Attendance — 10%.

Assignment — 10%. (2)

Quiz — 20%. (4) → n-1

Mid — 25%.

Final — 35%.

LECTURE 1

MONDAY

DATE: 23/01/23

Letter/Symbol: Anything we can write down is a letter/symbol

e.g. a, অ, m, x, ক, #, ৮, ☺.

Alphabet: Finite set of letter/symbol.

e.g. {a, b, c, ... , x, y, z}

{অ, আ, ... , ঃ}

{0, 1} → Binary alphabet

{আ, m, #, ঃ, ক}

String: Finite sequence of letters

e.g. apple [apple is a string over English alphabet]

1011 [1011 is a string over Binary alphabet]

m#ঃ [m#ঃ is a string over some alphabet]

* There is a special string called empty string, which is denoted by ε. Length of ε is 0.

Language: Any set of string

e.g. {apple, ball, cat}

{0, 00, 1011, 1110}

* There is a special language called empty language which has no string { }

$\Sigma = \{0, 1\}$ L = A language set, which has string ending with 0

1100 ∈ L | ε ∉ L

1010 ∈ L

1011 ∉ L

Computational model

(1) Deterministic Finite Automaton (DFA)

L → a language L is fixed

L - Membership : a class of problem for every language.

Input : a string w

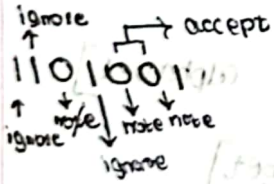
Output : Is $w \in L$?

$L_1 = \{w : w \text{ contains } 00\}$ * The language L_1 has string w with 00 {2 zeros side by side}

$100 \in L_1$

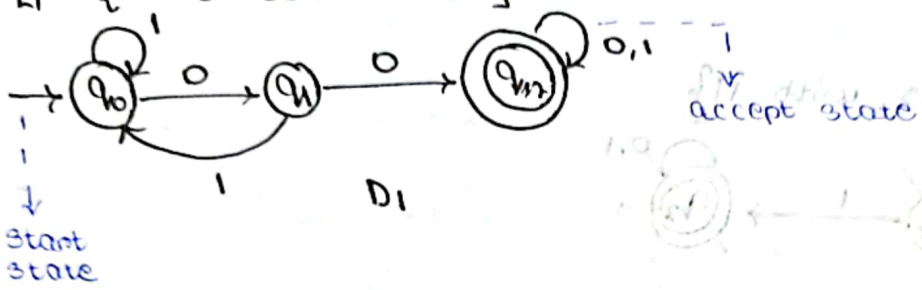
$001 \in L_1$

$101 \notin L_1$



DETERMINISTIC FINITE AUTOMATON

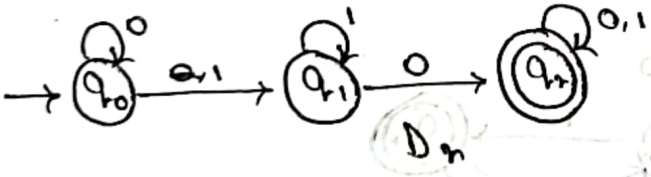
$L_1 = \{w : w \text{ contains } 00\}$



11101010011 → D_1 recognizes L_1

* When the entire input is read upto the accept state, then the output will be Yes, which means the input has the characteristics of L_1.

$L_2 = \{w : w \text{ contains } 10\}$



*** Regular Language: A special class of language for which DFA can be made for that language.
e.g. L_1 and L_2

Non-regular language: Language for which DFA cannot be made for that language.

LECTURE 2

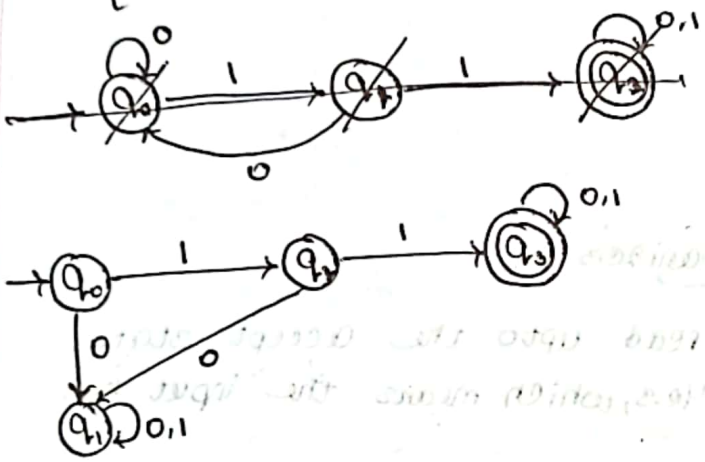
WEDNESDAY

DATE: 25/01/23

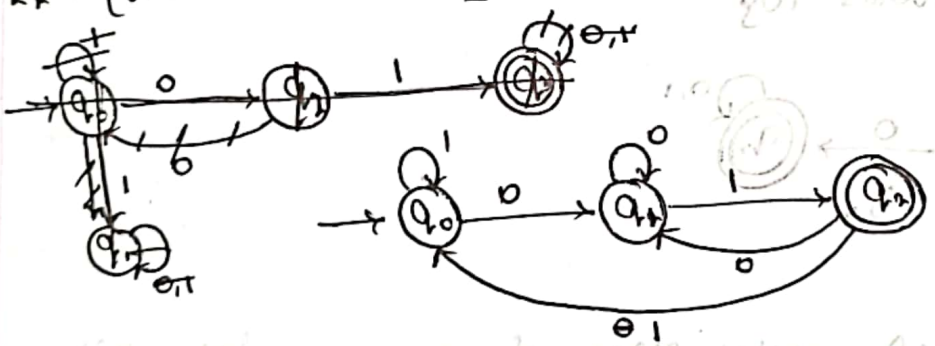
DFA

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

$L_1 = \{w : w \text{ starts with } 11\}$

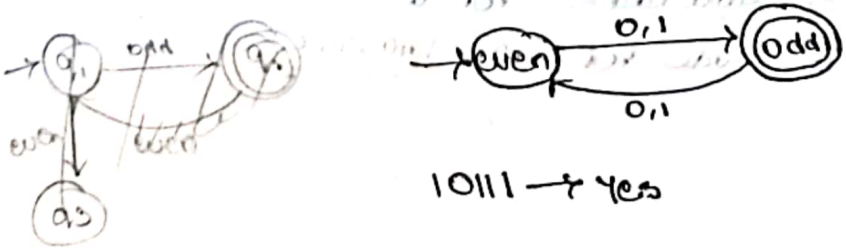


$L_2 = \{w : w \text{ ends in } 01\}$



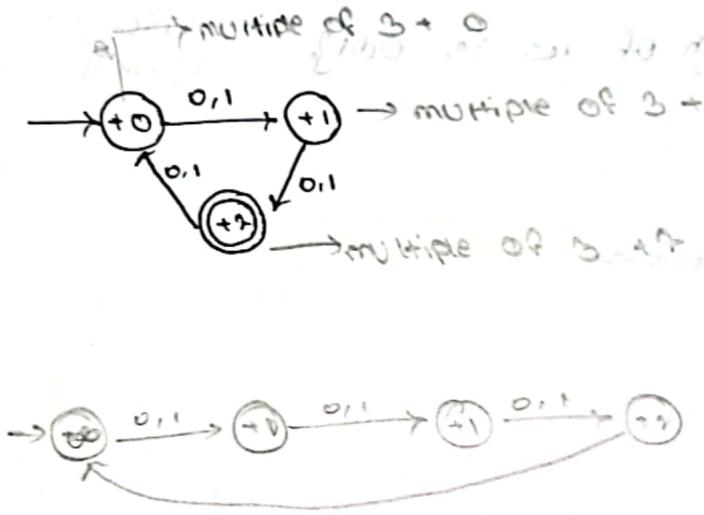
1100101011101 → Yes

$L_3 = \{w : w \text{ the length of } w \text{ is odd}\}$



10111 → Yes

$L_4 = \{w : \text{the length of } w \text{ is two more than a multiple of three}\}$



REGULAR OPERATIONS

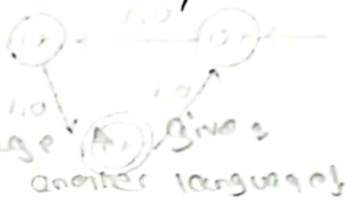
- Union {operates on 2 language & gives another language}

  e.g. $L_1 = \{w : w \text{ starts with } 11\}$

  $L_3 = \{w : \text{the length of } w \text{ is odd}\}$

  $L_1 \cup L_3$

→ Symbol of union: —
∪, ~~or~~ +



- Concatenation { operates on 2 language → concatenate

  $L_1 \quad L_2 \rightarrow L_1 \cdot L_2$

$L_1 = \{00, 101\}$

$L_2 = \{111, 1110, 1011\}$

$L_1 \cdot L_2 = \{00111, 001110, 001011, 101111, 1011110, 1011011\}$

- Kleene closure | star

$L = \{00, 101\}$

$L^* = \{\varepsilon, 00, 101, \text{(1 units)}, \text{(2 units)}$
$0000, 00101, 10100, 101101,$
$000000, 0000101, 0010100,$
$1010000, 10110100, 10100101,$
$00101101, 10110110101\}$
(3 units)

a^* → 0 or more

$a^+ = aa^*$ → 1 or more

Language ∅: language with empty string

= language

$L_1 \cdot \{\varepsilon\} = L_1$

$1 + 0 = 1$

$1 \cdot 1 = 1$

$\begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}$

* Union, concatenation and Kleene closure/star preserves regularity. → Removes DFA

If L_1 & L_3 for union are regular, $L_1 \cup L_3$ must be regular.

LECTURE 3

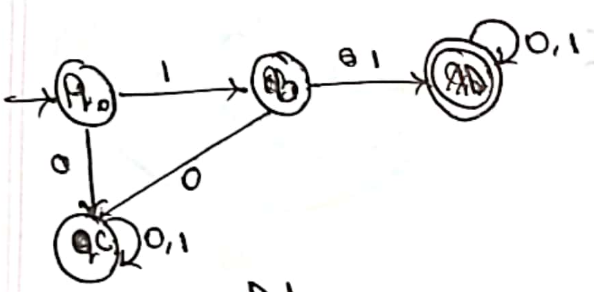
MONDAY DATE : 30/01/23

If L1 and L2 are regular, then so is L1 U L2

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

L1 = {w : w starts with 11}

L2 = {w : the length of w is odd}

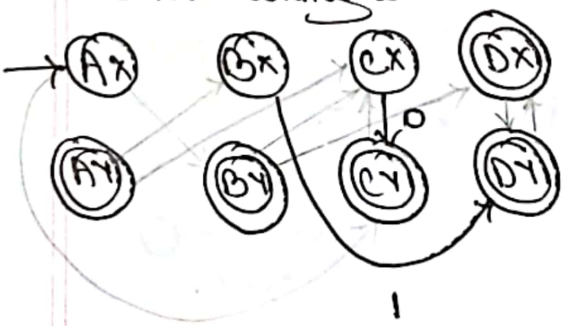


D1

D2

* for L1 U L2, either D1 or D2 or both needs to be accepted. to prove ~~L1 U L2~~ a string belongs to L1 U L2.

* Possible configurations



* 16 arrows

"Cross product construction" → Disadv.

Too many states, many arrows are needed.

~~L1 = 4 , L2 = 8 ,~~

~~48 states~~

L1 = 6 , L2 = 8 , L1 U L2 = 48

L2 is present, L1 is absent/not present

For L2 \ L1, only AY, BY, CY will be the accepting state

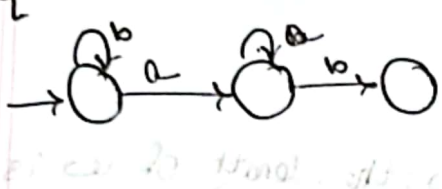
For L1 ∩ L2, only (DY) will be the accepting state.

For L1 \ L2, only (DX) will be

1.4, 1.5,

Ques

5(a) $\bar{L}^2 \{w : w \mid$ does not contain the string 'ab'$\}$



$L = \{w : w \mid$ does contain substring ab$\}$

top $\bar{L}$

L1 D1 ⟶ 10 state — 3 accepting state

L2 D2 ⟶ 6 states — 4 accepting state

L1 ∪ L2 (10+18) — (12) = 16

LECTURE 4

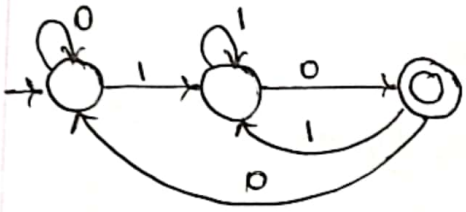
WEDNESDAY

DATE: 01/02/23

If L_1 and L_2 are regular, then so is $L_1 \cdot L_2$

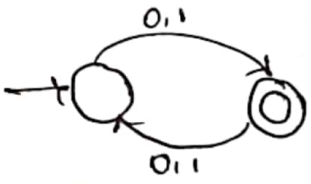
$\Sigma = \{0, 1\}$

$L_1 = \{w : w \text{ ends in } 10\}$



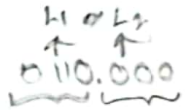
D1

$L_2 = \{w : \text{the length of } w \text{ is odd}\}$



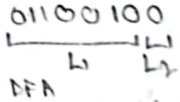
D2

* DFA cannot record this process.



