SRN

PES University, Bengaluru

(Established under Karnataka Act 16 of 2013)

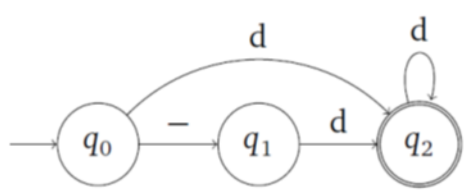
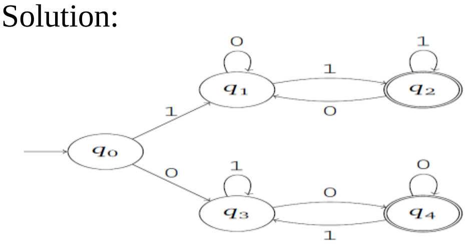
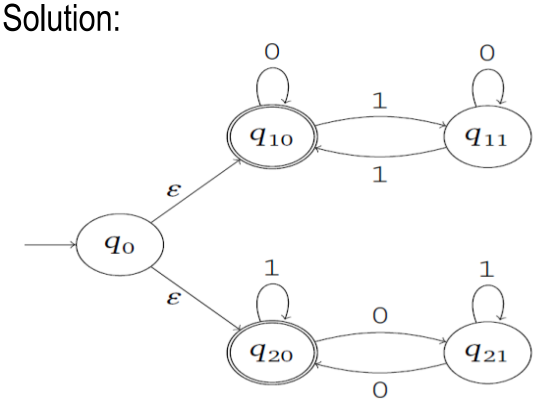
UE18Cs254

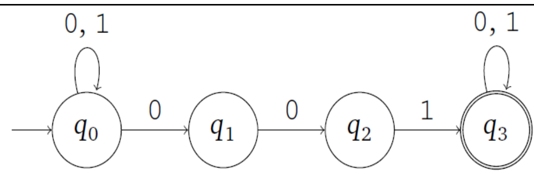
ESA(Model Pape)– B. TECH. (CSE) – IV Sem

April 2020

UE18CS254 – Theory of Computation

Time: 3 Hrs | Answer All Questions | Max Marks: 100

1	a.	Consider the problem of determining if a string is an integer in the following format: an optional minus sign followed by at least one digit. Design a finite automaton for this problem Solution:  	04
	b.	Construct A DFA over the alphabet {0,1}*for the language of strings of length at least two that begin and end with the same symbol Solution:  	06
	c.	Construct An NFA over the alphabet{0,1}* for the language of strings that contain either a number of 0's that's even or a number of 1's that's even. Solution:  	04
	d.	Consider the following NFA, There are three possible sequences of states that this NFA could go through while reading all of the string 001001. What are they? Does the NFA accept this string?	06



Solution:

	$\xrightarrow{0}$		$\xrightarrow{0}$		$\xrightarrow{1}$		$\xrightarrow{0}$		$\xrightarrow{0}$		$\xrightarrow{1}$	
q_0		q_0		q_0		q_0		q_0		q_0		q_0
q_0		q_0		q_0		q_0		q_1		q_2		q_3
q_0		q_1		q_2		q_3		q_3		q_3		q_3

The NFA accepts because the last two sequences end in the accepting state.

2. a. Give regular expression for The language of strings that contain a number of 1's that's a multiple of k — 02

Solution:

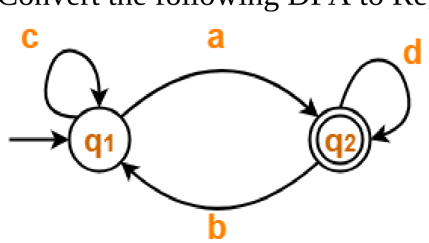
$((0^*1)^k)^*0^*$

b. Give regular expressions for the language of strings that contain at least k 1's. Do this in general, for every k >= 0 — 04

Solution:

$((0+1)^*1)^k\,(0+1)^*$

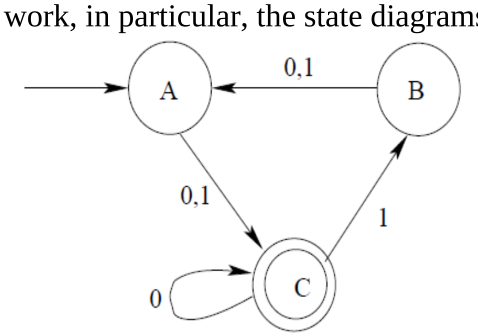
c. Convert the following DFA to Regular expression — 04



Solution:

c*a(d+bc*a)*

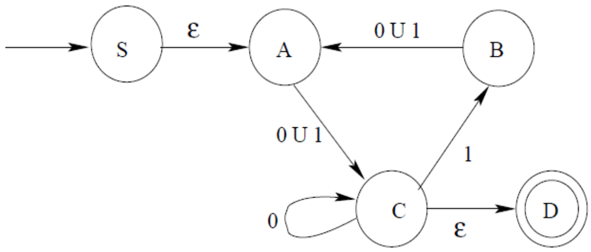
d. Find a regular expression for the language recognized by this machine, Show all your work, in particular, the state diagrams after the removal — 08



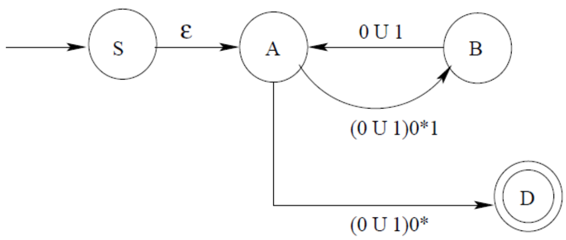
<table>
<tr><td></td><td></td><td>

Solution:

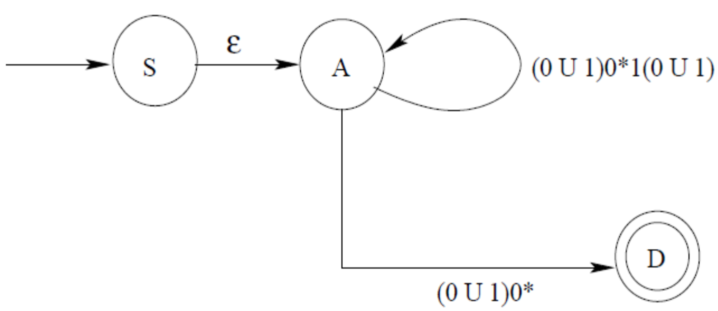
before state removal:



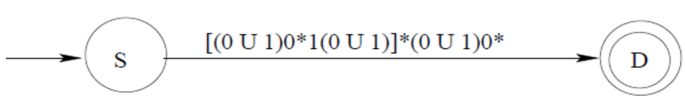
After removing state C



After removing state: B



After removing state A



Regular expression representing the language recognized by the original DFA:

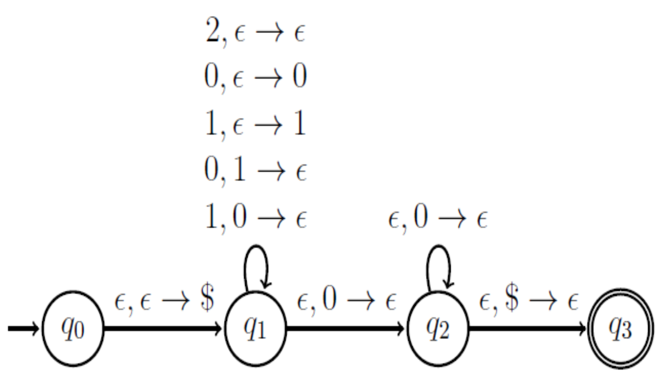
{(0 U1)0*1)(0 U1)}*(0 U1)0*

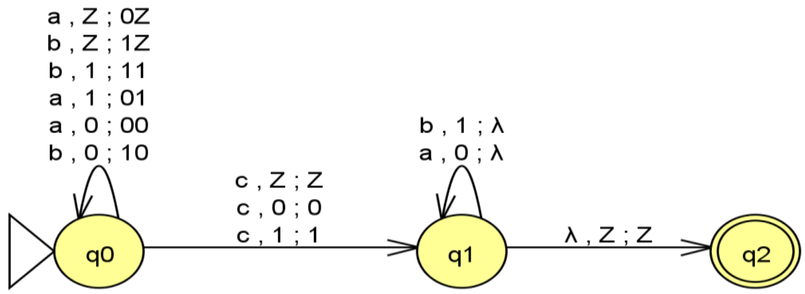
</td><td></td></tr>
<tr><td></td><td>e</td><td>

1) List the operations that the regular languages are closed under

Solution:

</td><td>02</td></tr>
</table>

		Regular languages are closed under following operations. 1) Union 2) Intersection 3) Complement 4) Set Difference 5) Reverse operation	
3	a.	Let L1 ={w ∈ {0,1,2}* : w contains more 0’s than 1’s. Give a transition diagram for a PDA to recognize L1 Solution: 	06
	b.	Let 2 ={w ∈ {0,1,2}* : w has same number of 1’sand 2’s. Give a CFG to generate L2 Solution: S-> λ | S0S| S1S2S | S2S2S	04
	c.	State whether or not L1 ∩ L2 from the abovequestion a and b is context-free. Justify your answer carefully. No, it is not context-free. L1 ∩ L2 = {w ∈ {0,1,2}* | |w|0 > |w|1 = |w|2} Assume (to the contrary) that L1∩ L2 is context-free, and has pumping length p. Choose w $0^{p+1}1^p2^p$ to pump. There are two cases to consider for w = uvxyz. vy contains no 2: Then vy contains at least one 0 or 1 because |vy|. But uv^0xy^0 z.. = uxz has at most as many 0's or fewer 1's (or both) than 2's, so the result is not in L1∩ L2. vy does contain a 2: Then it cannot contain a 0, because |vxy|<=p. so uv^2 xy^2z has atleast as many 2's as 0's, again yielding a string not in L1∩ L2. So the string w cannot be pumped, a contradiction. So L1∩L2 is not context-free.	06
	d.	Construct a PDA for the following language Odd palindromes wcwR over {a, b, c} where w = (a + b)* Solution:	04



4 a. Use the Pumping lemma to show that the language $\{a^i b^j c^k \mid i<=j<=k\}$ is not context free . The alphabet is {a,b,c}. **08**

Solution: Let L denote that language and suppose that L is context-free. Let p be the pumping length. Consider the string w = $a^p b^p c^p$. Clearly, w∈L and |w|≥ p. Therefore, according to the Pumping Lemma, w can be written as uvxyz where

1. Vy ≠ λ

2. $Uv^k xy^k z \in L$, for every k≥0.

There are three cases to consider. First, suppose that either v or y contains more than one type of symbol. Then $uv^2 xy^2 z \notin L$ because that string is not even in a∗b∗c∗.

In the other two cases, v and y each contain only one type of symbol. The second case is when v consists of a's. Then, since y cannot contain both b's and c's, $uv^2 xy^2 z$ contains more a's than b's or more a's than c's. This implies that $uv^2 xy^2 z \notin L$

The third case is when v does not contain any a's. Then y can't either. This implies that $uv^0 xy^0 z$ contains less b's than a's or less c's than a's

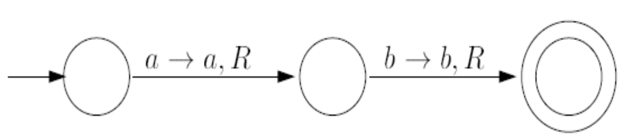
Therefore, $uv^0 xy^0 z \notin L$. In all cases, we have that $uv^k xy^k z \notin L$ for some k≥0. This is a contradiction and proves that L is not context-free.

b. List the operations that the Context free languages are closed under **02**

Solution:

Context free languages are closed under following operations.

