We make existing final state as non-accepting.



Now we pick states to rip other than s and f

Rip B

A-B-A
C-B-C
A-B-C
C-B-A

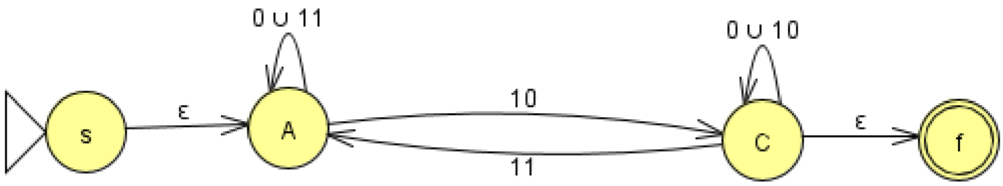
R(A,A) = R(A,A) ∪ R(A,B)R(B,B)*R(B,A)

= 0 ∪ 1φ*1 = 1ε1=0 ∪ 11

R(C,C) = R(C,C) ∪ R(C,B)R(B,B)*R(B,C)

= 0 ∪ 1 φ* 0 = 1 ε 0 = 0 ∪ 10

R(A,C) = 10   R(C,A) = 11



Rip A

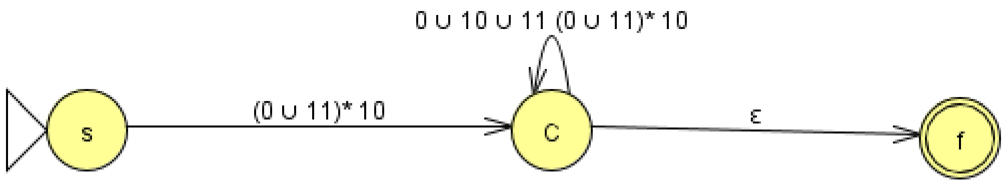
s-A-C
C-A-C

R(s,C) = R(s,C)∪ R(s,A)R(s,A)*R(A,C)

= φ ∪ ε (0 ∪ 11)* 10 = (0 ∪ 11)* 10

R(C,C) = R(C,C) ∪ R(C,A)R(A,A)*R(A,C)

= 0 ∪ 10 ∪ 11 (0 ∪ 11)* 10

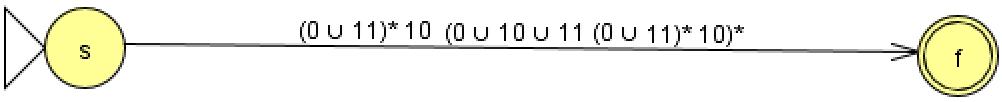


Rip C

R(s,f) = R(s,f) ∪ R(s,C) R(C,C)* R(C,f)

= φ ∪ (0 ∪ 11)* 10 (0 ∪ 10 ∪ 11 (0 ∪ 11)* 10)* ε

= (0 ∪ 11)* 10 (0 ∪ 10 ∪ 11 (0 ∪ 11)* 10)*



b Write the regular expression for the following language.

(i) Strings of a's and b's that contains abb as a substring over {a,b}*

(a ∪ b)* abb (a ∪ b)*

(ii) String of a's and b's whose 4th symbol from the left is b.

(a ∪ b) (a ∪ b) (a ∪ b) b (a ∪ b)*

7 (a) Define CFG. Consider a Grammar G with production. [06] CO 3 L3

E→ +EE | *EE | -EE | x |y

Obtain the left most Derivation ,rightmost Derivation and parse tree for the string *+-xyxy

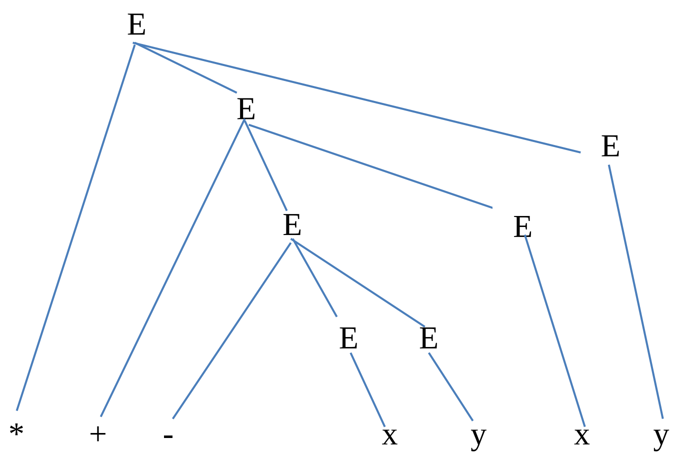
Leftmost Derivation

$$E \Longrightarrow * EE \Longrightarrow * +EEE \Longrightarrow * + - EEEE \Longrightarrow * + - xEEE \Longrightarrow * + - xyEE \Longrightarrow * + - xyxE \Longrightarrow * + - xyxy$$

