Theory of Computation
Assignment 2

Name → Pulkit Jain
Roll no → 23/MC/112

Quest 1) a)
The powers of 'a' form an arithmatic progression with first term 2 and common difference 3
The set can be written as $L = \{a^{3k+2} / k \geq 0\}$
The regular expression is
$R.E = a^2(a^3)^*$ or $aa(aaa)^*$

b) $\{a^n / n$ is divisible by 2 or 3 or $n = 5\}$
This is a union of three conditions
→ n is multiple of 2: $(a^2)^*$
→ n is multiple of 3: $(a^3)^*$
$R.E = (a^2)^* + (a^3)^* + a^5$

c) This includes the string a itself, and all strings of length greater than or equal to 2 that start and end with 'a'; with any combination of 'a's and 'b's in between
$R.E = a + a(a+b)^*a$

Quest 2) $L = \{vwv, v, w \in \{a, b\}^*, |v| = 2\}$ is Regular
We'll construct R.E for it

i) The alphabet is $\Sigma = \{a, b\}$

ii) The string v must have a length of 2 ($|v| = 2$) The set of all possible strings for v is $\{aa, ab, ba, bb\}$

iii) The string w can be any string from $\{a, b\}^*$ The regular expression for w is $(a+b)^*$

iv) The language L is the set of strings vww, we can form the language by taking the union of possibilities for v.

→ v = aa, string of form → $aa\,(a+b)^*aa$

→ v = ab, string of form → $ab\,(a+b)^*ab$

→ v = ba, string of form → $ba\,(a+b)^*ba$

→ v = bb, string of form → $bb\,(a+b)^*bb$

Combining these

RE → $aa\,(a+b)^*aa + ab\,(a+b)^*ab + ba\,(a+b)^*ba + bb\,(a+b)^*bb$

∴ The language is Regular

Quest 3) The language L consists of all strings over $\Sigma = \{0, 1\}$, that do not contain the substring "00"

i) Any string in L can be viewed as blocks of 1s seperated by single 0s

ii) this means that every 0 must either be the last character of the string or be immediately followed by a 1

iii) Let's consider strings that do not end in 0. These strings can be constructed from the building blocks 1 and 01, (3)

This generates string like, 1, 11, 01, 111, 101, 0101 etc.

iv) Now consider strings that end in 0. A valid string can have a single 0 at the very end. For this to be valid. The preceding part of the string must not end in 0. The string generated by $(1+01)^*$ satisfy this

v) Therefore we can take any string from $(01+01)^*$ and optionally append a 0 at the end. This is represented by $(\lambda+0)$

Combining Them, RE: $(1+01)^*(\lambda+0)$

Quest 4) Let $L_1 = L((a^*ab+ba)^*a^*)$ and

$L_2 = L((a+ab+ba)^*)$, we prove $L_1 = L_2$

by showing $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$

part 1) proving $L_1 \subseteq L_2$

$a \in L_2$, since L_2 is closed under Kleene star,

$L(a^*) \subseteq L_2$

$ba \in L_2$,

$ab \in L_2$

Consider a^*ab, since $L(a^*) \subseteq L_2$ and $ab \in L_2$ then concatenation $L(a^*ab)$ is a subset of L_2

$L(a^*ab) \subseteq L_2$

$\therefore L(a^*ab + ba^*)^* \subseteq L_2$

finally, any string in L_2 is concatenation of a string $L(a^*ab + ba)^*$ and a string from $L(a^*)$ since both parts are subsets of L_2. this concatenation is also in L_2

$\therefore L_1 \subseteq L_2$

part 2) $L_2 \subseteq L_1$ ⇒

take Λ from $(a^*ab^* + ba)^*$ and 'a' from a^*

so, $a \in L_1$

take ab from a^*ab inside $(a^*ab + ba)^*$ and take Λ from the trailing a^* so $ab \in L_1$

take ba from $(a^*ab + ba)^*$ and take Λ from the trailing a^* so, $ba \in L_1$

Let $w \in L_2$,

→ if w has no 'b' then $w = a^n$ for $n \geq 0$

→ if w was a 'b' we can write $w = w'a^n$ where $n \geq 0$ and w' is with Λ or ends with 'b'

since $w' \in L_2$ and it ends in 'b' (or is Λ), it must be formed by a sequence of $\{a, ab, ba\}$

so any string $w \in L_2$ can be generated by the expression for L_1

$\therefore L_2 \subseteq L_1$

Thus $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$ we conclude that $L_1 = L_2$