$L_1 \cdot L_2$ ⇒

* we have to cut the string ~~into half~~ in such a way. LHS has L_1 and RHS has L_2.



* If by cutting, we are getting L_1 and not L_2. That does not mean that L_2 is actually absent. By cutting from different ~~side means~~ side, we can have L_2. But, the different possible outcomes cannot be find out using DFA: by rewinding. NFA can do it.

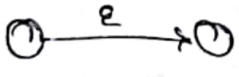
NONDETERMINISM

DFA

* Deterministic means a state has only one ~~to~~ arrow. for a particular state letter.

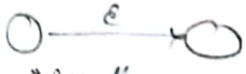
* NFA.

- can have more or fewer than one outgoing arrow for a letter from a state.
- has ε-transition

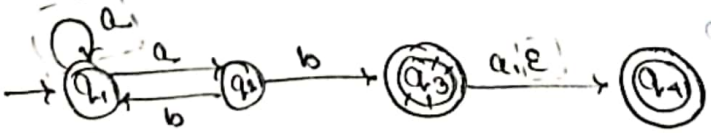


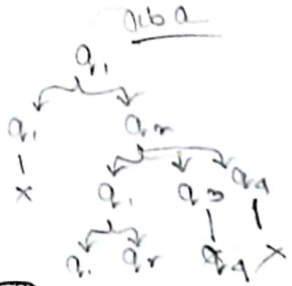
"free"

- ε-transition



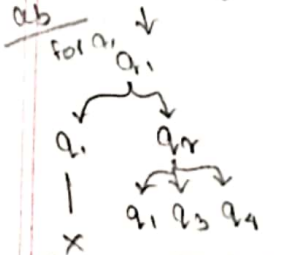
"free"



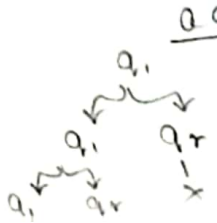


* Every DFA is a NFA, but not all DFAs are NFAs.

* NFA is a machine with a lucky charm.
- always guesses to choose the right choice



* there is any version present to accept the state for NFA.



*NFA is not more powerful than DFA. We can do the same work with DFA and NFA. But DFA needs to do it with more steps than NFA, but to do, the correct guess of NFA does not make DFA more powerful.

*NFA and DFA have equivalent power.

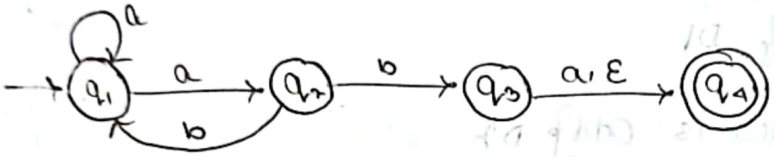
*Regular language: language which can make DFA/NFA.

LECTURE 5

MONDAY

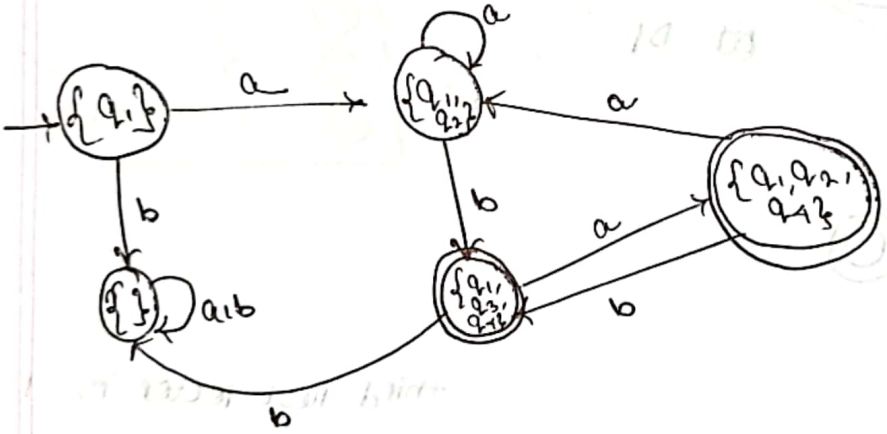
DATE: 06/02/23

Algorithm: NFA to DFA



* DFA keeps track of everything that happens unlike NFA who just guesses the correct one.
* DFA is not lucky as NFA.

Possible outcomes: 2^4
(Number of tracks)



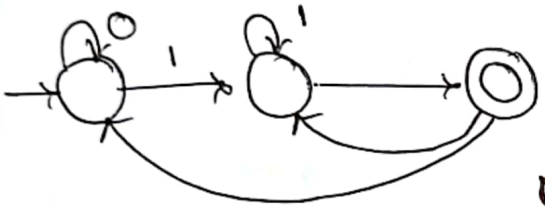
* Here, 16 possible configurations are not present because no more configurations are possible using this starting state, with any

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

$L_1 = \{w : w \text{ ends in } 10\}$ D1

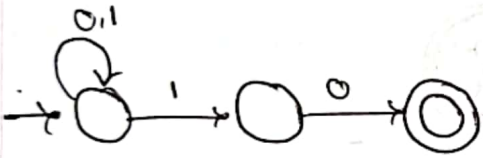
$L_2 = \{w : \text{the length of } w \text{ is odd}\}$ D2

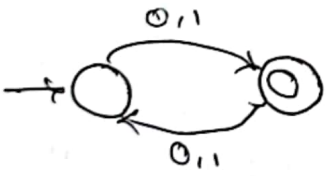
DFA



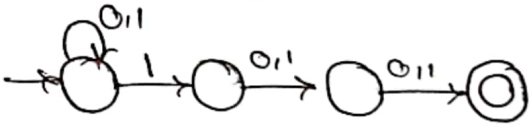
D1

NFA



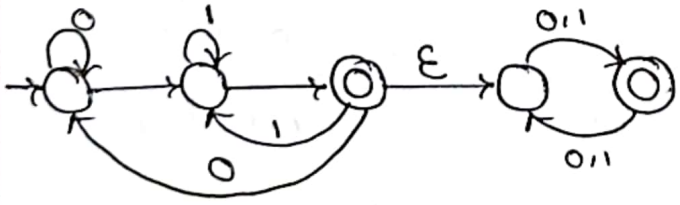


third last letter is 1



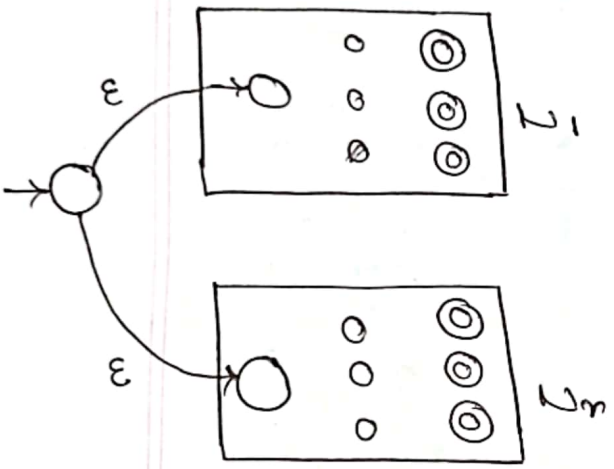
* Using DFA, this language needs almost 8 states
* NFA saves states.

NFA for $L_1 \cdot L_2$

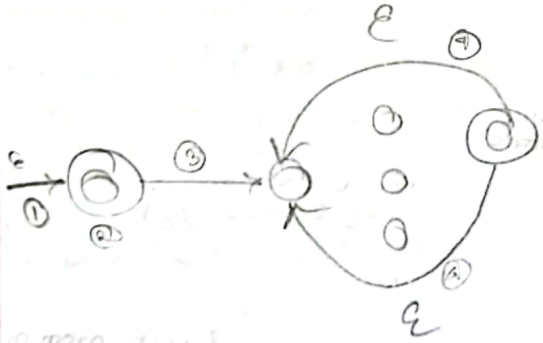


* Accepting state of D1 has an ε arrow connecting to the starting state of D2 [for $L_1 \cdot L_2$]

$L_1 \cup L_2$



L^*



*

(1) new start state is added.

(2) make the new start into accepting state.

(3) give arrow from new start state to the starting state

(4) draw ε arrow from the previous/ old accepting states to starting state

* If 2 integers are added we get an integer.
So, a set of integer is closed under addition.

* Set of regular language is closed under regular operati

LECTURE 6

WEDNESDAY

DATE: 08/02/23

REGULAR EXPRESSIONS

Arithmetic expressions are those expressions which are connected by arithmetic operations.

In regular expressions, every ~~alphabets of~~ letters in an alphabet including ε (empty string) and ϕ (empty set) ~~of~~ is the atom.

Regular operations are the operations of regular expressions.

Regular operations : $\cup$ (union)
$\cdot$ (concatenation)
$*$ (star)

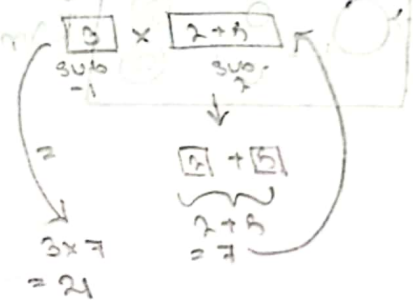
Arithmetic expressions

$3 \times (2+5)$

Every expressions is made of atoms and operations

Here, atoms are 3, 2, 5 and operations are $\times$, $+$

Every arithmetic expression evaluates to a number.



$(0 \cup 1)^* \cup 10$ → regular expression

$\cup$ → OR

* → zero or more / however many times

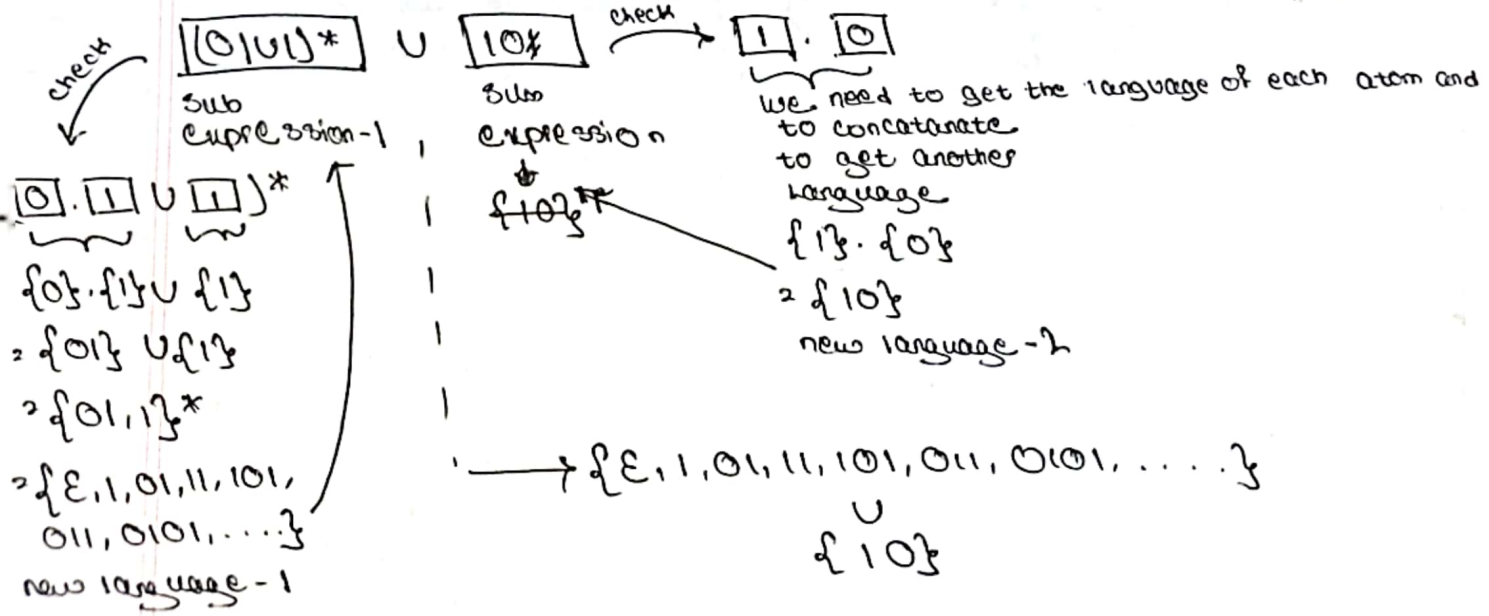
This regular expression is the union of two sub expressions.

