USN | | | | | | | | | | |

Internal Assessment Test II – October 2019

Sub:	Automata Theory and Computability				Sub Code:	17CS54	Branch :	ISE	
Date:	14/10/2019	Duration:	90 min's	Max Marks:	50	Sem / Sec:	V A & B		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 (a)	State and prove pumping lemma for regular languages. Prove that the given language is not regular- $\{w \in \{0,1\}^* : \#_0(w) \neq \#_1(w)\}$	[10]	L3	CO3
2 (a)	Let $L = \{w \in \{a, b\}^*$: every a in w is immediately followed by at least one b. i) Write a regular expression that describes L. ii) Write a regular grammar that generates L. iii) Construct an FSM that accepts L.	[6]	L2	CO3
(b)	Explain with example Inherently ambiguous grammar.	[4]	L1	CO3
3(a)	Consider the following grammar S->ABC|BaB A->aA|BaC|aaa B->bBb|a|D C->CA|AC D ->ε i) Eliminate ε rules ii) Eliminate any unit rules from the resulting grammar. iii) Eliminate any useless symbols from the resulting grammar	[6]	L2	CO3
(b)	Obtain the following grammar in GNF S -> ABA | AB | BA | AA | A | B A -> aA | b B -> bB | b	[4]	L2	CO3
4) (a)	Design a context free grammar for the following : i. $L=\{0^m1^m2^n \mid m\geq1, n\geq1\}$ ii. $L=\{a^ib^j \mid i\neq j, i\geq0, j\geq0\}$	[6]	L3	CO3
(b)	What is ambiguity? Show that the following grammar is ambiguous S->aB|bA A->aS|bAA|a B->bS|aBB|b	[4]	L2	CO3
5 (a)	Let G be the grammar S->aB|bA A->a|aS|bAA B->b|bS|aBB For the string aaabbabbba find a i) Leftmost derivation ii) Rightmost derivation ii) Parse tree	[6]	L2	CO3

(b)	Eliminate recursion from the given grammar A --> B x y | x B --> C D C --> A | c D --> d	[4]	L3	CO3
6 (a)	Define Chomsky Normal form. Obtain the following grammar in CNF. S->ASB|ε A->aAS|a B->SbS|A|bb	[8]	L3	CO3
(b)	Show a regular grammar for the given language :{ *w* ∈ {a, b}*: *w* does not end in aa}.	[2]	L3	CO3

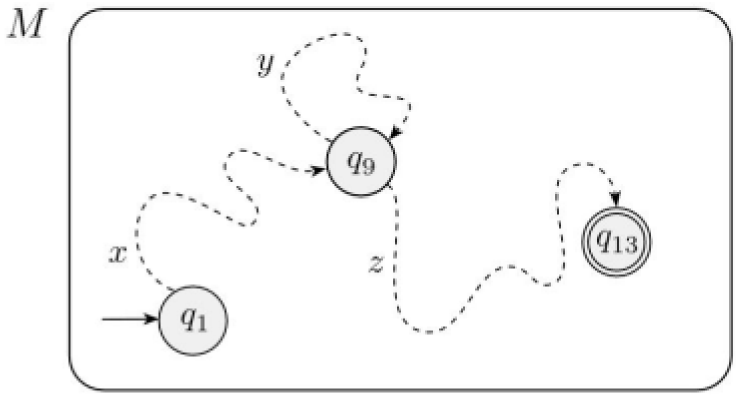
Solutions

1(a)State and prove pumping lemma for regular languages. Prove that the given language is not regular- $\{w \in \{0,1\}^* : \#_0(w) \neq \#_1(w)\}$

Pumping Lemma (for Regular Languages) : If A is a Regular Language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into 3 pieces, s = xyz, satisfying the following conditions:

a. For each $i \geq 0$, $xy^iz \in A$,
b. $|y| > 0$, and
c. $|xy| \leq p$

Proof :Let $M = (Q, \Sigma, \delta, q, F)$ be a DFA recognizing A and p be the number of states of M. Let $s = s_1 s_2 ...s_n$ be a string in A with length n, where $n \geq p$. Let $r_1, ..., r_{n+1}$ be the sequence of states M enters when processing s. $r_{i+1} = \delta(r_i, s_i)$ for $1 \leq i \leq n$. The sequence has length n+1, which is at least p + 1. Among the first p + 1 elements in the sequence, two must be the same state, via the pigeonhole principle. The first is called r_j , and the second is r_l



