AUTOMATA Assignment - 1

202211501

1.

~~To prove~~

Given: (i) There is a Language L.
(ii) L has finite no. of strings

To Prove: L is a regular language.

Proof:

- ~~If~~ L is a finite language, i.e. it contains finite no. of strings. (say n)
- Let these strings be $l_0, l_1, \ldots l_{n-1}$
- The language ~~containing~~ $L_i = \{l_i\}$, s.t. l_i = a single literal string $\forall i = \{0, \ldots (n-1)\}$ ~~is~~ is regular.
- ~~The~~ We know that the union of a finite no. of regular languages is a regular language.
- Thus,

$$L = \{l_0\} \cup \{l_1\} \cup \{l_2\} \cup \ldots \cup \{l_{n-1}\}$$

$$L = \{l_0, l_1, \ldots l_{n-1}\}$$

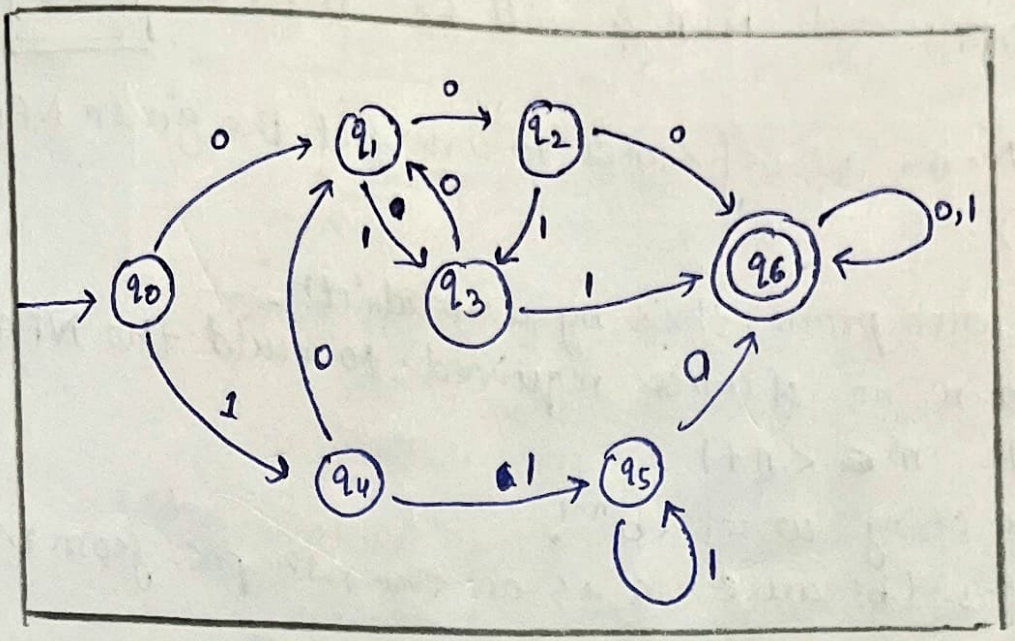
is regular.

Hence, proved.

2. Language $L = \{w \mid$ a length 3 substring of w is div. by $3.\}$
Alphabet $\Sigma = \{0, 1\}$

possible substrings $\equiv$ $\underbrace{(000)}_{0}, \underbrace{(011)}_{3}, \underbrace{(110)}_{6}$

Thus, our DFA should reach the final stage only in case one of these substrings are detected.



#	Binary Rep
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111
~~8~~	
~~9~~	

fig. DFA for the given language (M)

~~The above DFA can also be represented as:~~

~~$(Q, \Sigma, \delta, q_0, q_6)$~~

Thus, since there exists a finite automata (M) that recognizes L, thus, L is a regular language.