$1(0 \cup 1)^* 1$

Read like:

1 → 0 or 1 → however many times → 1

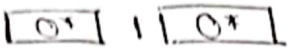
(1) (0 ∪ 1) *



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

string with exactly 1 language

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



0^*10^*

1] string with at least one 1

$(0 \cup 1)^*1(0 \cup 1)^*$

$(0 \cup 1)^*1(0 \cup 1)^*$ or $0^*1(0 \cup 1)^*$ or $(0 \cup 1)^*10^*$

0^*1^*0

- String such as 10001 won't be present in this language → Type-1 error
- ε will be present which is not supposed to be present → Type-2 error

$(0 \cup 1)^*$

- All strings out of the condition are present → Type-2 error

Type 1 error — we are not getting all strings needed for the condition in the expression.

Type 2 error — we are finding the language for the strings which does not fall under the language condition at all.

String with atmost one 1

$0^*(0 \cup 1)^* 0^*$

$0^* \cup 0^*10^*$ or $0^*(1 \cup \varepsilon)0^*$

$0^*1^*0^*$

- String with more than 1 are present — Type 1 error

0^*10

- 000 ... need not to be present — Type 2 error
- we are not getting all the strings with less than one 1 — Type 1 error

0^*10^*

- we will be getting ~~all the~~ exactly one 1 — Type 1 error

$0^* \cup 0^*10^*$ → correct ans

↓ ↓ ↓ ↓ ↓

no 1 no 1 one 1 no 1

$0^*(0 \cup 1)0^*$

- does not hit the empty string (ε) — Type 1 error

Q. String with same starting and ending letter

$(10 \cup 01)(0 \cup 1)^* (10 \cup 01)$

$(1 \cup 0)(0 \cup 1)^* (1 \cup 0)$

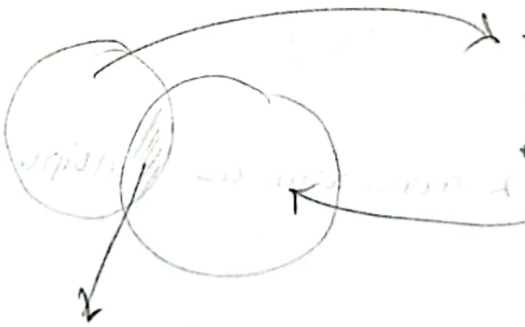
$(0 \cup 1)(0 \cup 1)^*(0 \cup 1)$

- There is no guarantee that first and end letter ~~0 or~~ will be same — Type 2 error

$1(0 \cup 1 \cup \varepsilon)^* 1$

- forced the language to start and end with 1, and 0 start and end with 0 is missing — Type 1 error

$1(0 \cup 1)^* 1 \cup 0(0 \cup 1)^* 0 \cup 0 \cup 1$ → correct



Think of all languages for which we can make DFA. (regular languages

Think of all languages for which we can make regular expressions

* DFA and regular expressions are equivalent in power.

- 2 Algorithms

i/p - regular expression
o/p - DFA

i/p - DFA
o/p - regular expression

LECTURE 7

MONDAY

DATE: 13/02/23

String with the second last letter is 1

$(0 \cup 1)^* 1 (0 \cup 1)$

String ~~with~~ having even length ~~string~~

$(00 \cup (01 \cup 11 \cup 00)^*$

$((0 \cup 1)(0 \cup 1))^*$

$(01 \cup 11 \cup 00 \cup 10)^*$

String having odd length

$((0 \cup 1) \cup (0 \cup 1)^*) \cup (1 (0 \cup 1)^* (0 \cup 1))$

$(1 (0 \cup 1)(0 \cup 1))^* (0 \cup 1)$

String having 2 more than a multiple of 3

0, 3, 11

$\underbrace{((0 \cup 1) \cup (0 \cup 1) \cup (0 \cup 1))^*}_{\text{multiple of 3}} \underbrace{(0 \cup 1)(0 \cup 1)}_{\text{2 more than}}$

- string without 11

$(00 \cup 10 \cup 01)^*$

$+00$

$+(00 \cup 01)^* (0 \cup 1$

$(0 \cup 10)^* (1 \cup \varepsilon)$

$(1 \cup \varepsilon)(00 \cup 01)^*$

- string without 000

$1^* (0 \cup \varepsilon)(0 \cup 1)(1 \cup \varepsilon)$

$(0 \cup 01 \cup 001)^* (0 \cup 00 \cup \varepsilon)$

- String with even number of 1's

001001 | 101 | 1000101

$((10 \cup 01 \cup 11)^* (10 \cup 01 \cup 11)^*)^*$

$((10 \cup 01 \cup 11)(10 \cup 01 \cup 11))^*$

$(0^* 10^* 1)^* 0^*$

- string with 1's count will be divided by 3 and remaining will be 2.

$(0^* 10^* 10^*)$

8 / 3 | 8 | 12 | 6 / 2

0001001010 | 0100

101000

w has 1s in positions 1, 4, 7 and length that is two more than a multiple of 3.

(0U1)1(0U1)(0U1)(0U1)(ε 5, 8

~~(0U1)1~~

(1(0U1)(0U1)1(0U1)(0U1)1)*

(1(0U1)(0U1)1(0U1)(0U1)1)

(1(0U1)(0U1))*1(0U1) ✓

w does not start with 01 and has a length that is multiple of 3.

~~(1(0U11U0)(0U1)(0U1))*~~

(00U10U11)(0U1)(0U1)(0U1)(0U1))* U ε ✓

~~(1(0U11U0)(0U1)(0U1)*~~

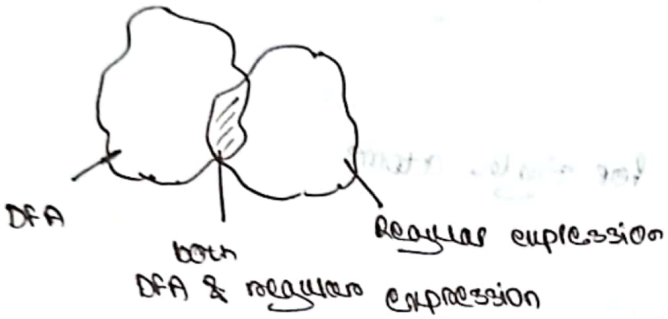
LECTURE 8

MONDAY

DATE: 20/02/23

* DFA is like a judge/recognizer.

* Regular expression is like generator.



Regular Language: ~~Language~~
For a given language, regular expression can be made.

RE → NFA → DFA ..... algorithm (i)
i/p → o/p

if the above algorithm is TRUE, (i)
then

DFA → RE ..... algorithm (ii)

if the above algorithm is TRUE, (ii)
then

empty
(no regular expression)

THUS, proves

RE → NFA → DFA ..... algorithm (i)

(ab ∪ a)*

↓

ab ∪ a

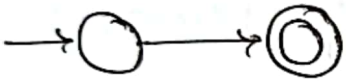
↓

a.b

*We have to find NFA for single atoms

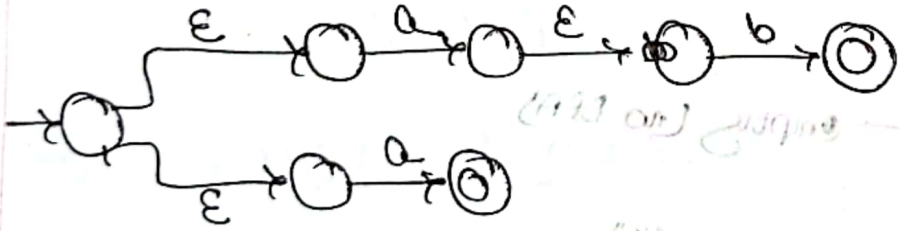
For a.b

a 

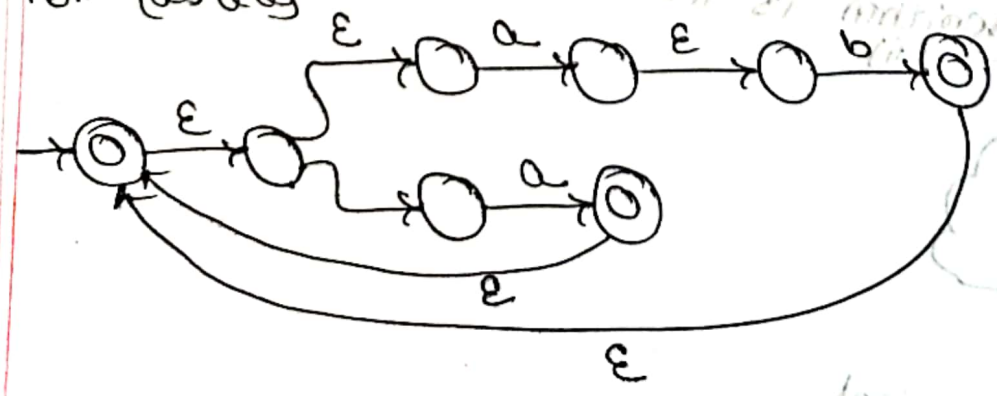
b 

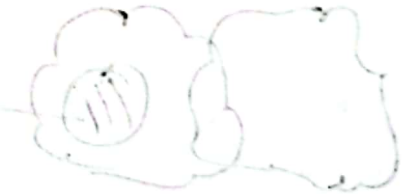
ab

For ab ∪ a



For (ab ∪ a)*





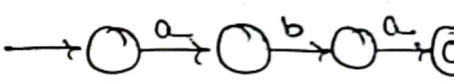
$(a \cup b)^* aba$

↓

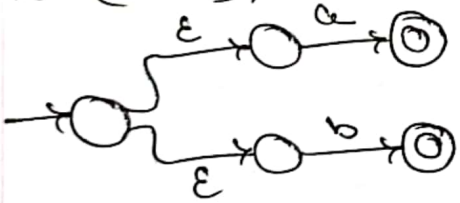
$(a \cup b)aba$

For a.b.a.,

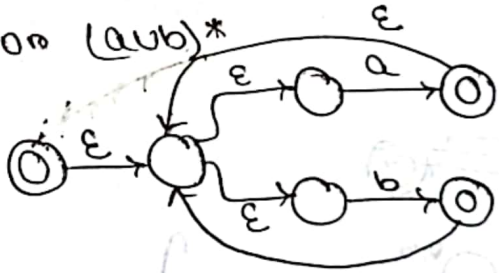
or



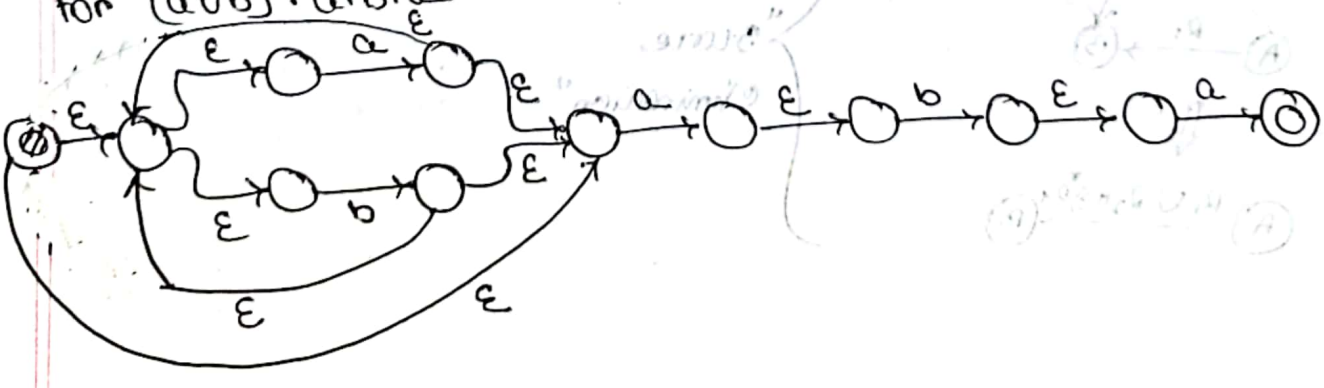
For $(a \cup b)$,



For $(a \cup b)^*$



For $(a \cup b)^* . a.b.a$



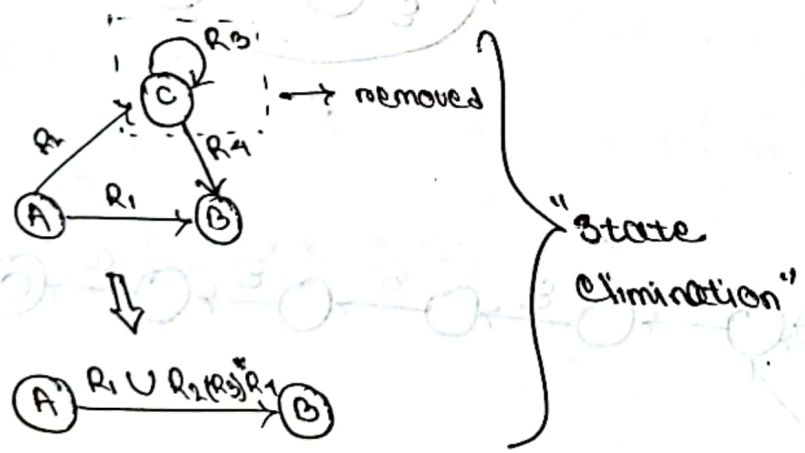
**** DFA → RE ..... algorithm (ii) [state elimination]

(1) i/p → DFA

(2) Pre-processed DFA:

Now, a state is taken and thrown. Damage control is applied to preserve the path of the thrown state to avoid changing the language.