Because r_l occurs among the first p + 1 places in a sequence starting at r_1 , we have $l \leq p + 1$. Now let $x = s_1 ...s_{j-1}$, $y = s_j ...s_{l-1}$, and $z = s_l ...s_n$. As x takes M from r_1 to r_j , y takes M from r_j to r_l , and z takes M from r_l to r_{n+1}, which is an accept state, M must accept xy^iz for $i \geq 0$. We know $j \neq l$, so $|y| > 0$; and $l \leq p + 1$, so $|xy| \leq p$. Thus, we have satisfied all conditions of the pumping lemma.

Not regular. This one is quite hard to prove by pumping. Since so many strings are in *L*, it's hard to show how to pump and get a string that is guaranteed not to be in *L*. Generally, with problems like this, you want to turn them into problems involving more restrictive languages to which it is easier to apply pumping. So: if *L* were regular, then the complement of *L*, *L′* would also be regular.
$L' = \{w \in \{0, 1\}^* : \#0(w) = \#1(w)\}$.
It is easy to show, using pumping, that *L′* is not regular: Let $w = 0k1k$. *y* must occur in the initial string of 0's, since $|xy| \leq k$. So $y = 0i$ for some $i \geq 1$. Let *q* of the pumping theorem equal 2 (i.e., we will pump in one extra copy of *y*). We now have a string that has more 0's than 1's and is thus not in *L′*. Thus *L′* is not regular. So neither is *L*. Another way to prove that *L′* isn't regular is to observe that, if it were, $L'' = L' \cap 0^*1^*$ would also have to be regular. But L'' is 0^n1^n, which we already know is not regular.

2 a) Let $L = \{w \in \{a, b\}^*$: every a in *w* is immediately followed by at least one b.

i) Write a regular expression that describes *L*.

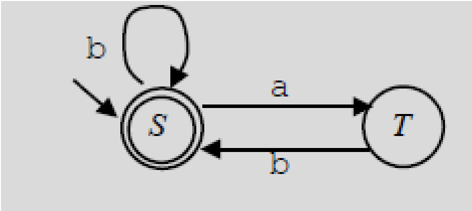
$(ab \cup b)^*$

ii) Write a regular grammar that generates *L*.

$S \rightarrow bS$

$S \rightarrow \mathrm{a}T$
$S \rightarrow \varepsilon$
$T \rightarrow \mathrm{b}S$

iii) Construct an FSM that accepts L.



b) Explain with example Inherently ambiguous grammar.

In many cases, for an ambiguous grammar G, it is possible to construct a new grammar G′ that generates L(G) and that has less (or no) ambiguity. Unfortunately, it is not always possible to do this. There exist context-free languages for which no unambiguous grammar exists. We call such languages inherently ambiguous.

Let $L = \{\mathrm{a}^i\mathrm{b}^j\mathrm{c}^k : i, j, k \geq 0, i = j \text{ or } j = k\}$. An alternative way to describe it is $\{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^m : n, m \geq 0\} \cup \{\mathrm{a}^n\mathrm{b}^m\mathrm{c}^m : n, m \geq 0\}$. Every string in L has either (or both) the same number of a's and b's or the same number of b's and c's. L is inherently ambiguous. One grammar that describes it is $G = (\{S, S_1, S_2, A, B, \mathrm{a}, \mathrm{b}, \mathrm{c}\}, \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}, R, S)$, where:

$R = \{ S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow S_1\mathrm{c} \mid A$ /* Generate all strings in $\{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^m : n, m \geq 0\}$.
$A \rightarrow \mathrm{a}A\mathrm{b} \mid \varepsilon$
$S_2 \rightarrow \mathrm{a}S_2 \mid B$ /* Generate all strings in $\{\mathrm{a}^n\mathrm{b}^m\mathrm{c}^m : n, m \geq 0\}$.
$B \rightarrow \mathrm{b}B\mathrm{c} \mid \varepsilon \}$.

Now consider the strings in $\mathrm{A}^n\mathrm{B}^n\mathrm{C}^n = \{\mathrm{a}^n\mathrm{b}^n\mathrm{c}^n : n \geq 0\}$. They have two distinct derivations, one through S_1 and the other through S_2. It is possible to prove that L is inherently ambiguous: given any grammar G that generates L there is at least one string with two derivations in G.

