Theory Of Automata and Formal Languages 13-03-23

- valid characters of your language → Alphabets.
- Sigma set = Σ = {set of alphabets of your language x}
- Formal Language = collection of operators & symbols.
- multiple characters make a string/combination of one or more characters.
- word = valid string of your language.
- Rules to check validity.
- Regular Expression (Reg Ex) = Rules will be written in the form of Reg Ex.

  match ("int", Reg Ex).
- Identifier = variables (user defined).
- Tokenization: a word belongs to which class & it is valid ← tokenization.

Finite State machine.
↳ an algo to check a given word applies Reg Ex.

Decider Machine.
↳ decides for regular languages (expressions).

Characters + Rules (RegEx) → Language.

Push down automaton.

14-03-23

Regular Expression.

$\Sigma = \{a\}$

Reg Ex.
(*) Kleen's Closure.
+ Empty Closure.

$a^* = \{\lambda, a, aa, aaa, aaaa, \ldots\}$

L_1 = {All strings of 'a' of any length}.

λ	0	$\Sigma^0 = \{\lambda\}$
a	1	$\Sigma^1 = \{a\}$
aa	2	$\Sigma^2 = \{a, a\}$
aaa	3	$\Sigma^3 = \{a, a, a\}$.

λ or ϵ - string of 0 length

$\Sigma^* = \{\lambda, a, aa, aaa, \ldots\}$.

$\overset{+}{a} = \{a, aa, aaa, aaaa, \ldots\}$.

λ is not acceptable.

$L_2 = \{$ All strings of a at least one length $\}$.

- When we concatenate or take union of 2 Reg Ex, a new Reg Ex can be made.

Concatenation

$L_3 = \{a, b\}$

$L_4 = \{c, d\}$ $\quad L_4L_3 = \{ca, cb, da, db\}$

$L_3L_4 = \{ac, ad, bc, bd\}$.

ac is a word of L_3L_4 $\quad L_3L_4 \neq L_4L_3$.

$W = ac. \wedge \; W \in L_3L_4$.

$ca \notin L_3L_4$.

$\overset{*}{a}\overset{+}{a} = \{\lambda, a, aa, aaa, \ldots\}\{a, aa, aaa, \ldots\}$ $\lambda \cdot a = a$

$= \lambda\{a, aa, aaa, \ldots\}$. $\quad a \cdot \lambda = a$.

$= a\{a, aa, aaa, \ldots\}$.

$= \{a, aa, aaa, aaaa, \ldots\}$.

$\overset{*}{a}\overset{+}{a} = \overset{+}{a}$ or $\overset{+}{a}\overset{*}{a} = \overset{+}{a}$

- concatenation is not always commutative but this example is!

Ex: 2 Union

$\Sigma = \{a, b\}$

$L_5 = \{a\}$

$L_6 = \{b\}$

$L_7 = \{a, b\}$. ✓ $a + b$

$L_7 = L_5 \cup L_6$

$(a+b)^* = \{a, b\}^* = \{\lambda, a, aa, aaa, \ldots,$
$b, bb, bbb, \ldots,$
$ab, ba, abb, bba,$
$abbba, abba, aaab \ldots\}$

$(a+b)^* = \{$ Any string of $a, b\}$

match $(b, (a+b)^*)$.

L_{10} = { all strings of a, b that starts with a }.
$a(a+b)^*$

L_{11} = { All strings of a, b that start with a and at b }.
$a(a+b)^* b$.

20-03-2023

$a^* b^* = \{\lambda, a, aa, aaa, \ldots\}\{\lambda, b, bb, bbb, \ldots\}$.
$= \{\lambda, a, aa, aaa, \ldots, b, bb, bbb, \ldots, ab, abbb, aa\,bbb \ldots, aabb,$

Describe the language of the following regular expression.
$a^* b^*$ = { All strings of a, b in which there is no a after b }

Books.
① Automata Theory by cohen
② Introduction to Theory of computation by Michael Sipser
③ Intro. To TOC 3rd by Ulman

$a^* b^+ = \{\lambda, a, aa, aaa, \ldots\}\{b, bb, bbb, \ldots\}$.
$= \{b, bb, bbb, \ldots, ab, abb, \ldots, aab, aabb, \ldots\}$
$a^* b^+$ = { all strings of a, b in which there is no a after b and if a, it must followed by b & a string must end at at least one b }

$(b + ab)^*$ = { all strings of a, b in which either only single b is acceptable or all strings of a, b in which no consecutive a }

$(ba^*b)^*$ = {all strings of a, b in which λ or

$(ba^*b)^*$ = {λ, bb, bbbb, ..., bab, baab, babbab, bbbab,

1) Even no of b's ✓
2) Start & end with b ✓
3) There should no aba substring ✓
4) ~~No single length string~~ ✗ atleast 2 length string

① Null is acceptable
② Strings of atleast two b's or even no of.
③ all those strings in which there is no substring aba & string must end at b. with even no of bs

① Even b's are acceptable. or

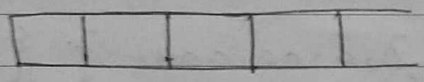
$(a+b)^*\ aa^+(a+b)^*$ (HW).

FINITE STATE MACHINE (FA/FSM) 21-03-2023

The decider of regular language.

Components

① Input Tape

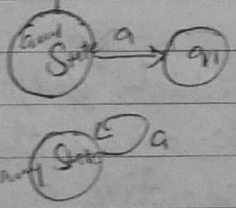


Types:

Deterministic

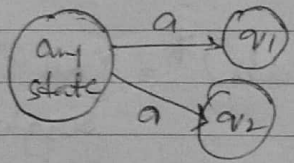
One-way

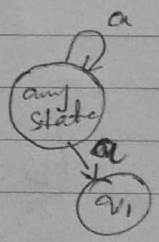
Example machine



Non-deterministic.