3. $\Sigma = (a, b)$ $L(M) = \{w \mid w$ has an 'a' at nth pos. from end$\}$

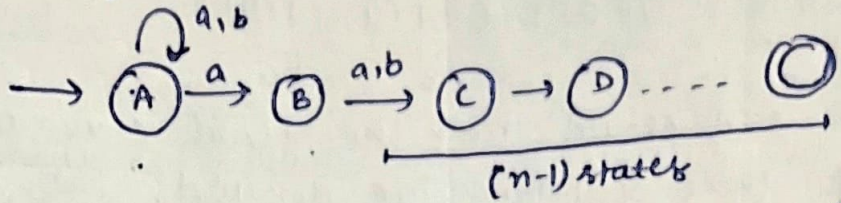
(a) The Reg. Expression for the language (L) would be :-

~~$(a+b)^* a (a+b)^{n-1}$~~

$$R = (a+b)^* . a (a+b)^{n-1}$$

(b)

NFA for L,



Thus, the minimum/number of states needed to make the required NFA will be $n-1+2 = \boxed{n+1}$.

To prove: Min. no. of states to construct the given NFA is $(n+1)$.

Proof: We will prove this by contradictn.

- Let the min. no. of states required to build the NFA be n' s.t. $n' < n+1$.
- Assume a string $w = ab^{n-1}$. Clearly, $w \in L$ (because a is on the nth pos. from the end).
- We start reading the stringth till we read it's length (n) completely.
- The first $(n-2)$ b's would lead to a state in the first $(n-1)$ states.
- We know that each transitn requires two states. Thus, for $ab^{n-2} \Rightarrow (n-1)$ transitns we require $(n-1+1)$ states $\Rightarrow n$ states. $(n'=n)$
- and thus, the $(n-1)^{th}$ b (last 'b') would read another state but our assumptn was that $n' \leq (n+1)$ causes this transitn to fail. Thus w won't be accepted by the NFA.
- Thus, our assumptn that $k < n+1$ is wrong as it doesn't satisfy the NFA.

Thus, minm no. of states required for the NFA is $n+1$.

4.

Let A be the set of english alphabets

$A = \{a, b, c, \ldots, z\}$

Thus,

the required Reg Ex is :

$$A^* (\{.\} + \epsilon) A^* \{@\} (\{research.\} + \{students.\} + \epsilon) (\{iiit.ac.in\})$$

5.

Task :- Converting Regular Expression into right linear grammar.

Algo. :- The following steps can be followed to convert a RegEx into a right linear grammar.

Note: f, g are terminals. , S = start state, RG = right gram

(a) Single Terminal : If the RegEx is simply f, we can write G, with only one production rule S → e (where S is the start symbol), is an equivalent R.G.

(b) Union operation : If the RegEx is of form f+g, we can write G with two rules of production, S → f | g, is an equivalent RG.

(c) Concatenation : If the RegEx is of form gf, we can write G with two rules of production, S → Ag, A → f, is an equivalent R.G.

(d) Star Closure : If the RegEx is of form g* (* is the Kleene star closure operation), the rules in G will be, S → gS | ε, is an equivalent R.G.
ε = epsilon.

(e) Star Closure on Concatenation : If the RegEx is of form (gf)*, G will have 3 rules of production, S → gA | ε, A → fS. is an equivalent R.G.

(f) Star closure on Union : If the RegEx is of the form (g+f)*, There will be three rules of production in G, S → gS | fS | ε, is an equivalent RG.

(g) Plus Closure : If the RegEx is of the form g+, we can write two production rules in G, S → gS | g, is an equivalent RG

(h) Plus closure on Union : If the RegEx is of form (g+f)+, then there are 4 production rules in G, S → gS | fS | g | f

(2) Plus closure on concatenath : Rules of producth Gr, S → εA, A → fS | f

Operath	Regular Expression	Right-Linear Grammar
Single Terminal	g	S → g
Union	g+f	S → g | f
Concatenath	g•f	S → gA, A → f
Star closure	g*	S → gS | ε
Star closure on Concatenath	(gf)*	S → gA | ε, A → fS
Star closure on Union	(g+f)*	S → gS | fS | ε
Plus Closure	g+	S → gS | g
Plus closure on union	(g+f)+	S → gS | fS | g | f
Plus closure on concatenation	(gf)+	S → gA, A → fS, f

6.

(a) $A = \{ bits(n) \mid n \text{ is a perfect square} \}$

$bits(n) =$ binary string rep. of no. n.