3(a) Consider the following grammar

S->ABC|BaB
A->aA|BaC|aaa
B->bBb|a|D
C->CA|AC
D ->ε

Eliminate ε rules
Eliminate any unit rules from the resulting grammar.
Eliminate any useless symbols from the resulting grammar

(a)

$S \to ABC \mid BaB$

$A \to aA \mid BaC \mid aaa$

$B \to bB \mid b \mid a \mid D$

$C \to CA \mid AC$

$D \to \varepsilon$

i) Eliminate ε rules.

Nullable set = $\{D, B\}$

After elimination of D

$S \to ABC \mid BaB \mid AC \mid aB \mid Ba$ (since B is nullable)

$A \to aA \mid BaC \mid aaa$

$B \to bB \mid b \mid a$

$C \to CA \mid AC$

i) There are ~~no~~ unit rules.

i) Eliminate useless symbols.

Generating rules $A \to aaa$
$B \to b \mid a$

$S \to BaB \mid aB \mid Ba$ (since B is generating

$A \to aA \mid aaa$ (since A is generating

$B \to bB \mid$ since B is generation

C is non generating (eliminate it)

Reachable - $S \to BaB \mid aB \mid Ba$

$B \to bB \mid b \mid a$

A is not reachable eliminate it.

∴ Final grammar

$S \to BaB \mid aB \mid Ba$

$B \to bB \mid b \mid a$ //

(b) GNF

$S \to ABA \mid AB \mid BA \mid AA \mid B \mid A$

$A \to aA \mid b$ } already in GNF

$B \to bB \mid b$

replace A in the rule with $A \to aA \mid b$.

replace B with $bB \mid b$.

$S \to aABA \mid bBA \mid aAB \mid bB \mid bBA \mid bA \mid aAA \mid bA \mid aA \mid b$

$A \to aA \mid b$

$B \to bB \mid b$ //

4) (a) Design a context free grammar for the following :

i. $L=\{0^m1^m2^n \mid m\geq 1, n\geq 1\}$

ii. $L=\{a^ib^j \mid i\neq j, i\geq 0, j\geq 0\}$

(b) What is ambiguity? Show that the following grammar is ambiguous

Sometimes a grammar may produce more than one parse tree for some (or all) of the strings it generates. When this happens we say that the grammar is ambiguous. More precisely, a grammar *G* is ***ambiguous*** iff there is at least one string in *L*(*G*) for which *G* produces more than one parse tree

S->aB|bA
A->aS|bAA|a
B->bS|aBB|b

aaabbabba → aaabbabbA (rule 2a)
aaabbabbA → aabbabS (rule 1b)
aabbabS → aabbaB (rule 3b)

aabbaB → aabbS (rule 1a)
aabbS → aabB (rule 3b)
aabB → aaBB (rule 3a)
aaBB → aB (rule 3c)
aB → S (rule 1a)