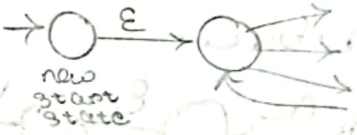
This is continued upto a single arrow is present from starting and accepting state with the regular expression on the arrow.



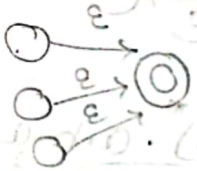
"State Elimination"

PRE-PROCESSING STEPS

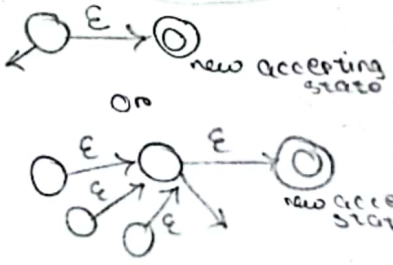
(1) If DFA has incoming arrow in start state, we have to fix it to remove the incoming arrow ~~without~~ by giving a new start state ~~and~~ with an ε arrow to the previous starting state.

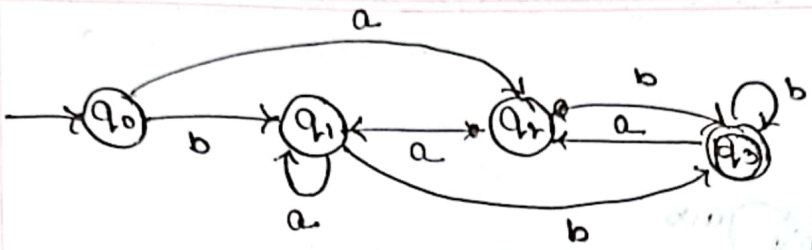


(2) Multiple accepting states is not allowed. A new accepting state is taken with ε arrow from previous accepting state ~~is~~ ε

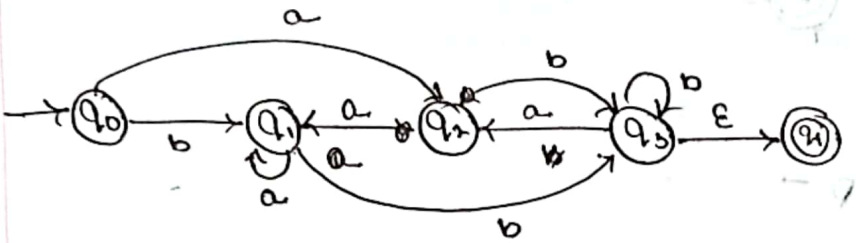


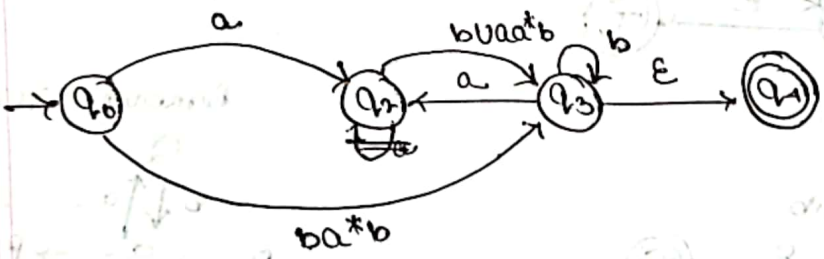
(3) Accepting state should not have outgoing arrow. A new accepting state is added with an ε arrow from the previous accepting state.

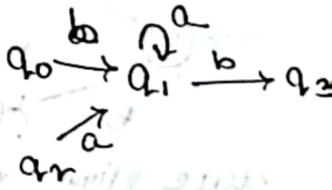


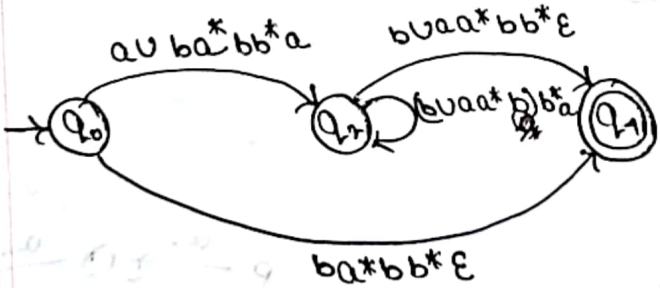


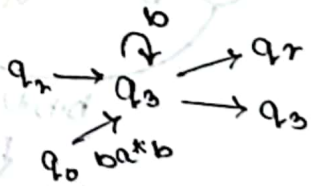
pre-processed

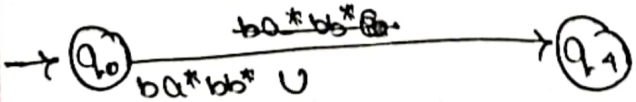








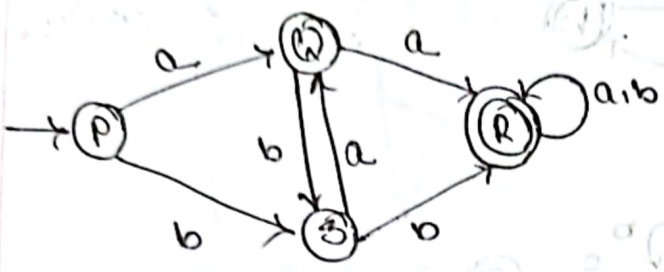




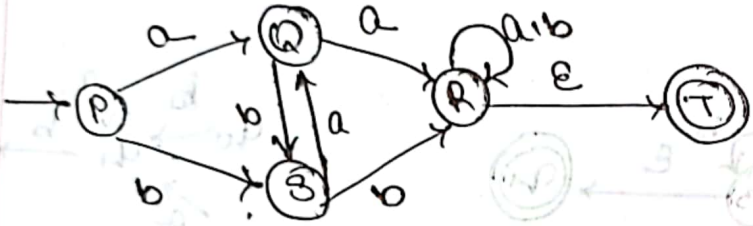
(aUba*bb*a)(bUaa*b)b*a)* (bUaa*bb*ε)

HOMEWORK

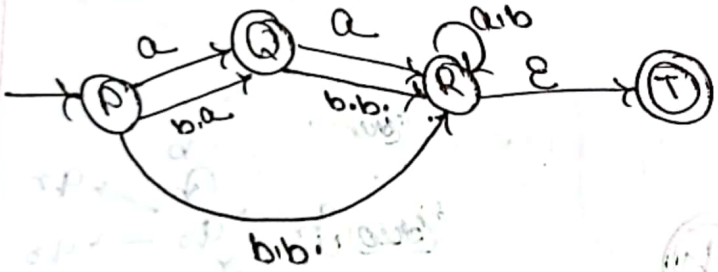
DFA → RE

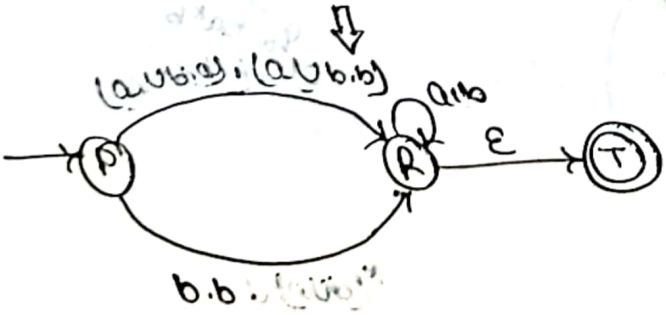


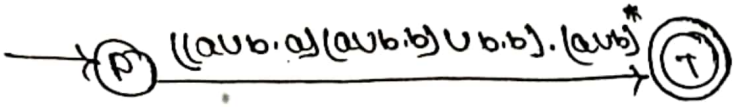
pre-processing step:-



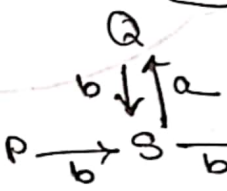
state elimination:-

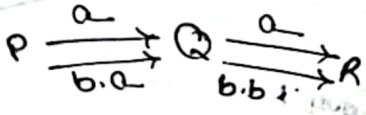


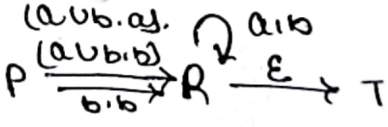




Removing state







LECTURE 9

WEDNESDAY

DATE: 22/02/23

NONREGULAR LANGUAGES

To prove a theorem, we need to prove some sub theorems which are called lemma.

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

$L = \{w : w = 0^n 1^n ; n > 1\}$

$0011 \rightarrow n = 2$

$000111 \rightarrow n = 3$

$0101 \rightarrow$ does not follow the pattern $0^n 1^n$

* If Every languages has some property, and if some language does not have this property, then that language is not regular.

DFA for length = 6

$q_1\ q_2\ q_3\ q_4\ q_5\ q_6$

There is a repeat of state in the first 6 letters of the string.

$0010111010 \ldots\ldots$

$00101101 \ldots\ldots$

$q_3 q_6 q_1 q_4 q_2 q_3 q_1 \ldots\ldots q_6$

* If you take a sufficiently large ~~amount of~~ string, there will be repetition of state and that string will be in the language.

Pumping lemma: 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 three pieces, $s = xyz$; satisfying the following conditions:

1 for each $i \geq 0$, $xy^iz \in A$

2 $|y| > 0$ and $y \neq \varepsilon$

3 $|xy| \leq p$

$|s|$ → length of s

→ $|y| \neq 0$, the middle part must not be empty.

→ $|xy| \leq p$, the length of first & semiddle part has to be less than pumping number

→ $xy^iz \in A$, if y is taken as many times/repeated as needed, yet it lies within the language.

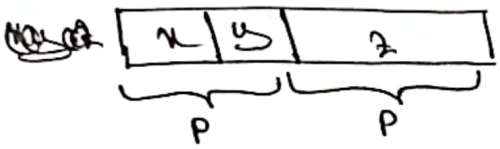
$xy^2z \rightarrow xyyz$

$L = \{w : w = 0^n 1^n : n \geq 1\}$

Let p be the pumping length.

$0^p 1^p$

length $= 2p$

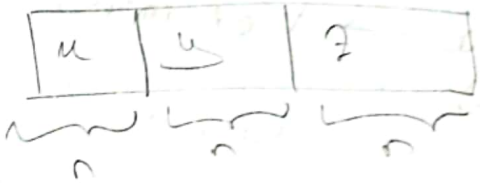


y is entirely made of 0.

If 0 is repeated once, then it crosses p. Thus, $0^p 1^p$ is contradicted.