- Assume A is a regular language
- By pumping lemma, Let 'p' be the pumping length and let $w \in bits(n)$ i.e. $w \in A$ s.t. $|w| \geq p$; $|w| =$ perfect square
- Acc^{n} to pumping Lemma, w can be divided into 3 parts i.e. $w = xyz$ s.t.
  (a) $|xy| \leq p$ (b) $|y| \geq 1$ (c) $xy^iz \in A \quad \forall i \geq 0$
- $|w| = p^2$

Let $i = 2 \Rightarrow |xy^2z|$

6. (b)

$L = \{a^p \mid p \text{ is prime no.}\}$

Assumptn: L is a regular language

- If L is a regular language, then, it will satisfy the pumping lemma.

Let 'p' be the pumping length of the pumping lemma.

Let a string $S = a^p$ (s.t.) p is a prime no. & $|s| \geq p$ ← (1)

Thus, the pumping lemma is satisfied (from (1)) and S can be divided into 3 parts,

$$S = xyz$$

According to the Lemma,

(i) $|xy| \leq p$

(ii) $|y| \geq 1$

(iii) $xy^i z \in L \; \forall \; i \geq 0$

For S,

$x = a^\alpha \quad \alpha \geq 0$

$y = a^\beta \quad \beta \geq 1$

$p - \alpha - \beta$

$z = a$

6. (b) cont.

$\therefore xy^i z = a^{\alpha} a^{\beta i} a^{p-\alpha-\beta}$

$xy^i z = a^{p+(i-1)\beta}$

let $i = p+1$; $s' = a^{p+(1+p-1)\beta}$; $s' \in L$ from (c) of lemma.

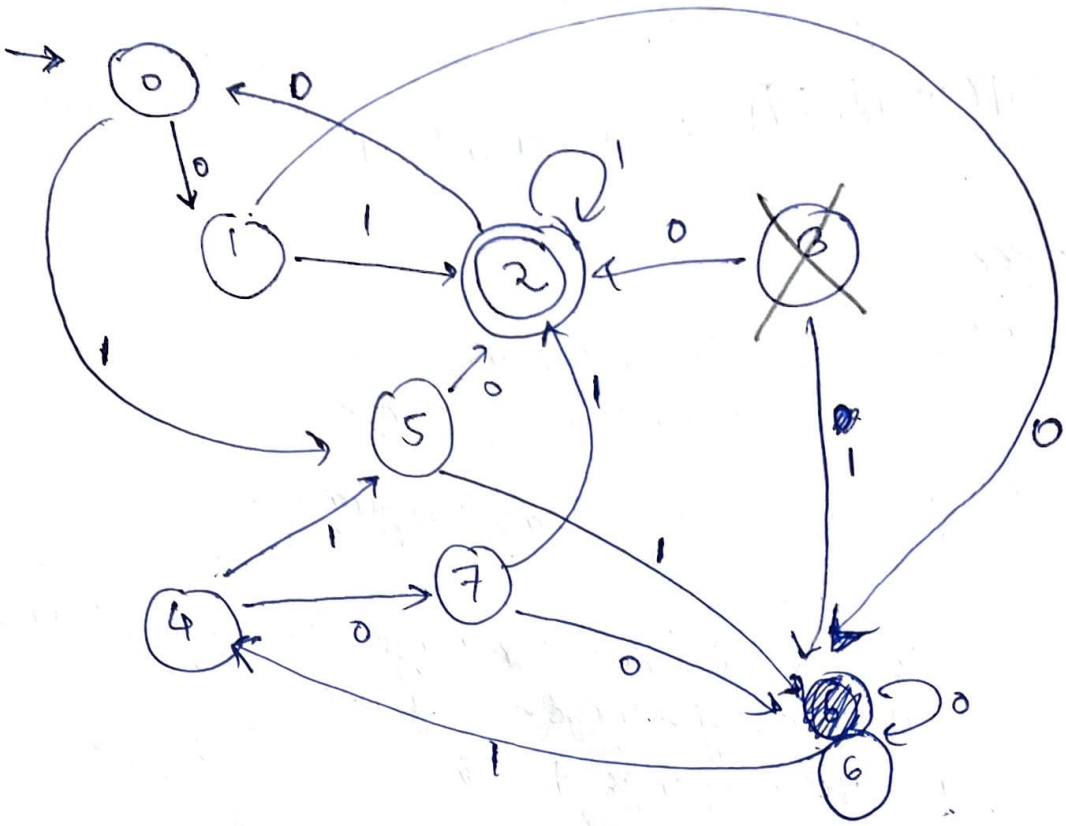
$s' = a^{p(1+\beta)}$

However, we can clearly see that $p(1+\beta)$ is not a prime no. (as $\beta \geq 1$). $\therefore s' \notin L$. This contradicts the lemma.