Rightmost Derivation

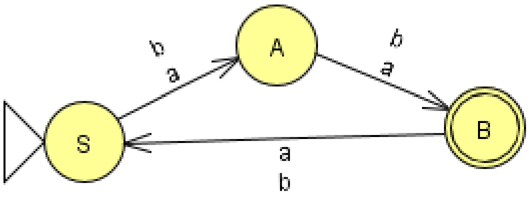
$$E \Longrightarrow * EE \Longrightarrow * Ey \Longrightarrow * +EEy \Longrightarrow * +Exy \Longrightarrow * + - EExy \Longrightarrow * + - Eyxy \Longrightarrow * + - xyxy$$

Parse Tree



b Write the regular grammar for the following language. [04] CO 3 L3

(i) L={W | W ∈ {a,b}*, |W| mod 3 = 2}



$$S \longrightarrow aA \mid bA$$
$$A \longrightarrow aB \mid bB$$
$$B \longrightarrow aS \mid bS \mid \varepsilon$$

$$G = (V, \Sigma, R, S)$$

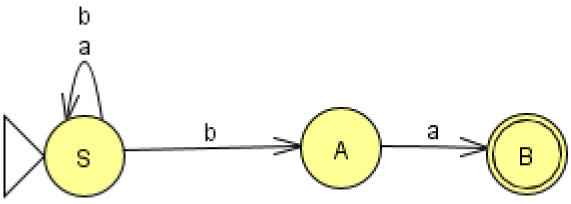
G = ({S,A,B}, {a,b}, R,S)

w= ababb |w| mod 3 = 5 mod 3 = 2

$$S \Longrightarrow aA \Longrightarrow abB \Longrightarrow abaS \Longrightarrow ababA \Longrightarrow ababb$$

(ii) String of a's and b's ending with ba.

Solution 1 – from NDFSM



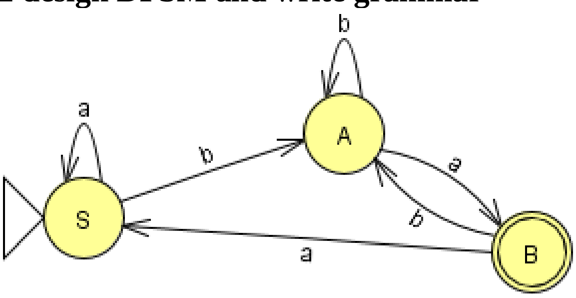
$$S \longrightarrow aS \mid bS \mid bA$$
$$A \longrightarrow aB$$
$$B \longrightarrow \varepsilon$$
$$G = (V, \Sigma, R, S)$$

G = ({S,A,B},{a,b},R, S)

w = abba

$$S \Longrightarrow aS \Longrightarrow abS \Longrightarrow abbA \Longrightarrow abbaB \Longrightarrow abba$$

Solution 2 design DFSM and write grammar



$$S \longrightarrow aS \mid bA$$
$$A \longrightarrow aB \mid bA$$
$$B \longrightarrow aS \mid bA \mid \varepsilon$$
$$G = (V, \Sigma, R, S)$$

G = ({S,A,B}, {a,b}, R, S)

w =abba

$$S \Longrightarrow aS \Longrightarrow abA \Longrightarrow abbA \Longrightarrow abbaB \Longrightarrow abba$$

CO PO Mapping

Course Outcomes		Modules covered	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	Acquire fundamental understanding of the core concepts in automata theory and Theory of Computation	**1,2,3,4 ,5**	2	3	-	-	-	2	-	-	-	-	-	-	-	3		3
CO2	Learn how to translate between different models of Computation (e.g., Deterministic and Non-deterministic and Software models).	**1,2**	2	3	2	2	2	2	-	-	-	-	-	-	-	3	3	3
CO3	Design Grammars and Automata (recognizers) for different language classes and become knowledgeable about restricted models of Computation (Regular, Context Free) and their relative powers.	**2,3**	2	3	2	2	2	2	-	-	-	-	-	-	2	-	3	-
CO4	Develop skills in formal reasoning and reduction of a problem to a formal model, with an emphasis on semantic precision and conciseness.	**3,4**	2	3	2	2	-	2	-	-	-	-	-	-	2	2	3	3
CO5	Classify a problem with respect to different models of Computation	**5**	2	3	2	2	-	3	-	-	-	-	-	-	3	3	3	3

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				