$0^n 1^n 2^n$

length $= 2n$



$|y| \geq 1, i \geq 0$

$|y| > 0$

$D = \{1^{n^2} \mid n \geq 0\}$

$n = 0, 0^{0^2} = 1^0 = \varepsilon$

$n = 1, 1^{1^2} = 1$

$n = 2, 1^{2^2} = 1111$

$n = 3, 1^{3^2} = 111111111$

$p^2 < p^2 + p < p^2 + 2p + 1$

$= (p+1)^2$

LECTURE 10

even number of 1's

MONDAY

DATE: 27/02/23

REVIEW CLASS

∪ . *

∪ / + ⟹ concatenation

a^* → 0 or more

(Kleene closure)

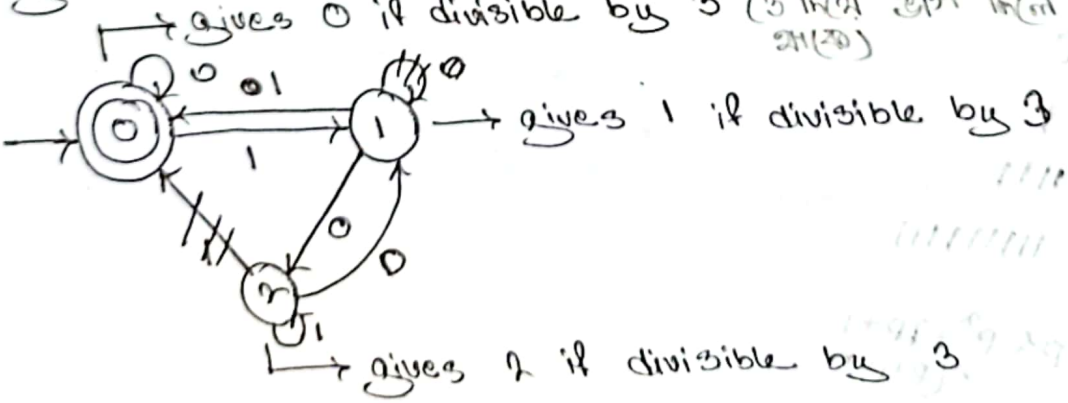
$a^+ = a^* a$ → 1 or more

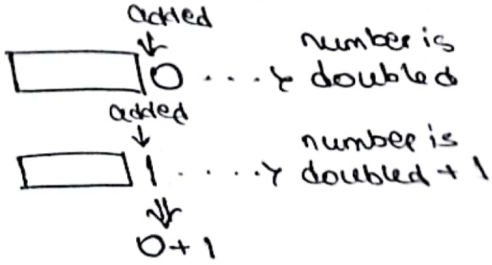
(+ve closure)

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

$L = \{w : w$, when interpreted as a binary number, is divisible by 3$\}$

00110 is 6 in binary number. Thus, it is divisible by 3.





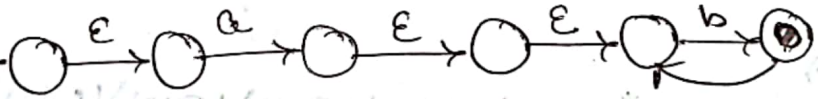
$a^* b^*$

[At first, there are some a's, and then there are some b's. After, b there must not be any a]

$a^* \Rightarrow$

$b^* \Rightarrow$

$a^* b^*$



FALL 2022 [SET - I]

(2) (a) $(0 \cup 1)(0 \cup 1)(0 \cup 1)^*$

~~(b) $(00 \cup 10)^*$~~

(b) $(0 \cup 1)0)^* (0 \cup 1 \cup \varepsilon)$

(c) (i) $\overline{L_1 \cap L_2}$ $(0 \cup 1)(0 \cup 1)(0 \cup 1)^* (0 \cup 1) \cup (0 \cup 1)(0 \cup 1)(0 \cup 1)^* (0 \cup 1)(0 \cup 1)$

~~(ii) $\overline{L_2} = (1(0 \cup 1)^* (0 \cup 1 \cup \varepsilon)) \cup (1 \cup \varepsilon)$~~

(ii) $\overline{L_2} = (0 \cup 1)(0 \cup 1)^* (0 \cup 1 \cup \varepsilon)$

(iii) $\overline{L_1} \cup \overline{L_2}$

$(0 \cup 1)(0 \cup 1)(0 \cup 1))^* (0 \cup 1) \cup (0 \cup 1)(0 \cup 1)(0 \cup 1))^* (0 \cup 1)(0 \cup 1)$

$\cup$

$(0 \cup 1)^* ((0 \cup 1)(0 \cup 1))^* (0 \cup 1 \cup \varepsilon)$

L_1 = length at most 3

$\overline{L_1}$ = length at least 4, L_3 = length at least 2

$\overline{L_1} \cap L_3 \Rightarrow$ a string of length of 4.

number of states in $L_2 \cap L_3$ = number of states in L_2 × number of states in L_3

LECTURE 11

WEDNESDAY

DATE: 15/03/23

SET-2

L = { w : 00 appears at the end of w but nowhere else }

~~(01 ∪ 11)* 00~~

~~(1 ∪ 01)*~~

(1 ∪ 10) 01)* 00

L = { w : 10 appears an even number of times in w }

~~0*1* 10~~ ~~(10)* 10 (10)* 10)*~~

(0*1*(10)10 0*1*10)* 0*1*

→ 10 NIL

11

0*1*

every 0 must appear by every 1

L = { w : w = 0^m 1^n 0^k, where m, n, k ≥ 0 and m + n + k is e

(00)* (10)* (00)* ∪ (00)* 1(11)* 0(00)*

∪ 0(00)*1(11)* ∪ 0(00)* (11)* 0 (00)*

(00)*

Set 2

(2) (a) $(1 \cup 01)^* (0 \cup \varepsilon)$

$(0 \cup \varepsilon)(1 \cup 10)^*$ — L1

(b) 0 111 0 1 0 1 0 1

(c) $0(1 \cup 10)^*$

(d) $(1^*01^*0)^*1^*$ — L3

(e) $(1^*01^*0)^*1^*01^*$ — count of 0 is odd

$(1^*101^*10)^*1^*101^*$

(1) (a)

(b)

(c) $3 \times 3 = 9$

(d) 1000 : 0000

(e)

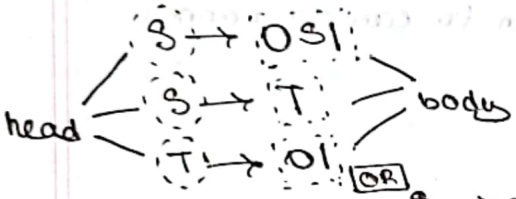
FINAL

LECTURE 12

WEDNESDAY DATE: 22/03/23

CONTEXT-FREE GRAMMAR (CFG)

* CFG is like DFA, NFA & regular expression, but is more powerful.



$S \rightarrow OS1 \mid T$

$T \rightarrow 01$

Substitution rule / Production : All the lines of CFG

Variable / non-terminal : the variable on the head of the production. {S, T}

Terminal : All the signs except variable in the body. {0, 1}

Start variable : The variable on the ~~first~~ body of the 1st production.

* We have to start variable and apply production until all the variables are removed and terminals are remaining.

Start

$S \rightarrow OS1 \rightarrow O\,OS1\,1 \rightarrow O\,O\,O11\,1$

- S → OS1 : variable, so replace with other head (S) e.g.
- O S 1 : variable, so replace with T
- O O O 1 1 1 : string is "derived"

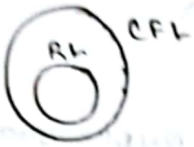
$L : \{w : 0^n 1^n, n \geq 1\}$

Start

$S \rightarrow OS1 \rightarrow O\,T\,1 \rightarrow 00\,11$

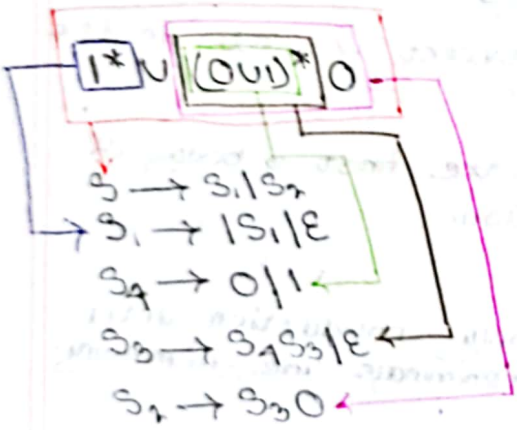
If all the strings are derived using CFG, then the set of ~~language~~ string received is the context free language (CFL).

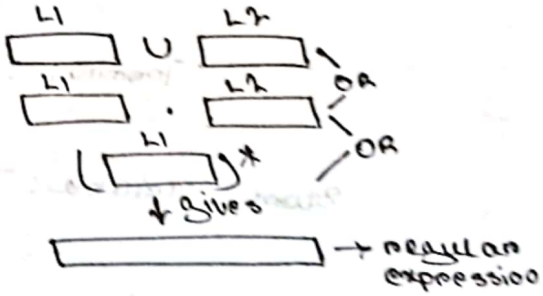
CFL is the language where CFG can be applied.



* If a language is regular, then it is bound to be context free.
* A language which is context free, then it can or cannot be regular.

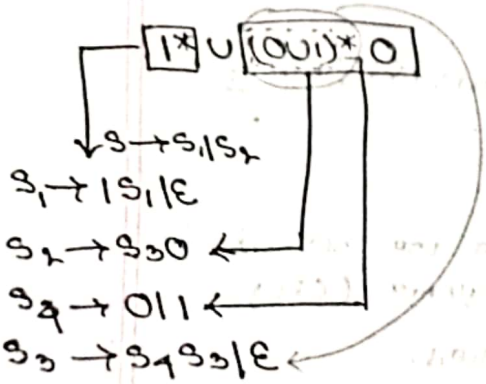
ALGORITHM : REGULAR LANGUAGE TO CONTEXT FREE GRAMMAR





* A regular expression can be obtained from star, union & contation.

* for union, (U, |)
  S_1 & S_2 are two CFG.
  $S \rightarrow S_1 | S_2$
* for concatanation, (.)
  $S \rightarrow S_1 S_2$
* for star, (*)
  $S \rightarrow S_1 S_2 | \varepsilon$

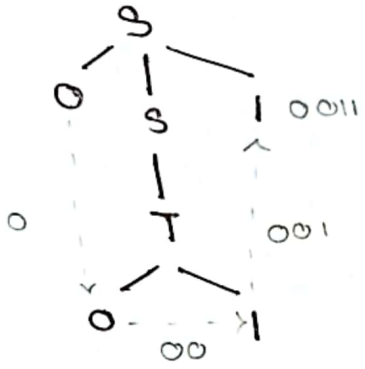


PARSE TREE

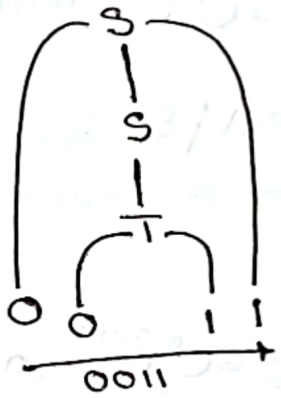
$S \rightarrow 0S1$

$S \rightarrow T$

$T \rightarrow 01$



OR



pictorial figure for CFG (parse trees)

* If we taken all the leaf o from the LHS and then take all the leaf from the RHS, then we get the string derived.
∴ derived string = "0011"

even number of 0s

$S \rightarrow 10S|0S|1S|10|00$

~~$I \rightarrow 10|00$~~

$L = \{w : w = 0^n 1^n ; n > 0\}$

$S \rightarrow 0S1 \mid \varepsilon$

① $0S1 \rightarrow 0\varepsilon1$

② $0S1 \rightarrow 00S11$ $[n = 2]$

$L = \{w : w = 0^n 1^{2n} ; n > 0\}$

$S \rightarrow 0S11 \mid \varepsilon$

① $0S11 \rightarrow 00S1111$ $[n = 2]$

$L = \{w : w = 0^n 1^{3n+2} ; n > 0\}$

$0^n 1^{3n} \cdot 1^2$

$S \rightarrow 0S111 \mid 11$

① $0S111 \rightarrow 011111$ $[n = 1]$

③ $0S111 \rightarrow 00S111111$

0011111111 $[n = 2]$

LECTURE 13

MONDAY

DATE: 27/03/23

$S \rightarrow X1X$ → 2 variables (S, X)

$X \rightarrow 0X|1X|\varepsilon$ → 4 productions

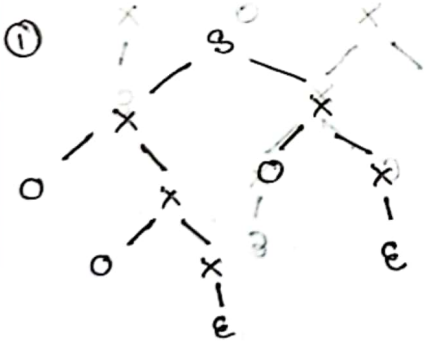
Derive 0010:

(i) $S \Rightarrow X1X \Rightarrow 0X1X \Rightarrow 00X1X \Rightarrow 00\varepsilon1X \Rightarrow 00\varepsilon10X \Rightarrow 00\varepsilon10\varepsilon \Rightarrow 0010$

(ii) $S \Rightarrow X1X \Rightarrow 0X1X \Rightarrow 0X10X \Rightarrow 0X10 \Rightarrow 00X10 \Rightarrow 0010$

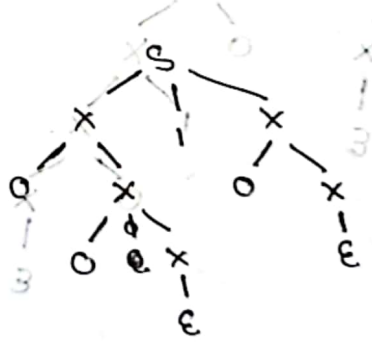
Leftmost derivation: If we have multiple variables, we start replacing the variables from the left.