Example





can choose any one path

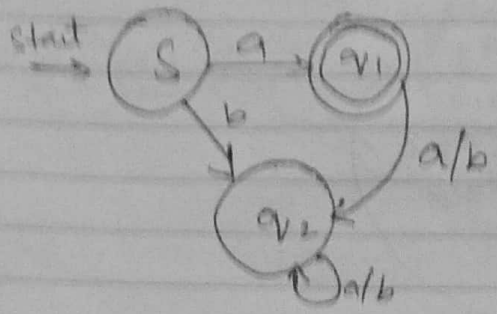
Acceptance State/Final state
Rejection State/Non-final State.

$M(S, \Sigma, \delta, F)$

δ(current state, a) = q1 (next state)

move

$L = \{a\}$ $\Sigma = \{a, b\}$



if(ch == a)
accept if(ch==a || ch==b)
else reject
else reject.

Input Tape | a | b | b | /0 |

Transition Table

	a	b
S	$\delta(S,a)=q_1$	$\delta(S,b)=q_2$
q_1	$\delta(q_1,a)=q_1$	$\delta(q_1,b)=q_2$
q_2	$\delta(q_2,a)=q_2$	$\delta(q_2,b)=q_2$

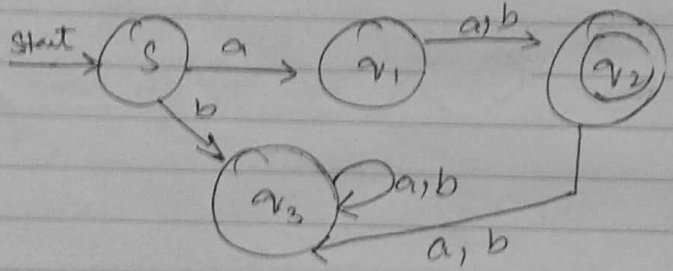
27-03-23

Finite State Machine

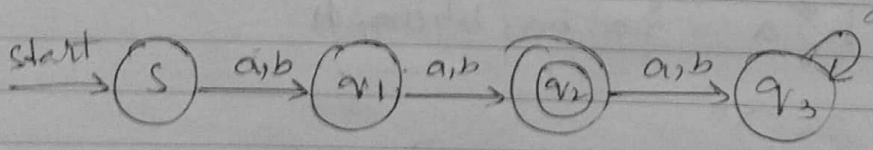
$\Sigma = \{a, b\}$

$L_2 = \{aa, ab\}$

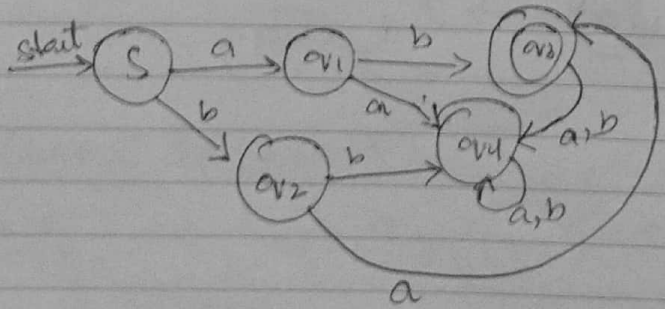
DFA => Deterministic Finite Automata.



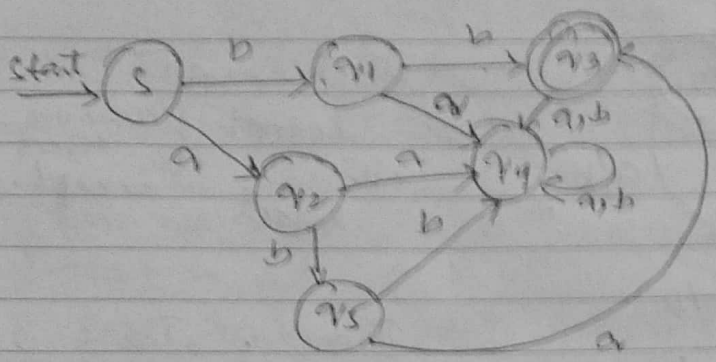
$L_3 = \{ab, ba\}$.

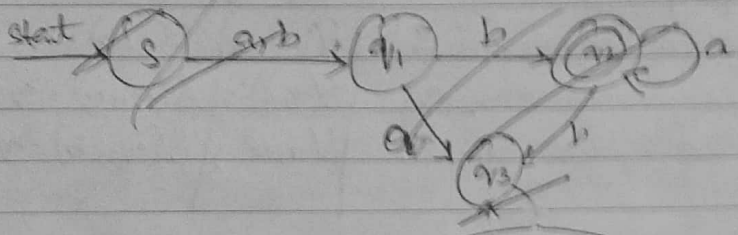


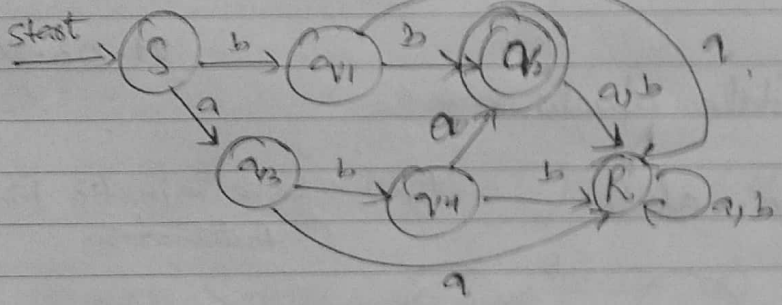
wrong



$L_4 = \{bb, aba\}$.