Thus, our assumption is wrong.

Thus, L is not a regular language.

7.



Step-1: As clearly seen, state 3 has no incoming arrows thus, it can never be reached. we will remove state 3.

Step-2:

{0, 1, 4, 5, 6, 7} — non-final states

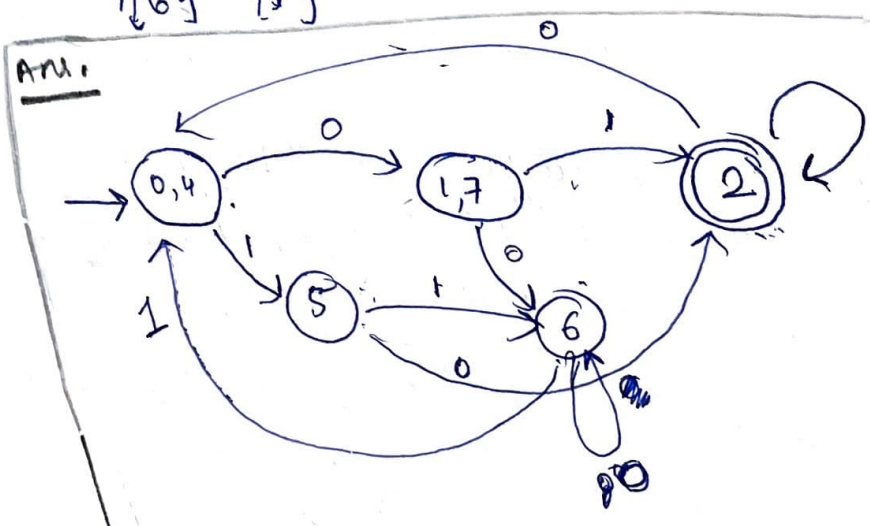
{2} — final states

	0	1
0	1	5
1	6	2
4	7	5
5	2	6
7	6	2
2	0	2
6	6	4

{0, 4} {1, 7} {2}

{6} {5}

Ans.

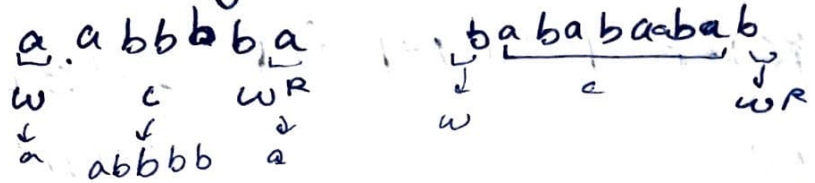


8. $a^+ = \{a, aa, \ldots\}$

Language(L) $= \{wcw^R \mid w \in \{a,b\}^+\}$

w^R = reverse of string w.

- Let string be,



As we can see, each time it is possible to select a c of maxm length s.t. the parts on it's right and left are w, w^R respectively.

Thus, to all such strings of form wcw^R will be accepted.

Thus, the Language L is regular.