(i)



* The tree is formed from the LHS & then from the RHS

(ii)



* The tree is formed randomly.

Ambiguous grammar: A string will be present which can have either multiple different leftmost derivations or multiple different parse tree.

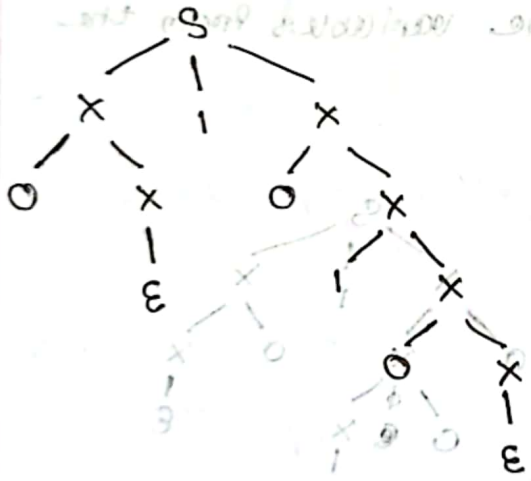
01010

(i) $S \Rightarrow X1X \Rightarrow 0X1X \Rightarrow 01X \Rightarrow 010X \Rightarrow 0101X \Rightarrow 01010X \Rightarrow 01010$

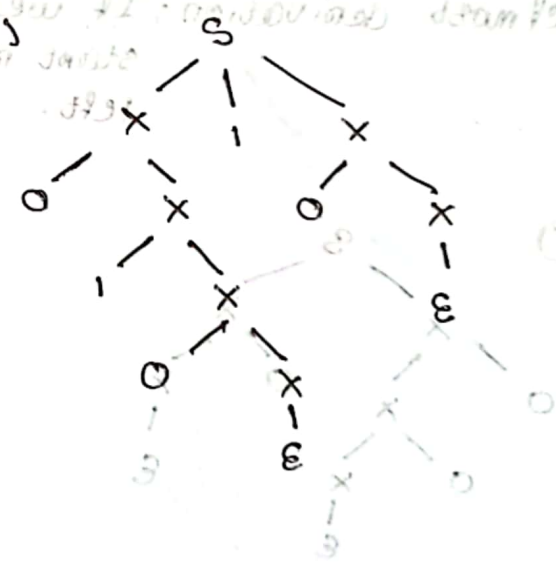
(ii) $S \Rightarrow X1X \Rightarrow 0X1X \Rightarrow 010X1X \Rightarrow 0101X \Rightarrow 01010X \Rightarrow 01010$

Grammar is ambiguous grammar

(i)



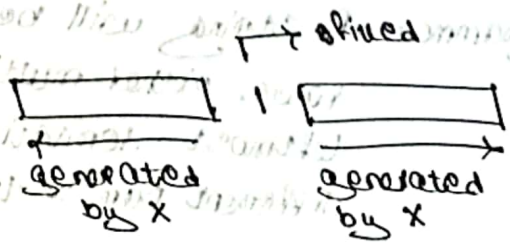
(ii)



* This string has 2 parse trees. So, the grammar is ambiguous grammar.

$S \rightarrow X1X$

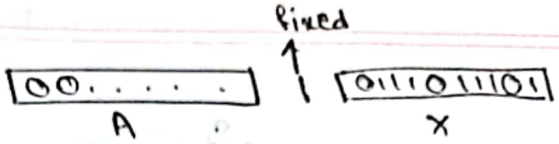
$X \rightarrow 0X|1X|\varepsilon$

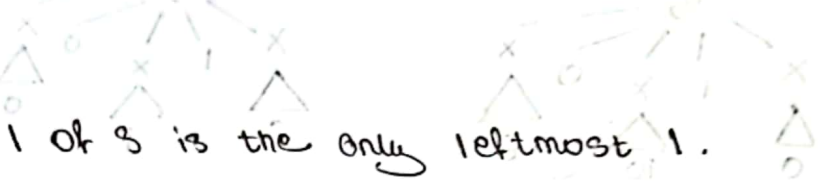


$S \to A \mid X$

$A \to 0A \mid \varepsilon$

$X \to 0X \mid 1X \mid \varepsilon$

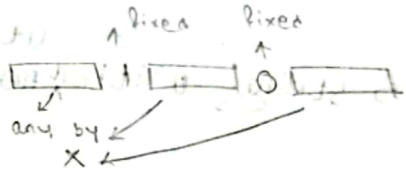




* Not ambiguous as $\neq 1$ of S is the only leftmost 1.

$S \to X \mid X0X$

$X \to 0X \mid 1X \mid \varepsilon$

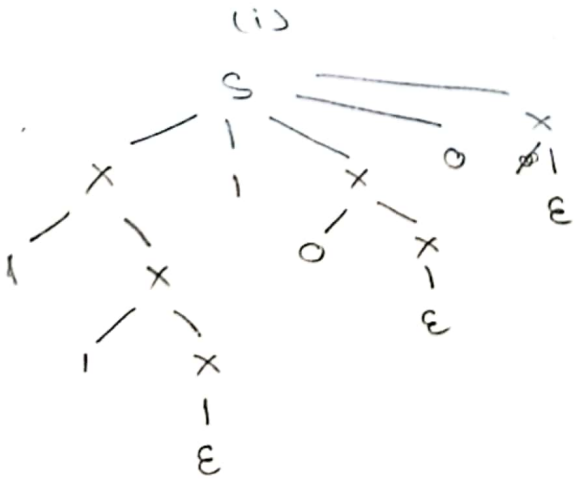


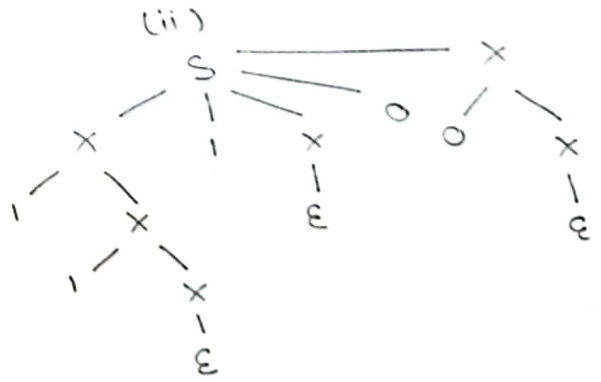
(a) Show that the grammar above is ambiguous by finding a length 5 string with two parse trees

11100

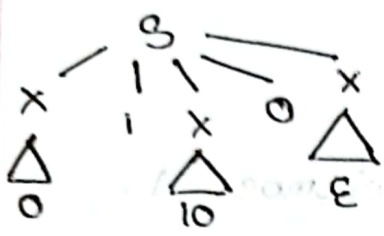
(i) $S \to X \mid X0X \to 1X1X0X \to 11X1X0X \to 111X0X \to 1110X0X$
$\to 11100X$
$\to 11100$

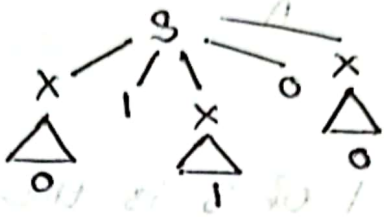
(ii) $S \to X \mid X0X \to 1X1X0X \to 11X1X0X \to 111X0X \to 1110X$
$\to 11100X$
$\to 11100$





01100





(b) Find a length 5 string ~~with~~ at least two without ~~NO~~ parse tree

00010

10111

$S \rightarrow S_1 C \mid A S_2$ (fixed: S_1 C or fixed: A S_2)

$S_1 \rightarrow a S_1 b \mid \varepsilon$ (a & b are equal)

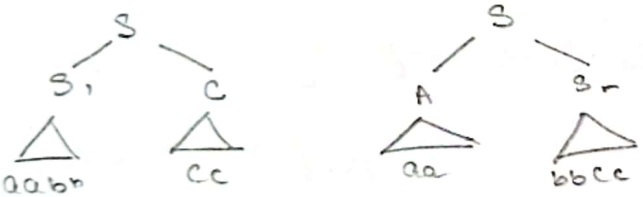
$S_2 \rightarrow b S_2 c \mid \varepsilon$ (b & c are equal)

$A \rightarrow aA \mid \varepsilon$ → যত খুশি তত a

$C \rightarrow cC \mid \varepsilon$ → যত খুশি তত c

(a) Show that the grammar above is ambiguous by finding a length 6 string with two different parse trees.

aabbcc : S_1 C or A S_2



LECTURE 14

WEDNESDAY

DATE: 29/03/23

DESIGN CFGs

(1) $0^n 1^n$, $n \geq 0$

$S \rightarrow 0S1 \mid \varepsilon$

$S \rightarrow 0S1 \rightarrow 00S11 \rightarrow 0011$ (2nd, 1st)

(2) $0^n 1^{2n}$, $n \geq 0$

$S \rightarrow 0S11 \mid \varepsilon$

$S \rightarrow 0S11 \rightarrow 00S1111 \rightarrow 001111$ (2nd, 1st)

(3) $0^{2n} 1^{3n}$, $n \geq 0$

$S \rightarrow 00S111 \mid \varepsilon$

$S \rightarrow 00S111 \rightarrow 00\ 00S111\ 111$ (2nd, 1st)

(4) $0^n 1^{2n+1}$, $n \geq 0$

$S \rightarrow 0S11 \mid 1$

$S \rightarrow 0S11 \rightarrow 0\ 0S11\ 11 \rightarrow 00S1111$ (1st, 2nd)

(5) $0^{2n} 1^{3n+2}$

$S \rightarrow 00S111 \mid 11$

$S \rightarrow 00S111 \rightarrow 00\ 00S111\ 111$ (2nd, 1st)

$\rightarrow 00\ 00S1111\ 111\ 11$

(6) $0^{2m} 1^n 2^{m+n}$

$0^m 1^n 2^n 2^m$

$S \rightarrow 0S2 \mid \$S_1$

$S_1 \rightarrow 1S_1 2 \mid \varepsilon$

$S \rightarrow 0S2 \rightarrow 00S22 \rightarrow 00S_1222$ (2nd, 1st)

$\rightarrow 001S_1222$ (2nd, 1st)

$\rightarrow 001222$

$m=2$, $n=1$

(7) $0^m 1^{m+n} 2^n$

$0^m 1^m 1^n 2^n$

$S \to S_1 S_2$

$S_1 \to 0 S_1 1 | \varepsilon$

$S_2 \to 1 S_2 2 | \varepsilon$

$S \to S_1 S_2 \to 0 S_1 1 | S_2 2$

$\to 0\ 0 S_1 1\ 1 \quad 1\ 1 S_2 2\ 2$

$\to 00\,1111\,22$

(8) CFG for palindrome

$S \to 0S0 | 1S1 | 0 | 1 | \varepsilon$

$S \to 0S0 \to 0\ 1S1\ 0 \to 0\ 1\ 0$

(9) $L = 0^i 1^j 2^k 3^m \quad j = 3k+2,\ i, j, k, m \geq 0$

$S \to 0S3 | S_1$

$S_1 \to 111 S_1 2 | 11$

$S \to 0S3 \to 0S_13 \to 0\ 111S_12\ 3$

$\to 0\ 111\ 11\ 2\ 3$

$i = 1,\ j = 3k+2 = 5,\ k = 1,\ m = 1$

$L = 0^i 1^j 2^k 3^m \quad j \neq 3k+2,\ i, j, k, m \geq 0$

$S \to 0S3 | S_1$

$S_1 \to 111 S_1 2 | 11T$

$T \to 1T | \varepsilon$

$S \to 0S3 \to 0S_13 \to 0\ 111S_12\ 3$

$\to 0\ 111\ 11T\ 2\ 3$

$\to 0\ 111\ 11\ 2\ 3$

$i = 1,\ j \neq 3k+2,\ k = 1,\ m = 1$

for $j \leq 3k+2$

$S \to 0S3 | S1$

$S_1 \to 111 S_1 2 | 11T$

$T \to 1T | \varepsilon$

$S \to 0S3 \to 00S33 \to 00S_133$

$\to 00111S_233 \to 00111T33 \to 00111133$

LECTURE 18

▢ BOOK → 2.14

MONDAY DATE: 03/04/23

Chomskey Normal form

Standard form used to make a certain CFG is the chomskey Normal form.

example from 137 of Book

$S \to ASA \mid aB$ — ASA: not allowed [3 variables]; aB: not allowed

$A \to B \mid S$ — B: not allowed; S: not allowed

$B \to b \mid \varepsilon$ — b: allowed; ε: not allowed

* start variable এর ডানে A ε থাকলে, তাহার
* ডানে, এ একটা / দুইটা symbol থাকতে হবে.
* একা থাকলে, symbol টা terminal হতে হবে.
* দুইটা থাকলে দুইটা symbol কেই variable হতে হবে, start variable থাকতে হবে না.