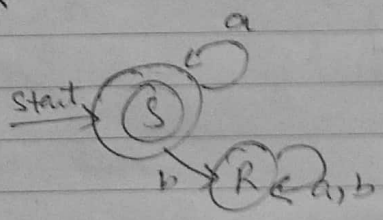




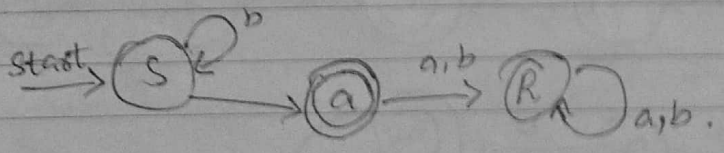


$\Sigma = \{a, b\}$

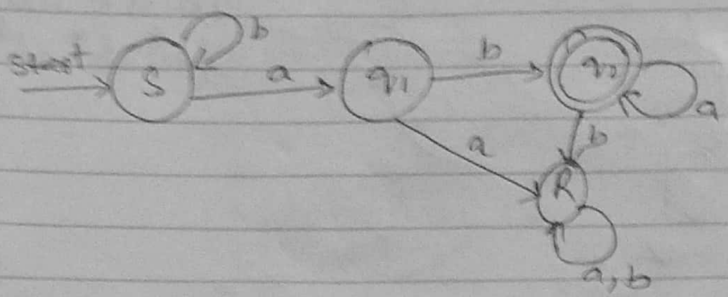
$L_5 = a^*$



$L_6 = b^*a = \{a, ba, bbba, \ldots$

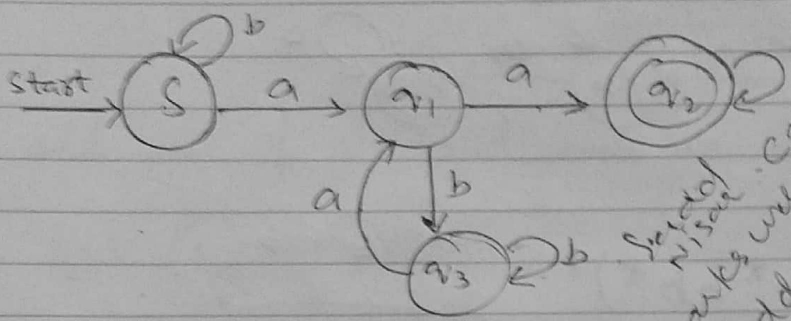


$L_7 = b^* a b a^* = \{ab, baba, \ldots$



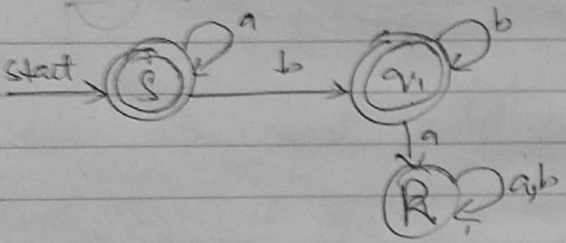
28-03-2023

$L_8 = (a+b)^* aa (a+b)^*$

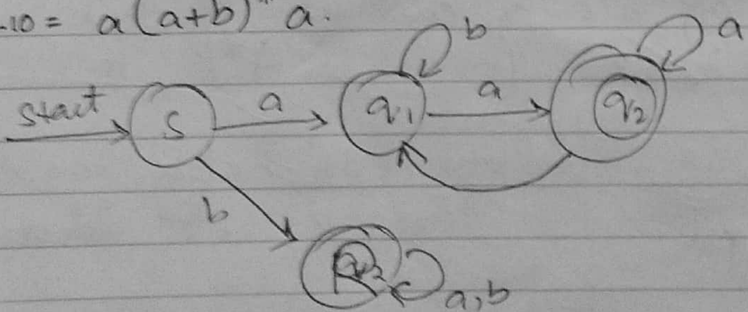


Syed Nisar CS211246 1 marks will be added in final. 28/march/2023

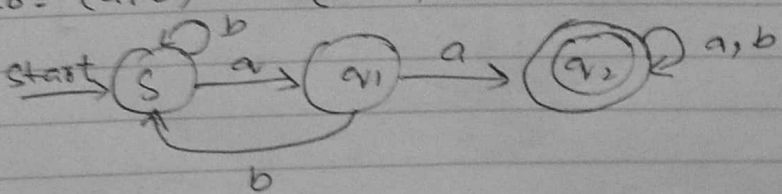
$L_9 = a^* b^*$



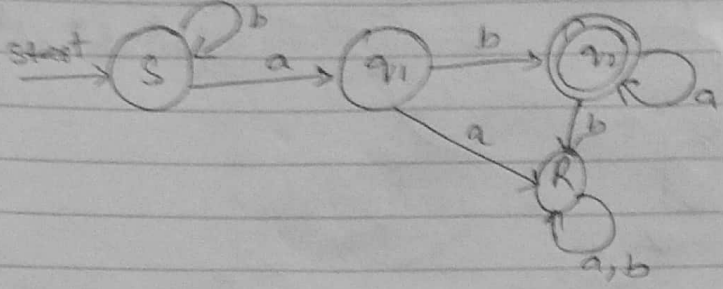
$L_{10} = a(a+b)^* a$