1) Union
2) Concatenation
3) Closure
4) Intersection
5) Closure

	c.	Convert the following grammar to Chomsky Normal Form $S \rightarrow abS \mid baS \mid \lambda$ *Solution:* *S-> AD* *D->BS* *S->BC* *C->AS* *S->AB* *S-> BA* *B->b* *A->a*	04
	d.	Convert the following grammar to Greibach Normal Form $S \rightarrow AB$, $A \rightarrow aA \mid bB \mid b$, $B \rightarrow b$ *Solution:* Only the *S* production is not in GNF. Substituting for *A*, we get: $S \rightarrow aAB \mid bBB \mid bB$, $A \rightarrow aA \mid bB \mid b$, $B \rightarrow b$	04
	e.	$S \rightarrow aSb \mid ab$ Solution: $S \rightarrow aSb \mid ab$	02
5	a.	Design a Turing Machine that accepts the following language. L = {ab(a + b)*} Solution:  The machine doesn't have to read all the input string in order to accept the string. This is because we know the input string alphabet which is {a,b} and after string ab any string made	04

from the input alphabet can follow

b. Let L5 = $\{a^i b^j c^k \mid 0 \le i \le j \le k\}$. Describe a Turing Machine that decides L5. **08**

Solution:

1) If the input is the empty string accept. Otherwise, shift the input rightwards by 1 cell and mark the left end of the tape with \$. While doing this, we can check whether the input is of the form $a^ib^jc^k$ for i, j, k ≥ 0 and reject if it is not. At the end we return to the start of the tape.

2) Now we make sure that i ≤ j and i ≤ k: While there is an a on the tape (scan right to find one), find b afterwards and cross it and find a c afterwards and cross it. If there is no b or no c reject. At the end of each iteration returns to the start of the tape.

3) Now we make sure that j ≤ k: While there is a b on the tape (scan right to find one), find a c afterwards and cross it. If there is no c reject. At the end of each iteration returns to the start of the tape.

4) If we complete all this without rejecting, then we accept.

c. How would one simulate a PDA on a Turing machine? Please do not write the Turing machine itself, but rather write the key idea in plain English. **08**

Solution:

We show how a PDA can be simulated by a non-deterministic two-tape TM. This would in a second step be transformed into a normal (deterministic, one-tape) TM using the transformations shown in the lecture.

We use the second tape of the TM to represent the stack of the PDA, whereas the first tape is only used to read the input of the PDA. The states of the TM basically represent the states of the PDA. In addition, we need some auxiliary states (see below). The TM works as follows: Whenever the PDA would read a symbol, the TM must read the symbol under the head on the first tape and moves the head one step to the right. Whenever the PDA uses the empty symbol as input symbol, the head on the first tape keeps its position. At the same time, the TM performs the appropriate action on the second tape in such a way the the head is always positioned on the “top-most” symbol:

• If the PDA consumes a symbol from the stack but does not write a new one, the TM deletes the symbol under the head and moves the head one position to the left.

• If the PDA consumes a symbol from the stack and writes a new one, the TM overwrites the symbol below the head with the new one.

• If the PDA does not consume a symbol from the stack but writes a new one, the TM moves one step to the right and writes the new symbol. For this we actually need to perform two

		steps which can be done using some auxiliary states (we need to remember the symbol to write and the current state of the TM). • If the PDA does not change the stack, the TM does not change the second tape. This TM accepts the input of the PDA iff it stops in an accept state (if there are several accepting states in the PDA, we use an additional transition that brings the TM from these states to its single accept state).	