Quest 5) Show that the given language is regular

→ We will generate regular expression that generates it,

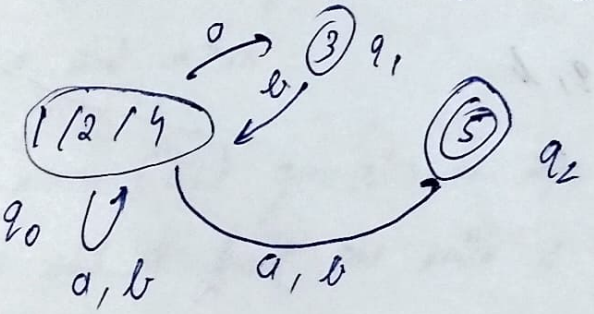
i) The language consists of string that must start with 'a' and end with 'a'

ii) The substring between the first and last 'a' is represented by w, where w can be any string over $\{a, b\}^*$

iii) By concating the parts we form the regular expression for L. the starting 'a' followed by the expression for w, followed by the ending a

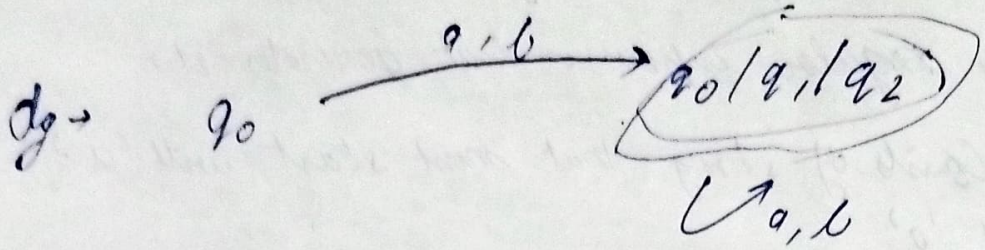
R.E → $a(a+b)^*a$ ∴ language is Regular

Quest 6) Combining nodes (1) and (2) and (4)

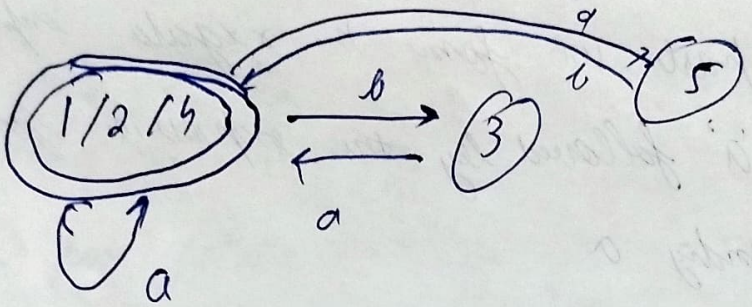


DFA	a	b
q_0	$q_0\ q_1\ q_2$	$q_0\ q_2\ q_1$
q_1	-	q_0
q_2	-	-

NDFA →		q	
	q_0	$q_0\ q_1\ q_1$	$q_0\ q_1\ q_2$
↓	$q_0\ a\ q_1\ q_2$	$q_0\ a\ q_1\ q_1$	$q_0\ a\ q_1\ q_2$



Ques 7) Combining ① & ② & ④



Ques 8) $q_1 = q_3 a + q_1 a + \Lambda$

$q_2 = q_2 b + q_3 b + q_1 b$

$q_3 = q_2 a$

$q_1 = q_1 a + q_2 aa$

$q_2 = q_2 b + q_2 ab + q_1 b$

$q_2 = q_1 b (b + ab)^*$

$q_1 = q_1 b (b + ab)^* aaa^*$

$q_1 = q_1 b (b + ab)^* aaa^*$

$q_1 = 1(b(b+ab)^* aa_* +)^+$

$q_1 = (b(b+ab)^* aa^+)^*$

Arden's Theorem:

Result in formal language theory that provides a method to solve linear equations involving regular expressions, specifically the equation

$$R = Q + RP$$

$$R = QP^*$$

Main usefullness: finite automaton into its corresponding regular expression

Proof q) a) The language is the union of two cases,

$n > m$ and $m > n$

Case 1) $n > m$ (L_1) generates an equal number of a's and b's then add at least one extra a at the beginning or in the middle

A grammar for this is

$S_1 \to aS_1 | A$

$A \to aAb | a$

Case $n_a > n_b$ (L_2) generates an equal number of a's and b's + add atleast one extra b at the end or in the mid.