$L_8 = (a+b)^* aa (a+b)^*$

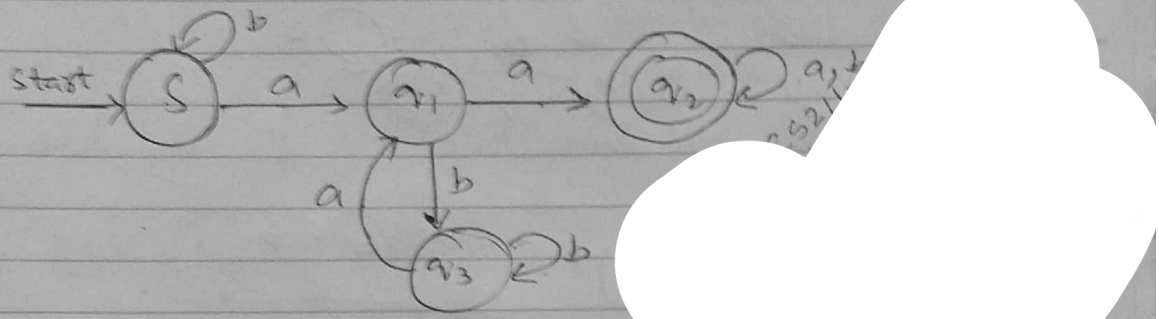


$L_7 = b^* a b a^* \cdot \{ab, baba, \ldots$

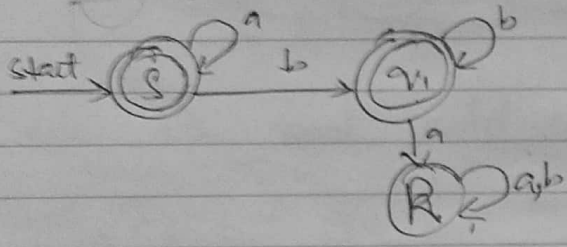


28-03-2023

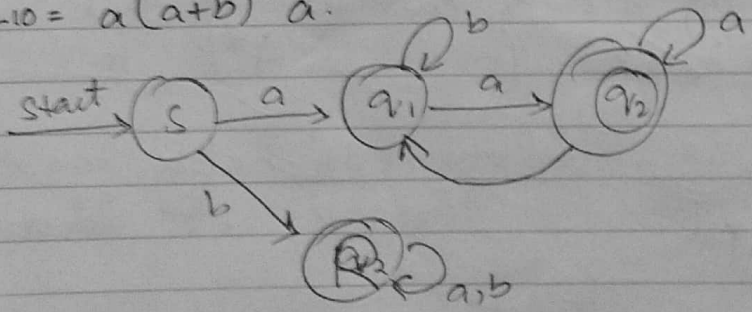
$L_8 = (a+b)^* aa(a+b)^*$



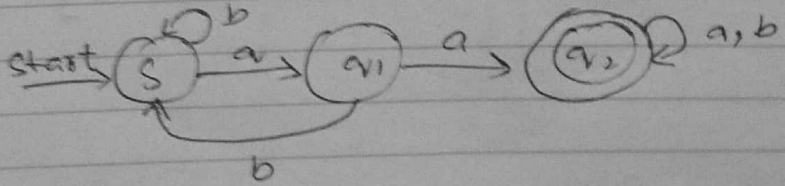
$L_9 = a^* b^*$



$L_{10} = a(a+b)^* a$



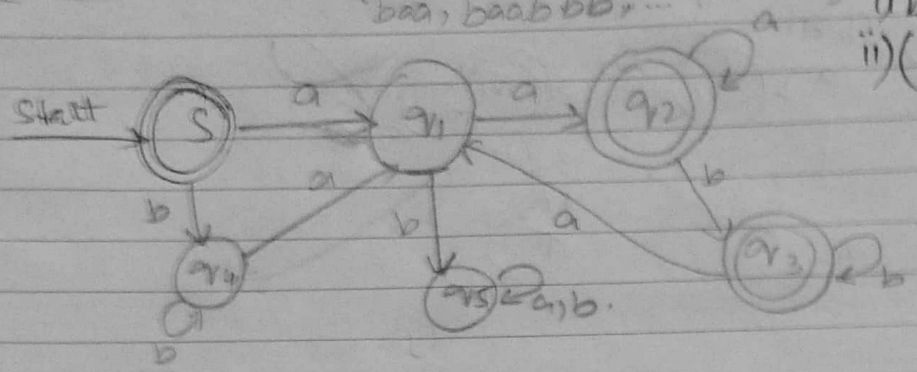
$L_8 = (a+b)^* aa(a+b)^*$



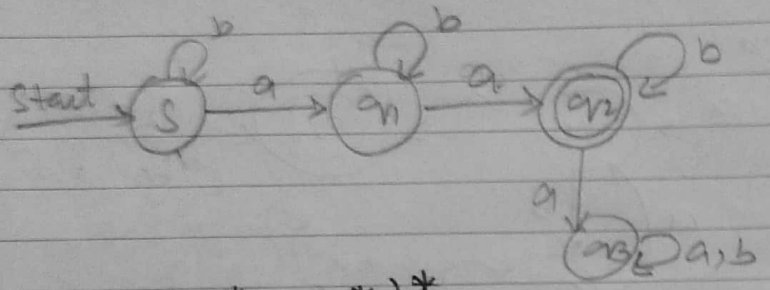
3/4/23.

$(b^* a a^+ b^*)^* = \{\lambda, aa, aaa, \ldots, aabaa \ldots$
$baa, baabbb, \ldots\}$

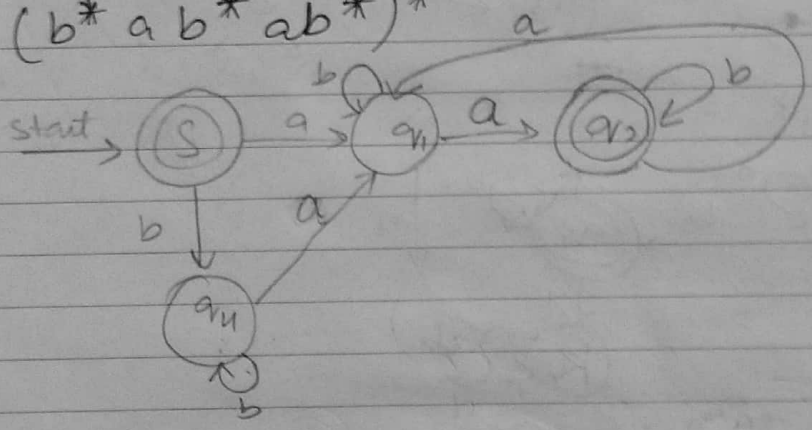
i) $b^* a b^* a b^*$
ii) $(b^* a b^* a b^*)^*$