Step 1: Add a new start variable

$S_0 \to S$
$S \to ASA \mid aB$
$A \to B \mid S$
$B \to b \mid \varepsilon$

* There is no start variable in RHS now *

* যদি ডানে নিজে start variable থাকে, new start variable add করে use করা লাগবে.

Step 2: Remove ε-rules ~~+ Then solve the pro~~

$S_0 \to S$
$S \to ASA \mid aB \mid a$
$A \to B \mid S \mid \varepsilon$
$B \to b$

$A \to xBxBx \mid BxBx \mid BBx \mid BB \mid$

$x \to \varepsilon$

$S_0 \to S$
$S \to ASA \mid aB \mid a \mid AS \mid SA \mid S$ — changes made (AS | SA | S)
$A \to B \mid S \mid$ ~~ε~~
$B \to b$ — ε is removed

* ε is removed & changes are made to overcome the problems caused by ε in other production.

Once we see, S→S dumb remove it

$S_0 \to S$
$S \to ASA \mid aB \mid a \mid AS \mid SA$ — S is removed
$A \to B \mid S$ $B \to b$

Step 3: Remove unit rules
→ ডানে 1 একটা terminal থাকে

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow b \mid ASA \mid aB \mid a \mid AS \mid SA$

$B \rightarrow b$

ডানে একটি একটা terminal থাকলে, ওটার পুরা production টা লেখা লাগবে replacing that terminal.

Step 4: Fix everything else

$S_0 \rightarrow AX \mid YB \mid a \mid AS \mid SA$

$S \rightarrow AX \mid YB \mid a \mid AS \mid SA$

$A \rightarrow b \mid AX \mid YB \mid a \mid AS \mid SA$

$B \rightarrow b$

$X \rightarrow SA$

$Y \rightarrow a$

e.g. $A \rightarrow BCDEF$ (DEF → X_3, CDEF → X_2, … → X_1)

$A \rightarrow BX_1$

$X_1 \rightarrow CX_2$

$X_2 \rightarrow DX_3$

$X_3 \rightarrow EF$

LECTURE 16

WEDNESDAY

DATE: 05/04/23

CYK ALGORITHM

* A string and a grammar in chomsky normal form is given. We need to find out whether the given string can be derived from the given grammar.
* A "YES" or "NO" answer is returned.
* CYK is dynamic programming.

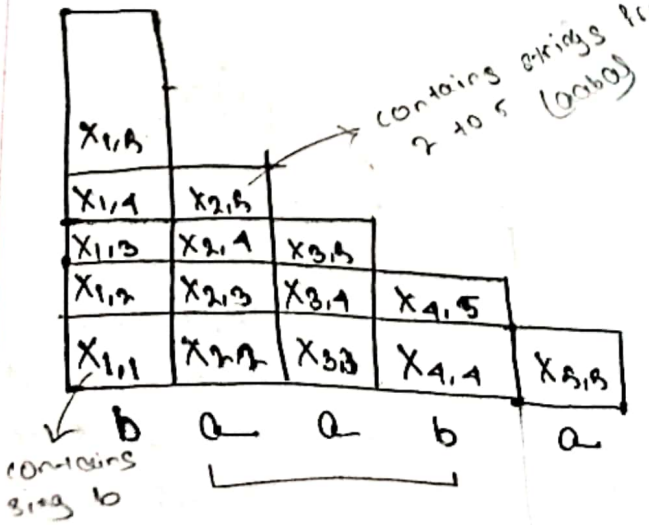
S → AB/BC
A → BA/a
B → CC/b
C → AB/a

} a grammar (in chomsky normal form)

baaba → string

* Step 1: A table with length of string x length of string dimension is taken.

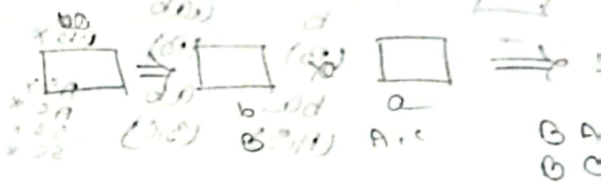
Step 2: The string is written at the bottom



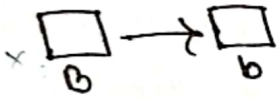
* For deriving a 'n' length string, we can use 2n-1 steps.

A,S,C				
—	A,S,C			
—	B	B		
A,S	B	S,C	A,S	
B	A,C	A,C	B	A,C
b	a	a	b	a

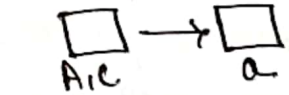
* for $X_{1,2}$, ba → 2n-1 = a variable is split into 2 and first variable needs to be b and the 2nd one needs to be a



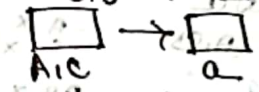
* $X_{1,1} \rightarrow b$



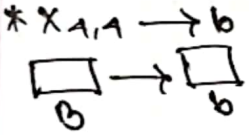
* $X_{2,2} \rightarrow a$



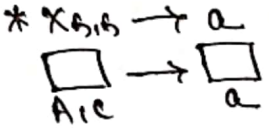
* $X_{3,3} \rightarrow a$



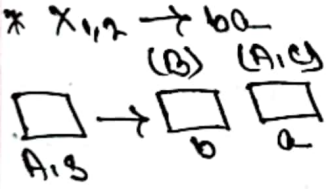
* $X_{4,4} \rightarrow b$



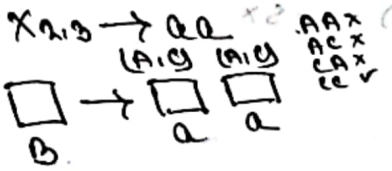
* $X_{3,3} \rightarrow a$



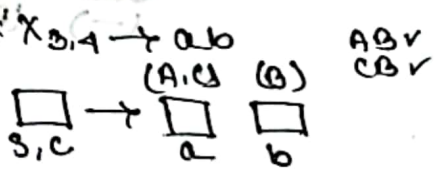
* $X_{1,2} \rightarrow ba$ (B) (A,C) BA✓ BC✓



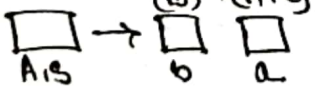
* $X_{2,3} \rightarrow aa$ (A,C) (A,C) AA✗ AC✗ CA✗ CC✓



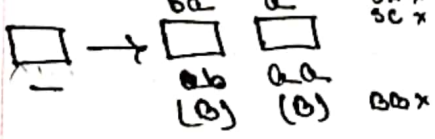
* $X_{3,4} \rightarrow ab$ (A,C) (B) AB✓ CB✓



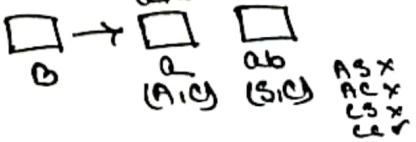
* $X_{4,5} \rightarrow ba$ (B) (A,C)



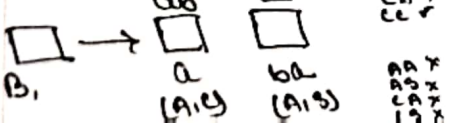
* $X_{1,3} \rightarrow baa$ (A,S) (A,C) ba a AA✗ AC✗ SA✗ SC✗ bb aa (B) (B) BB✗



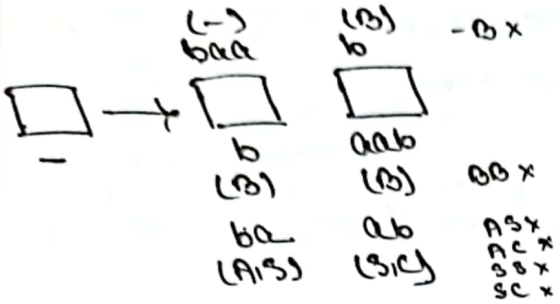
* $X_{2,4} \rightarrow aab$ (B) (B) aa b BB✗ a ab (A,C) (S,C) AS✗ AC✗ CS✗ CC✓



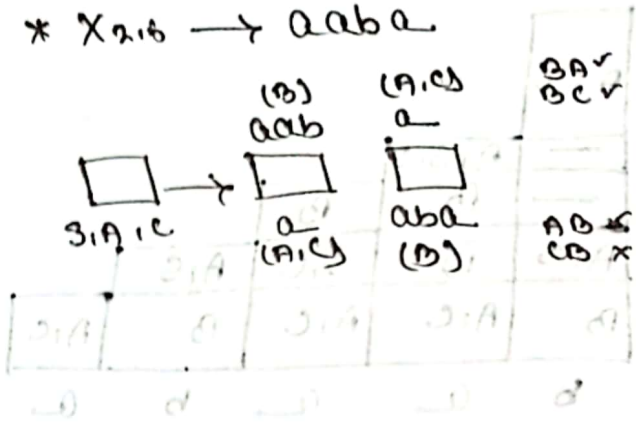
* $X_{3,5} \rightarrow abb$ (S,C) (A,C) ab a SA✗ SC✗ CA✗ CC✓ a ba (A,C) (A,S) AA✗ AS✗ CA✗ CS✗



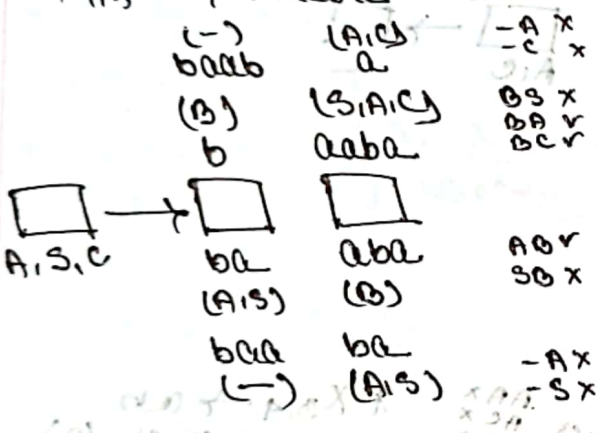
* $X_{1,4} \rightarrow baab$



* $X_{2,6} \rightarrow aaba$



* $X_{1,5} \rightarrow baaba$



LECTURE 17

MONDAY

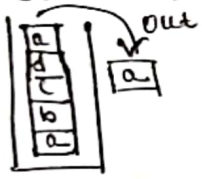
DATE: 10/04/23

PUSHDOWN AUTOMATA (PDA)

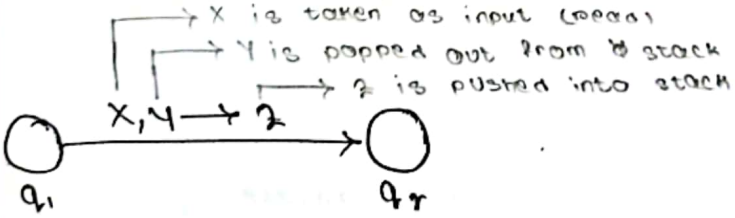
* PDA is like DFA/NFA, but for context free language.
* Memory of DFA and NFA is finite/limited. So, there is a limitation of how much DFA/NFA can perform.
* PDA has some access to some auxilary memory.

PDA = NFA + auxilary memory
- infinite
- stack

* PDA is a NFA with an access to an infinite stack.
  * stack is a FILO structure.



* Using PDA, string can be pushed into stack.



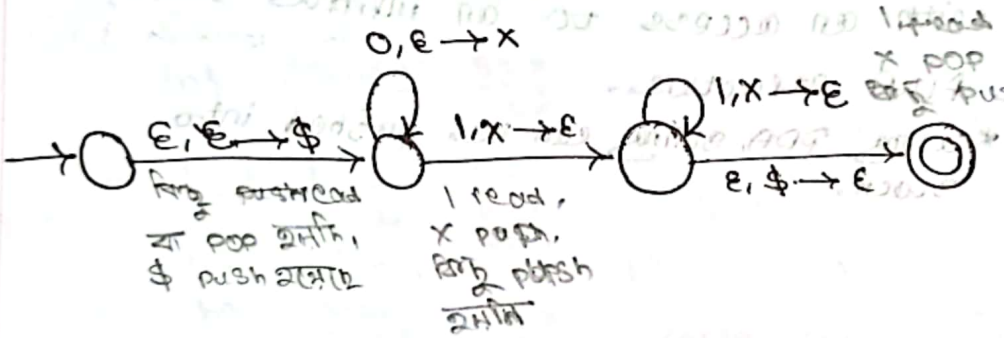
From q_1, X is read as input, Y is popped out of the stack & Z is pushed into the stack-to go to q_2.