The regular expression for the language can be given by,

$$(a(a+b)^*(a+b)(a+b)^*a) + (b(a+b)^*(a+b)(a+b)^*b)$$

OR $((a(a+b)^+a) + (b(a+b)^+b))$

9. (a) Grammar:

$G = S \to Sa \mid aS \mid a$

Counter example: "aaa"

Derivation for aaa :-

① $S \to Sa \to aSa \to aaa$

② $S \to aS \to aaS \to aaa$

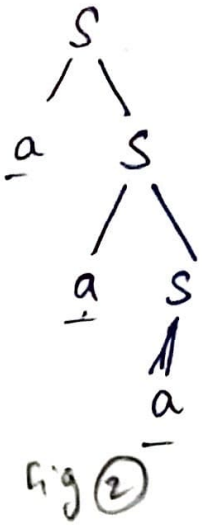


Fig ②

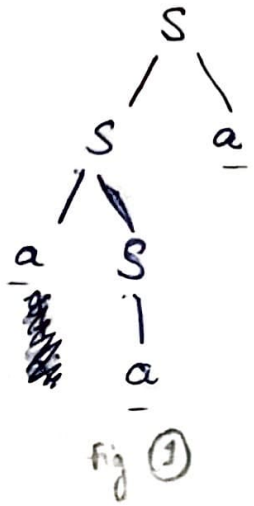


Fig ①

(b) Grammar:

$G = S \to SS \mid a \mid b$

Counter example : "aba"

Derivation for aba :-

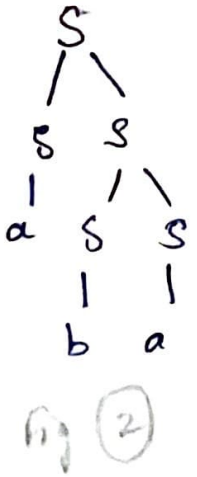


Fig ②

① $S \to SS \to Sa \to SSa \to aba$

② $S \to SS \to aS \to aSS \to aba$

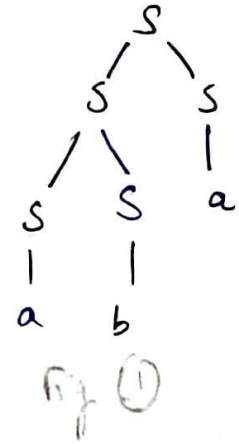


Fig ①

→

(c) Grammar:

$G = S \rightarrow S+S \mid S \times S \mid x$

Counter example: "$x+x \times x$"

Derivation for $x+x \times x$:

① $S \rightarrow S+S \rightarrow x+S \times S \rightarrow x+x \times x$

② $S \rightarrow S \times S \rightarrow S+S \times x \rightarrow x+x \times x$

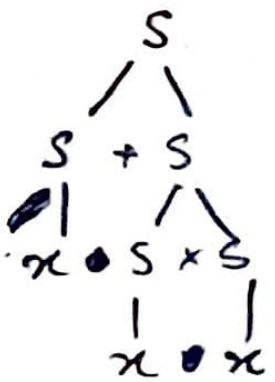


Fig. 1

Fig. 2

∵ for all the three parts (a), (b) (c) $\exists\ w \in L(G)$ s.t. there are two or more parse trees for w.

Thus, all the three grammars given above are ambiguo

10.

$$\Sigma = \{0, 1\}$$

$$L = \{0^n 1^m \mid n \leq m \leq 2n\}$$

(a) We know for a fact that the language, $(0^n 1^n)$ is not a regular language. Since this is ~~to~~ all strings of this form are contained in L. Thus, L is also not a regular language.

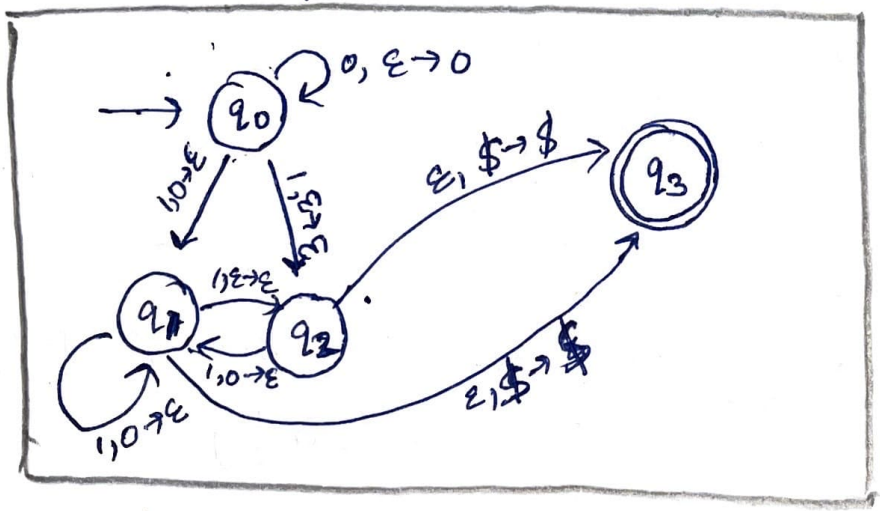
∴ L is a context free language.