i) $b^* a b^* a b^*$



ii) $(b^* a b^* a b^*)^*$



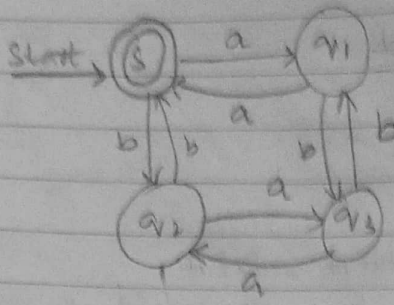
iii) even no. of a's & even no. of b's
~~for exactly 2 a's.~~ $(\quad)^*$

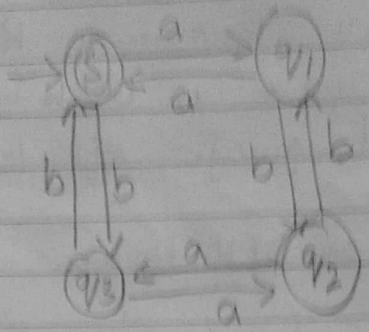
$= \{\lambda, aa, bb, aabb, abba, baba, aaabba,$
$\underline{ab}\quad\underline{ab}\quad aaabbabb$

$(aa+bb)^*$

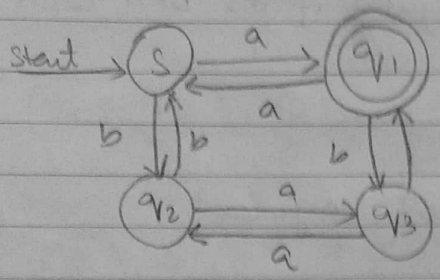
$[aa + bb + (ba+ab)(aa+bb)^*(ab+ba)]^*$

$[aa + bb + (ba+ab)(aa+bb)^*(ab+ba)]^*$

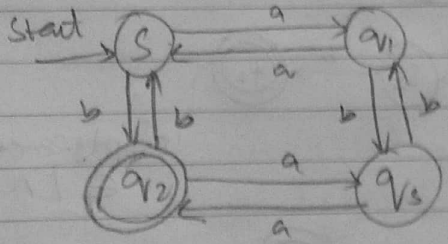


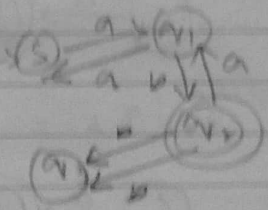


odd no. of a's and even no. of b's.

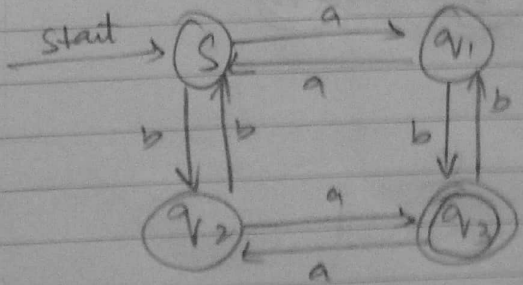


even a's and odd b's.



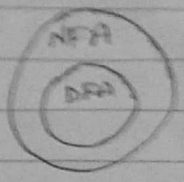


odd a's & odd b's.



10/04/23

NFA



Every NFA is a DFA
but not vice versa.

Kleen's Theorem. R.E = NFA = DFA
if $\forall w \in L_1$, w is accepted by DFA-"M_1".
language of M is L
$M_1(L) = L_1$
NFA or DFA kipaxu barabar he.

Pumping Lemma:

Recognize when a given language is not regular
→ a language is not regular if it's DFA can't be made.

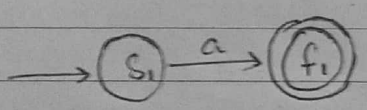
if $\exists w \in L \ni$
$w \neq x y^n z$
Then language is not regular

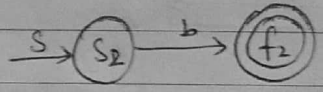
11/04/23

NFA

From Regular to NFA with empty moves:

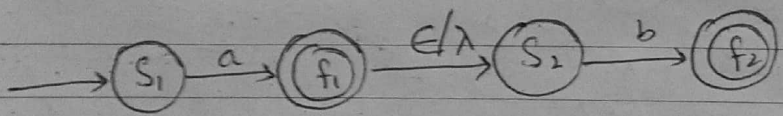
$r_1 = a$ $\qquad$ $r_2 = b$

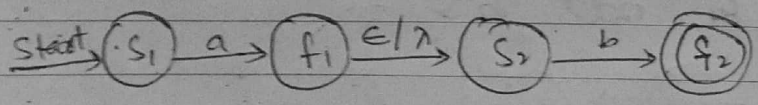




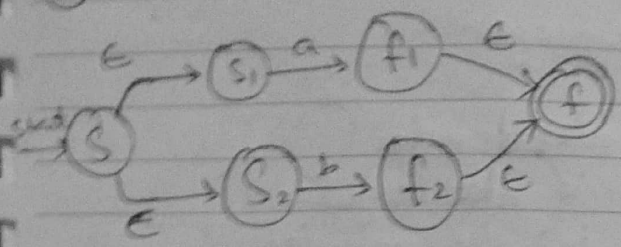
Empty-moves
$\in / \lambda$

$r_1 \cdot r_2 = ab.$

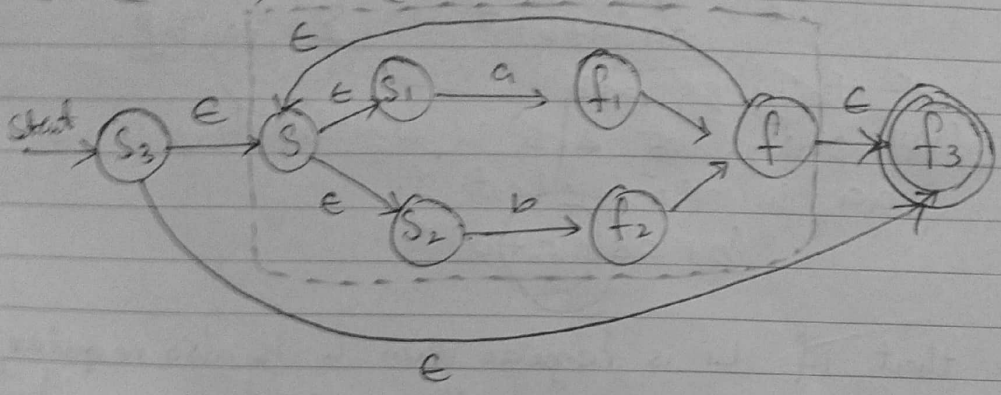




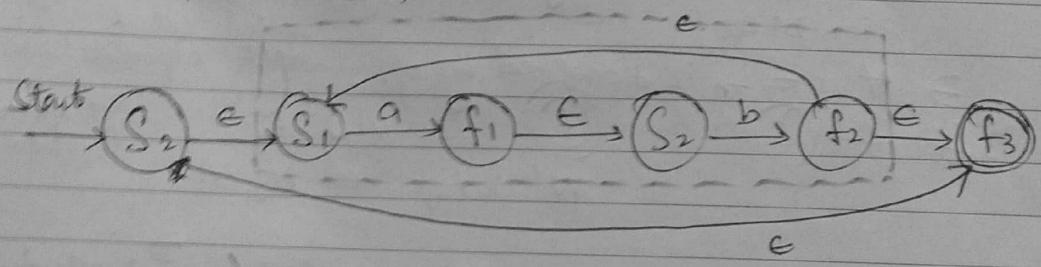
② $r_1 + r_2 = a + b$.



③ $(r_1 + r_2)^* = (a+b)^*$



④ $(r_1 r_2)^* = (ab)^*$



Transition function of NFA. $\hat{\delta}$

$\hat{\delta}$ (currentState, alphabet) = set of states.

$\Rightarrow \hat{\delta}(S, a) = ?$

for example # 3.

$\hat{\delta}(S_3, a) = \{S_3, f_1, f, f_3, S_1, S\}$.

$\hat{\delta}(S_3, b) = \{S_3, f_2, f, f_3, S, S_2\}$.

Automata Theory
By Cohen
Ch: 3 onwards.

Closure Properties Of R.E.

If L_1, L_2 are regular Languages.

i) Then L_1' is also regular
ii) Then $L_1 \cap L_2$ is also regular
iii) Then $L_1 \cup L_2$ is also regular.

L_1 = {All strings of a, b that start with aa}
$aa(a+b)^*$ $\quad \Sigma = \{a, b\}$

~~L_2 = All strings of a, b~~

L_1' {All strings of a, b in which there is no aa in the start}.

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

Prove that if L_1 is regular then L' is also regular.

Solution

Given that L_1 is regular $\Rightarrow \exists$ a $M(S, \Sigma, \delta, Q, F)$ this employs. (initial state, set of alphabets, set of states, set of final states, Transition function)

→ for DFA.

$\Rightarrow \forall w \in L_1$ is accepted by M. ie $M(L) = L_1$

such that start (s) ~~~ (f)

w' is rejected by M.

Now, transform this DFA M into 'M' $\Rightarrow$ such that Move all $Q \in F$ into F' and all F' states into F.

start (s) ~~~ (f)

This DFA M' will accept all $w' \in L_1'$
Hence, proved that L_1' is also regular.

$L_2 = (a+b)^* \; aa \; (a+b)^*$

$L_2' = (b+ab)^* + a \cdot (b+ab)^* (\lambda + a)$.

18-04-23.

Simulation Of DFA

Assignment No. 1.

Simulation: imitation of reality.

① Input number of states: Q

② Input number of alphabets: Σ

States [ ] = new statesArray (no. of states)

Loop: Input states in states list.

Alphabets [ ] = new alphabetArray (no. of alphabets).

TransitionTable [,] [,] = states x alphabets.

Set transition table.

initialState = "S"

currentState = initialState.

initialState = "S"
currentState = initialState.

FINALS

$(aa)^*(bb)^*b$

$(aa)^*(a+b)^*bb)^*b$.

08-05-23.

CONTEXT FREE LANGUAGES (CFL):

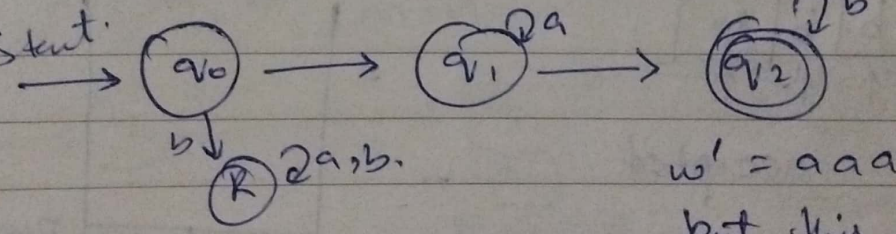
Claim:

There exist non-regular languages.

To prove this claim we will give an example. Language $L = a^n b^n ; n \geq 1$ that is not regular i.e. there does not exist any DFA which can accept all $w \in L$ and reject all $w' \notin L$.

$L = a^n b^n \quad n \geq 1$

$L = \{ab, aabb, \ldots\}$.