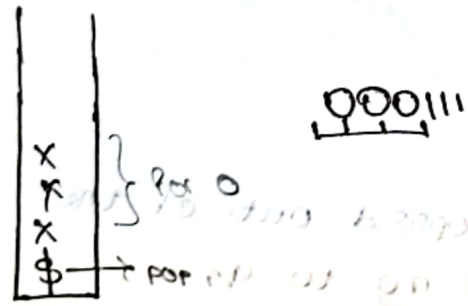
* If there is no 'Y' on the top of the stack, the state change is not done.

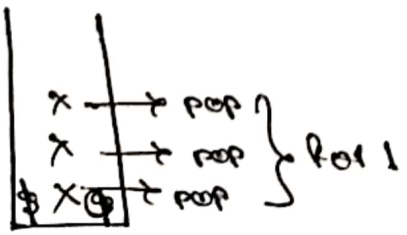
$0^n 1^n , n \geq 1$

* 0 এলেই stack push
* 1 এলেই pop
* stack underflow হলে reject

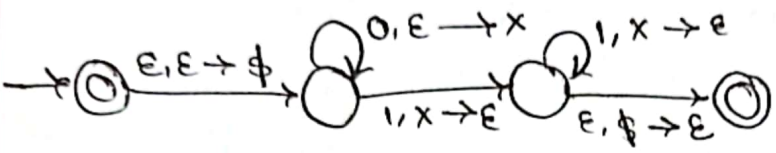
* A $ is pushed at the very first of the PDA before seeing the input.
* To check whether the stack is empty or not, $ is popped and then the stack is empty







111

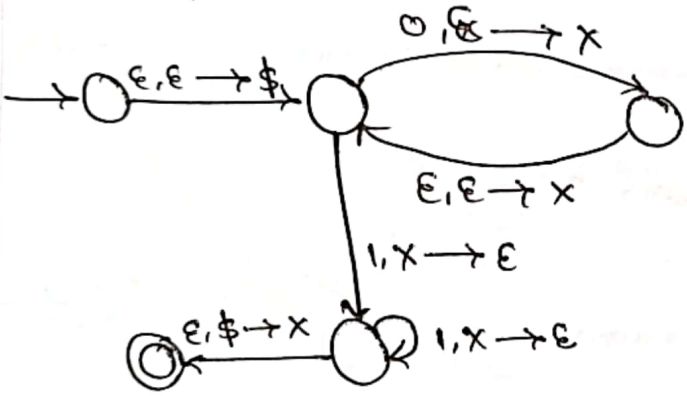


* In PDA, we cannot push two strings at a time

* For stack, NFA is more powerful than DFA.

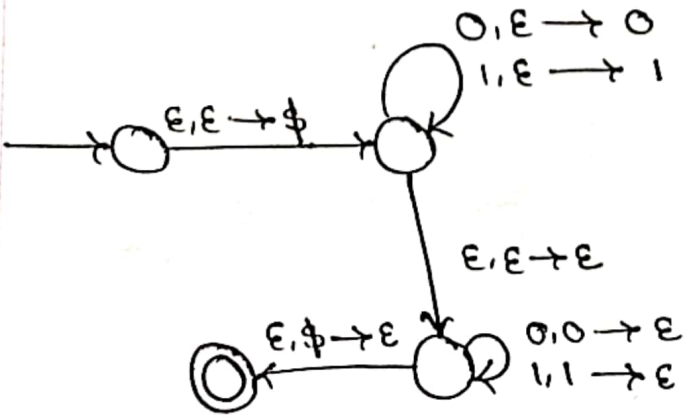
* We can pop or not pop
* We can push or not push.

Done by using ε

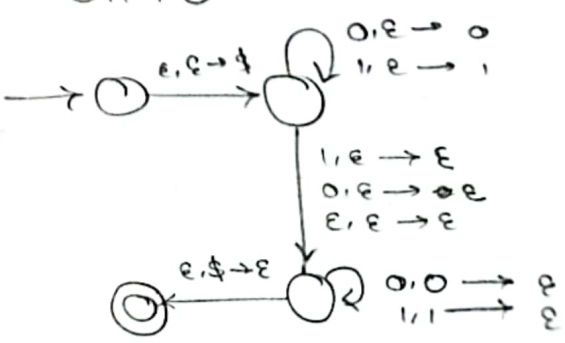


even length palindrome

* string is first half is pushed.
* second half of the string is popped
* one string is taken, pushed and then popped to one



* 01110



For all palindrome

LECTURE 18

00 , 0^n , 1^{2n}

WEDNESDAY

DATE: 12/07/23

* Pumping Lemma - not in final
  - Assignment এ ছিল

* Context free language & context free grammar
  - Language দিয়ে দিবে grammar করা লাগবে.

* Ambiguity
  - Grammar দিয়ে দিবে Are ambiguity prove করা লাগবে parse tree দিয়ে।

* Chomsky Normal Form
  - Grammar কে দেওয়া থাকবে chomsky normal form এ convert করতে হবে.
  - Rule violate না করে লিখতে হবে

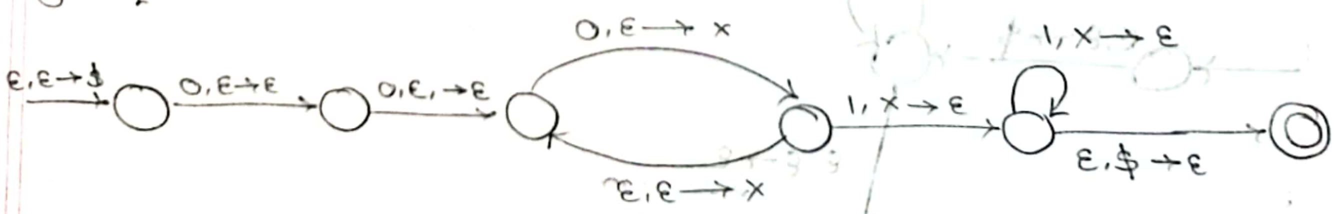
* CYK Algorithm
  - Box fill up করে বলা লাগবে string derive করা যাবে কি না using CYK Algorithm

* PDA
  - Language দেওয়া থাকবে PDA design করা লাগবে

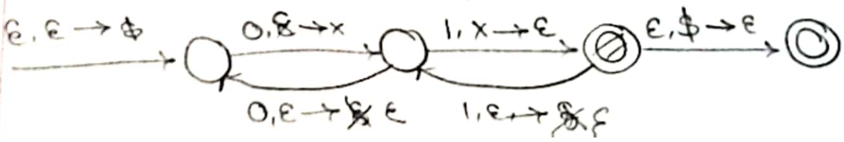
[Lecture 20 & 21 of PDA]

$0^{n+2}1^{2n}$
0^n1^n, n = odd } Fall'22 (Final)

$0^{n+2}1^{2n}$



0^n1^n, n = odd



LECTURE 19

MONDAY

REVISION CLASS

WEEK 10 (BUX)

(Q) $\Sigma = \{a, b\}$, $L = \{a^m b a^n b a^{m+n} \}$; $m, n \geq 0$

CFG :-

$S \rightarrow aSa \mid bS_1$

$S_1 \rightarrow aS_1a \mid b$

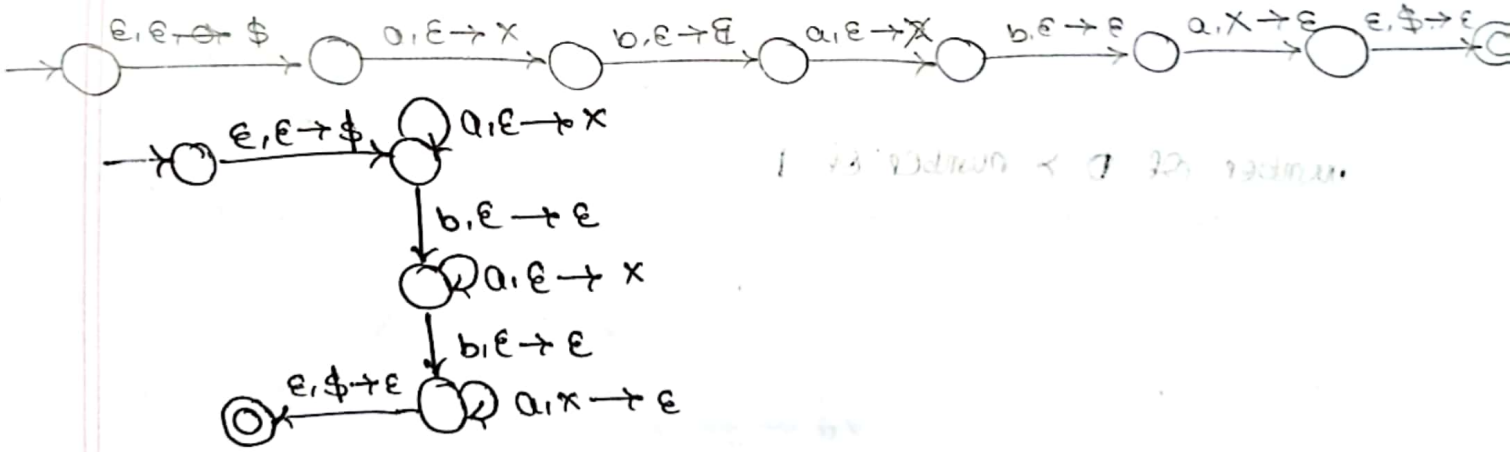
$a^m b a^n b a^{m+n}$

$\rightarrow a^m b a^n b a^m a^n$

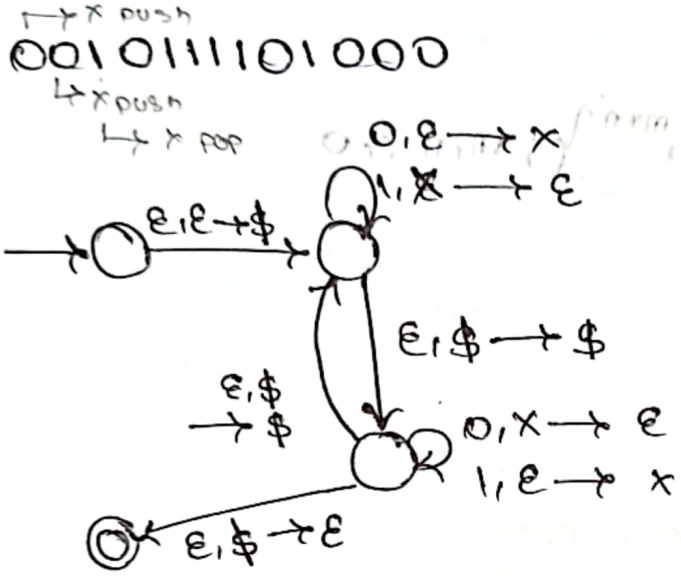
$S_0 \rightarrow aS_0a \mid bS_1$

$S_1 \rightarrow aS_1a \mid b$

PDA :-



number of 0 = number of 1

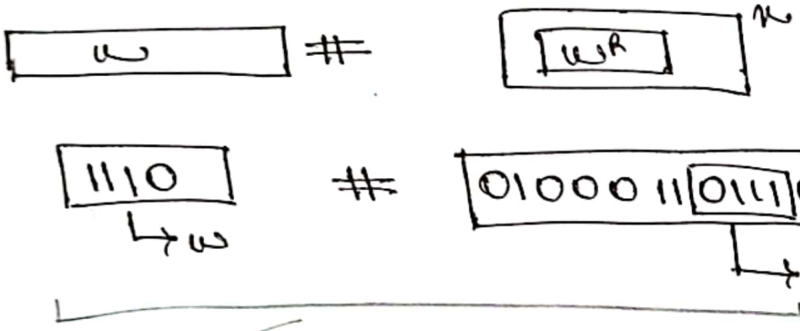


number of 0 > number of 1

2.6

(c) $\{w\#x \mid w^R \text{ is a substring of } x \text{ for } w, x \in \{0,1\}^*\}$

$w = 1101 \qquad w^R = 1011$



$S \rightarrow MX$

$M \rightarrow 1M1 \mid 0M0 \mid \#X$

$X \rightarrow 0X \mid 1X \mid \varepsilon$

PDA:-