(b) Grammar for L:

$$S \rightarrow 0S1 \mid 0S11 \mid \epsilon$$

(c) PDA (Push Down Automata):-

Assumpth:- stack is initially empty. Thus, $ is already pushed into the stack



$ = indicates an empty stack

where,

a, b→c indicates when a is read & b is popped, then c is pushed into the the stack.

11.

Given: Language $L = \{w \mid w \equiv 1 \bmod 2 \text{ and } w \equiv 2 \bmod 3\}$

$\Sigma = \{0, 1\}$

To prove: L is a regular language (draw a DFA).

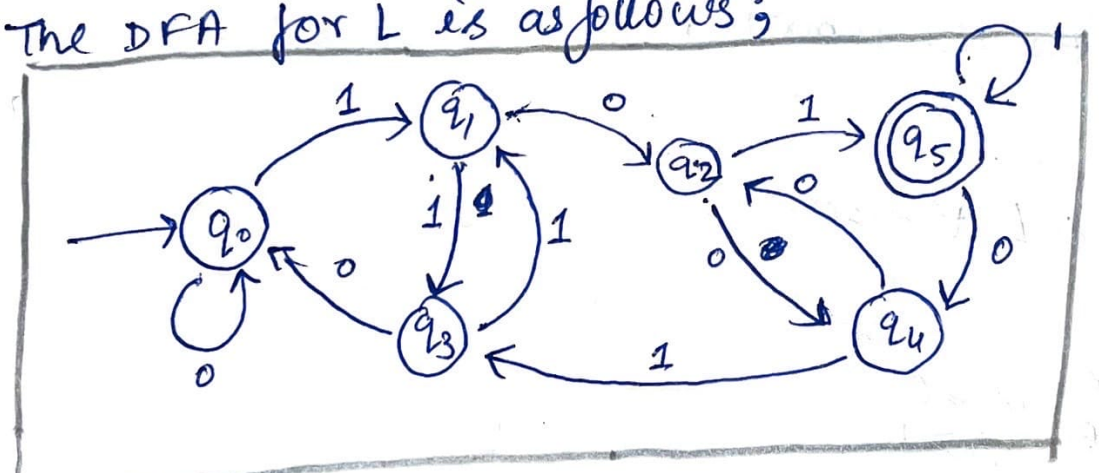
Proof:

As given, L contains those numbers which on being divided by 2 leave remainder 1 and on being divided by 3 leave remainder 2. Thus, this is equivalent to saying that it is a collection of those numbers that leave remainder 5 on being divided by 6.

i.e.,

$L = \{w \mid w \equiv 1 \bmod 2 \text{ and } w \equiv 2 \bmod 3\} \leftrightarrow L = \{w \mid w \equiv 5 \bmod 6\}$

The DFA for L is as follows;



where,
$q_r \Rightarrow$ remainder obtained is 'r' (for the part of the string read till the point)

As a DFA can be drawn which accepts w ($\forall w \in L$) thus, the language L is regular.

Derivation of the DFA: r = remainder

Let 'w' s.t. w mod 6 = 0 (q_0)

w 0 → 2×w . ⇒ r = 0 (q_0)

w 1 → 2w + 1 . ⇒ r = 1 (q_1)

w 10 → 2w(w₁) - ⇒ r = 2 (q_2)

⇒ 2(w) + 2

w 11 → 2w + 1 ⇒ r = 3 (q_3)

w 100 → 2(10) ⇒ r = 4 (q_4)

→ 100

w 101 → 2w + 1 ⇒ r = 5 (q_5)

→ shows that from q_2 if we read 1 we go to q_5 (as also shown in DFA)

Similarly, all the transitions of the DFA can be explained.