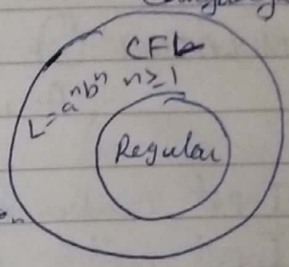


$w' = aaab \notin L$

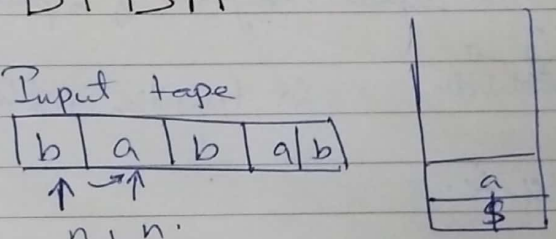
but this DFA accept this w'.

Push Down Automata (PDA).

- Those ~~free~~ languages which decides by PDA are context-free languages.
- All regular languages are context-free but not all context-free are regular.
- PDA's have memory element.
- Push Down Automatan is a decider of CFL.
  - → Deterministic Push Down Automaton (DPDA).
  - → Non-deterministic Push Down Automaton (NPDA)

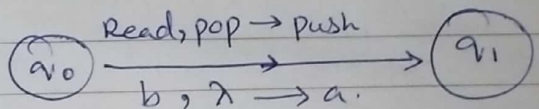


DPDA



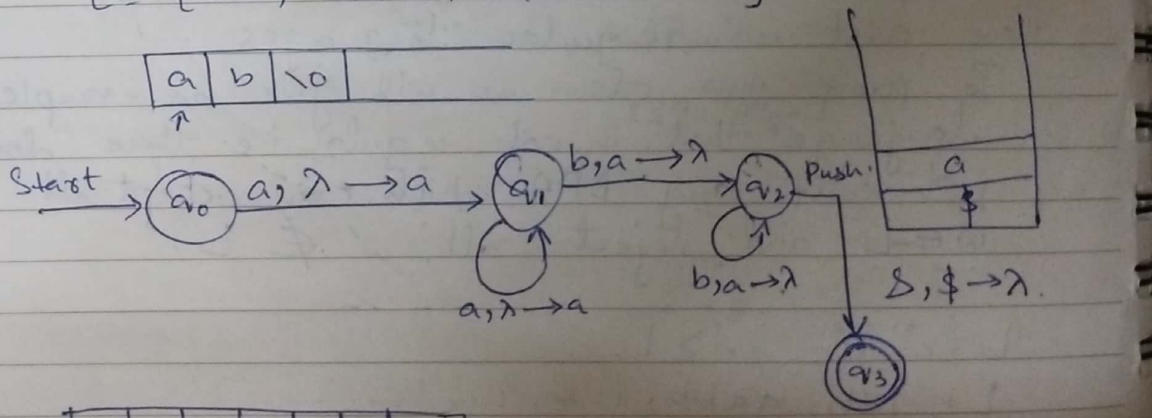
$ => empty stack.

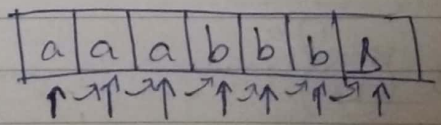
$L = a^n b^n$.



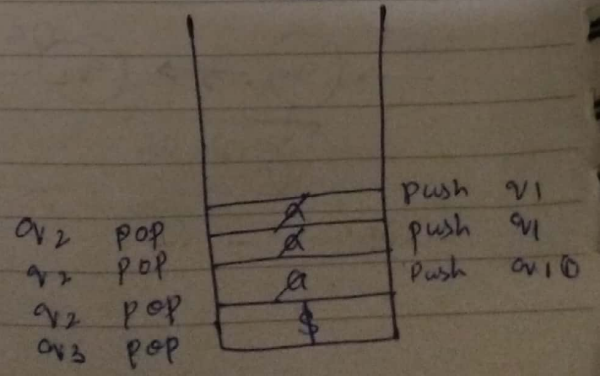
$L = a^n b^n \; ; n \geq 1.$

$L = \{ ab, aabb, aaabbb, \ldots \}$.



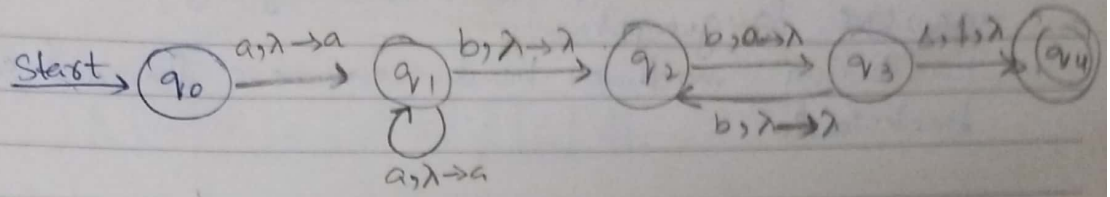


accepted

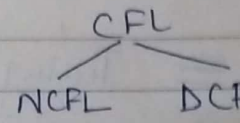


$L = a^n b^{2n}$.

$L = \{abb, aabbbb, aaabbbbbb, \ldots\}$.



NPDA >> DPDA. NPDA ≠ DPDA.