5) Let G be the grammar

S->aB|bA
A->a|aS|bAA
B->b|bS|aBB

For the string aaabbabbba find a

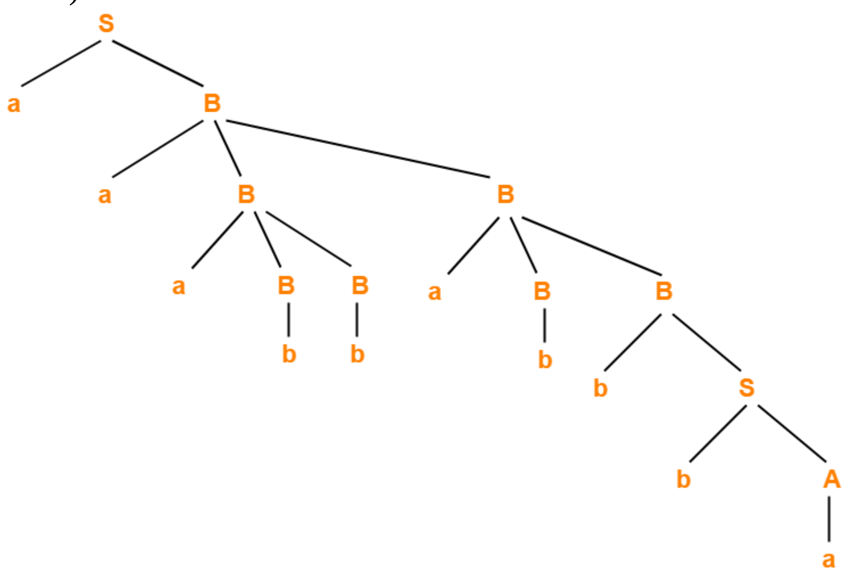
i) Leftmost derivation

S→ a**B**
→ aa**B**B (Using B → aBB)
→ aaa**B**BB (Using B → aBB)
→ aaab**B**B (Using B → b)
→ aaabb**B** (Using B → b)
→ aaabba**B**B (Using B → aBB)
→ aaabbab**B** (Using B → b)
→ aaabbabb**S** (Using B → bS)
→ aaabbabbb**A** (Using S → bA)
→ aaabbabbba (Using A → a)

ii) Rightmost derivation

S → a**B**
→ aaB**B** (Using B → aBB)
→ aaBaB**B** (Using B → aBB)
→ aaBaBb**S** (Using B → bS)
→ aaBaBbb**A** (Using S → bA)
→ aaBa**B**bba (Using A → a)
→ aa**B**abbba (Using B → b)
→ aaaB**B**abbba (Using B → aBB)
→ aaa**B**babbba (Using B → b)
→ aaabbabbba (Using B → b)

iii) Parse tree



Leftmost Derivation Tree

b) Eliminate recursion from the given grammar

A --> B x y | x
B --> C D
C --> A | c
D --> d

Indirect Recursion:

$A \to Bxy \mid x$
$B \to CD$
$C \to A \mid c$
$D \to d$.

There is no direct recursion.

Replace $B \to CD$ in $A \to Bxy \mid x$.

$A \to CDxy \mid x$
$B \to CDxy \mid x$
$C \to A \mid c$
$D \to d$.

Replace $C \to A \mid c$ in $A \to CDxy \mid x$.

$A \to ADxy \mid x \mid cDxy$

No other rules are having recursion.

$\therefore$ $A \to ADxy \mid cDxy \mid x$
$B \to CD$
$C \to A \mid c$
$D \to d$.

$A \to ADxy \mid cDxy \mid x$ Recursion rule
$A \to A\alpha \mid \beta_1 \mid \beta_2$

Final grammar.

$A \to \overbrace{cDxy}^{\beta_1}A' \mid \overbrace{x}^{\beta_2}A'$

$A' \to \underbrace{Dxy}_{\alpha}A' \mid \varepsilon$

$B \to CD$

$C \to A \mid c$

$D \to d$

6 a) Define Chomsky Normal form. Obtain the following grammar in CNF.

S->ASB|ε
A->aAS|a
B->SbS|A|bb

$S \to ASB \mid \varepsilon$
$A \to aAS \mid a$
$B \to SbS \mid A \mid bb$

(i) Nullable variables

$S = \{S\}$

Elimination epsilon rules

$S \to ASB \mid AB$
$A \to aAS \mid aA \mid a$
$B \to SbS \mid A \mid bb \mid Sb \mid bS$

Elimination of unit rules

$S \to ASB \mid AB$
$A \to aAS \mid aA \mid a$
$B \to SbS \mid aAS \mid aA \mid a \mid bb \mid Sb \mid bS$

Elimination of mixed rules

$S \to ASB \mid AB$
$A \to T_aAS \mid T_aA \mid a$
$B \to ST_bS \mid T_aAS \mid T_aA \mid a \mid T_bT_b \mid ST_b \mid T_bS$.
$T_a \to a$
$T_b \to b$.

$S \rightarrow AM_1 \mid AB$

$A \rightarrow T_aM_2 \mid T_aA \mid a$

$B \rightarrow SM_3 \mid T_aM_2 \mid T_aA \mid a \mid T_bT_b \mid ST_b \mid T_bS$.

$M_1 \rightarrow SB$

$M_2 \rightarrow AS$

$M_3 \rightarrow T_bS$

$T_a \rightarrow a$

$T_b \rightarrow b$

b) Show a regular grammar for the given language :{ $w \in \{a, b\}^*$: w does not end in aa}.

$S \rightarrow aA \mid bB \mid \varepsilon$
$A \rightarrow aC \mid bB \mid \varepsilon$
$B \rightarrow aA \mid bB \mid \varepsilon$
$C \rightarrow aC \mid bB$