A grammar for this is

$S_2 \to S_2 b / B$

$B \to a B b / b$

Combined grammar → $S_0 \to S_1 / S_2$

$S_1 \to a S_1 / A$

$S_2 \to S_2 b / B$

$A \to a A b / a$

$B \to a B b / b$

b) for every 'a' introduced, we must introduce between two and three 'b's. this can be captured by recursive calls that add one 'a' and either two or three 'b' in each step, base case is $n=0$

grammar →

$S \to a S bb / a S bbb / \Lambda$

c) The base string is 'a' when $n_a = 1$, $n_b = 0$, any other string can be generated by it

$n_a' = n_a + 2$, $n_b' = n_b + 1 \Rightarrow n_a + 2 = 2(n_b + 1)$

The two 'a's and one 'b' can be placed any where around a smaller valid string

grammar →

$S \to a$

$S \to aaSb / aSab / abSa$

$S \to SaaSb / aSbaS / aaSbS$

$S \to aSabS / abSa / baSa$

And all other formulations

now common & correct Grammar is → $S \to a / aSbSa / bSaSa$

Quest 10) a)

add a new start symbol :→

$S_0 \to S$

$S \to 1A / 0B$

$A \to 1AA / 0S / 0$

$B \to 0BB / 1S / 1$

Eliminate Null & unit productions →

there are no null productions. the unit production $S_0 \to S$ is resolved by $S_0 \to 1A / 0B$

Grammar is Now:

$S_0 \to 1A / 0B$

$S \to 1A / 0B$

$A \to 1AA / 0S / 0$

$B \to 0BB / 1S / 1$

Chomsky Normal form :→

Final CNF grammar

→ start symbol : S_0

→ non terminals : $\{S_0, S, A, B, X_0, X_1, A_1, B_1\}$

→ Terminals : $\{0, 1\}$

→ productions:

$S_0 \to X_1 A \mid X_0 B$

$S \to X_1 A \mid X_0 B$

$A \to X_1 A_1 \mid X_0 S \mid 0$

$B \to X_0 B_1 \mid X_1 S \mid 1$

$A_1 \to AA$

$B_1 \to BB$

$X_0 \to 0$

$X_1 \to 1$

6-10

(b) → Add a new starting symbol, $S_0 \to S$

→ Eliminate Unit productions

substitution → $S_0 \to a \mid b \mid cSS$

Grammar effectively becomes → $S_0 \to a \mid b \mid c S_0 S_0$

Convert to Chomsky Normal form

Start symbol : S_0

Non terminals : $\{S_0, C, S, S_1\}$

terminals : $\{a, b, c\}$

productions : $S_0 \to a \mid b \mid cS_1$

$C \to c$

$S_1 \to S_0 S_0$

Q 11) a)

Initial grammar Analysis: $S \to 5S$

$S \to 0S1$

$S \to 01$

Substitute to ensure all rules start with a terminal

$S \to 5S$ becomes

$S \to (0S1)S \Rightarrow S \to 0S1S$

$S \to (01)S \Rightarrow S \to 01S$

Set of rules for S is $S \to 0S1S \mid 01S \mid 0S1 \mid 01$

After removing terminals from the RHS (except for last one)

Final CNF grammar →

Non terminals: { S, X, }

terminals: { 0, 1 }

productions: $S \to 0SX_1S \mid 0X_1S \mid 0SX_1 \mid 0X_1$,

$X_1 \to 1$

b) → prepare the grammar

↳ a new non terminal X_b with

the production $X_b \to b$

→ the rule $B \to aAb$ becomes $B \to aAX_b$

Convert production to GNF

↳ production for B and X_b

↳ productions for A

↳ productions for S

Final CNF grammar

Non-terminals : { S, A, B, X_a, X_b }

terminals : { a, b }

Productions :

$S \to aAX_b \mid SBB \mid aSBB \mid aAX_bBB \mid aB$

$A \to aAX_b \mid aSB \mid aSB \mid aAX_bB \mid aB \mid b$

$B \to aAX_b \mid a$

$X_b \to b$

Quest 12) a)

grammar : $S \to a \mid abSb \mid aAb$,

$A \to bS \mid aAAb$

leftmost derivation 1 : $S \to abSb$

$S \Rightarrow abSb$

$\Rightarrow ab(a)b$ (using $S \to a$)

$\to ababb$

leftmost derivation 2 : $S \to aAb$

$S \Rightarrow aAb$

$\Rightarrow a(bS)b$

$\Rightarrow ab(a)b$

$\to ababb$

Grammar is ambiguous

grammar: $S \to aB \mid ab$, $A \to aAB \mid a$, $B \to ABb \mid b$

Grammar is ambiguous, consider the string ab, it has two distinct leftmost derivation

Leftmost derivation 1:

This derivation starts by using the production $S \to abSb$

$S \Rightarrow abSb$

$\Rightarrow ab(a)b$ (using $S \to a$)

$\Rightarrow abab$

Leftmost derivation 2:

This derivation

Leftmost derivation 1:

This derivation using the production $S \to ab$ directly.

$S \Rightarrow ab$

Leftmost derivation 2:

This derivation starts by using the production $S \to aB$

$S \Rightarrow aB$

$\Rightarrow a(b)$ (using $B \to b$)

$\Rightarrow ab$

∴ The grammar is ambiguous

Quest 13) a)

$L = \{ a^{n^2} \mid n \geq 1 \}$

we will prove by Contradiction

i) Assumption : Assume L is a context free language (CFL), by the pumping lemma for CFLs

ii) Choose a string : Let's choose the string $S = a^{h^2}$ since $h \geq 1$

we have $|S| = h^2 \geq h$ and $S \in L$

iii) decomposition → $|vwx| \leq h$

$|vx| \geq 1$

iv) pumping → $|S_2| = |uv^2wx^2y| = \{ |vwxy| + |vx| \leq h^2$

v) Contradiction →

From the lemma's condition we know

$1 \leq |vx| \leq |vwx| \leq h$

→ $h^2 + 1 \leq |S_2| \leq h^2 + h$

$(h+1)^2 = h^2 + 2h + 1$, since $h \geq 1$, we have $h^2 + h < 2h + 1$

$\therefore h^2 + h < h^2 + 2h + 1$

Combining → $h^2 < h^2 + 1 \leq |S_2| \leq h^2 + h < (h+1)^2$

The length of S_2 lies strictly between two consecutive perfect squares that $|S_2|$ can't be a perfect square

$s_2 \in L$ means

∴ contradicts the pumping lemma,

∴ our initial assumption was false, and L is not a context free language

6.13

b) Assumption :- Assume L is a CFL. Let h be its pumping length

Choose a string :- Let's choose $m = h$ and $n = 2h$. the string is

$s = a^h b^h c^{2h}$, $|s| = 4h \geq h$ and $s \in L$ because $m = h, n = 2h$ satisfies $h \leq 2h \leq 2h$

Decomposition :- $s = uvwxy$ where $|vwx| \leq h$ and $|vx| \geq 1$

The condition $|vwx| \leq h$ restrict the substring vwx to certain locations

Pumping and Contradiction :- Let's pump up by choosing $i = 2$, which gives the string $s_2 = uv^2wx^2y$

Case 1) vwx contains only a's then v and/or x must contain at least one a, the number of a's in s_2 will be greater than h, but the number of b's will still be h, so,

$n_a(s_2) > n_b(s_2)$ which violates the $a^m b^m$ condition

thus $s_2 \notin L$

Case 2) v can contain only b's

similarly $n_b(s_2) > h$ which $n_b(s_2) = h$, this violates the $a^m b^m$ condition thus $s_2 \in L$

Case 3) v can contains only c's let $|v| = k_2 > 1$ the string s_2 will be $a^h b^h c^{2h+k}$. let $m' = h$ and $n' = 2h + k$, for s_2 to be in L, we need $n' \geq 2m'$ this means $2h + k \geq 2h$, which is impossible since $k \geq 1$ thus $s_2 \notin L$

Case 4) v can contains @ a's and b's, v can contain only a's and x can contain only b's (or one of them contains both a and b's if it spans w, but in any case v and x cannot both be empty

Let $|v|_a = k_1$ & $|x|_b = k_2$ with $k_1 + k_2 \geq 1$

The string s_0 is $a^{h+k_1} b^{h+k_2} c^{2h}$.

for this to be in L, we must have $h + k_1 = h + k_2$ which means $k_1 = k_2$ if $k_1 \neq k_2$ the string is not in L,

if $k_1 = k_2$, the proof is more complex. Let's push down with $i = 0$, to $s_0 = uwy$

The string is $a^{h-k_1} b^{h-k_2} c^{2h}$, let's assume the decomposition forces $k_1 = k_2 = k \geq 1$ then

$s_0 = a^{h-k} b^{h-k} c^{2h}$, let $m' = h - k$ and $n' = 2h$

condition $n' \leq 2m'$ must hold this means

$2h \leq 2(h-k) \Rightarrow 2h \leq 2h - 2k$

which is false for $k \geq 1$; thus $s_0 \notin L$

Case 5) v can contains b's and c's

Let $|v|_b = k_1$, $|v|_c = k_2$ with $k_1 + k_2 \geq 1$. The string s_2 is $a^h b^{h+k_1} c^{2h+k_2}$. For this to be in L, the number of a's and b's must be equal, so $h = h + k_1$, which means $k_1 = 0$, this implies $k_2 \geq 1$. the string becomes $a^h b^h c^{2h+k_2}$

This is exactly case 3, which we already showed leads to a contradiction ($2h + k_2 \leq 2h$) thus $s_2 \notin L$

In all posible decompositions of s, we can find a value i for which the pumped string is not in L, this contradicts the pumping lemma

$\therefore$ L is not a context-free language