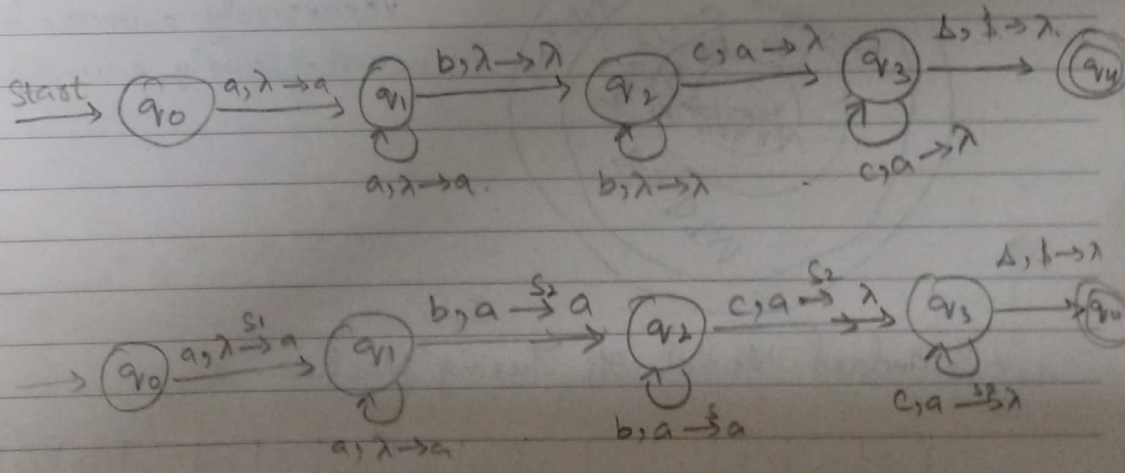
15-03-2023

. CFL are decideable by PDA and not by DFA.

$L_1 = a^n b^n \quad n \geq 1$

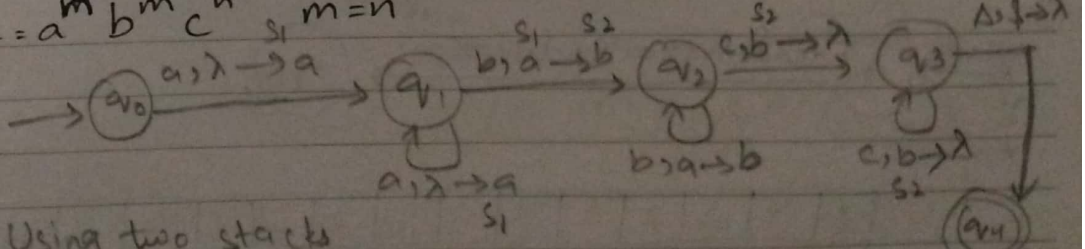
$L_1 = \{ab, aabb, \ldots\}$.

$L_1 = a^n b^m c^n \quad m \neq n$. (Non-context free languages)



This language can't be decided by PDA

$L_2 = a^m b^m c^n \quad m = n$



Using two stacks PDA can decide

RECURSIVE LANGUAGES:

- Turing decideable languages are recursive languages.
- There exist non-context free languages which are recursive languages.
- In turing machine, head can move forward or backward.

Context free = Context free' CFL = CFL'
Non-context free = Non-context free' NCFL = NCFL'

- 2-Stack PDA ≡ Turing Machine

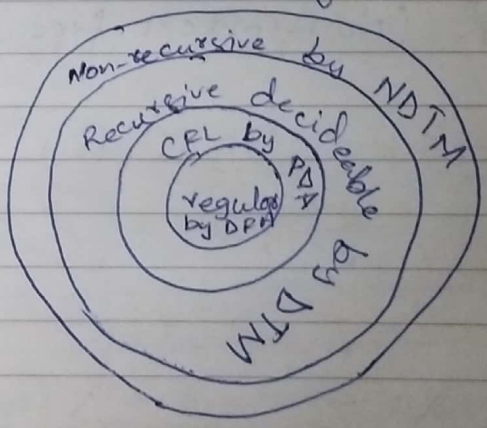
16-05-23.

Instantaneous Description / Computation.

$L = a^n b^n c^n \quad n \geq 1$. (can't be decided by PDA).

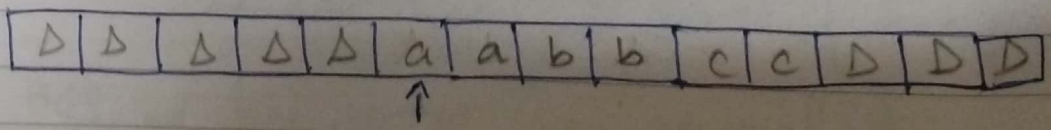
TURING MACHINE:

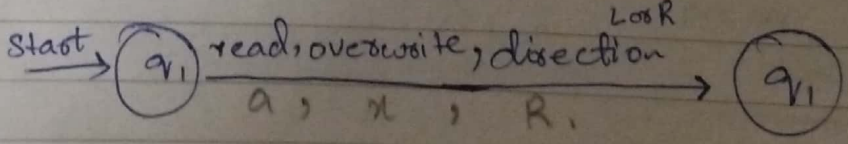
The decider of all recursive languages.

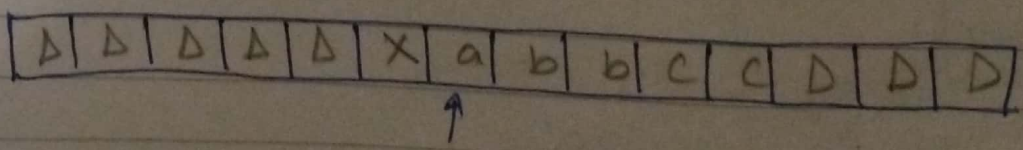


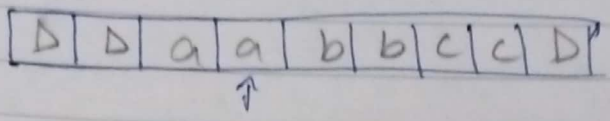
recursively enumerable
non-recursively enumerable.

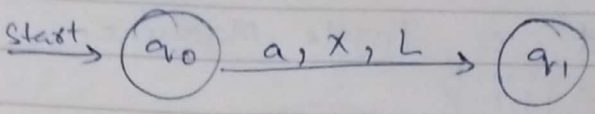
Standard Model of Turing Machine.

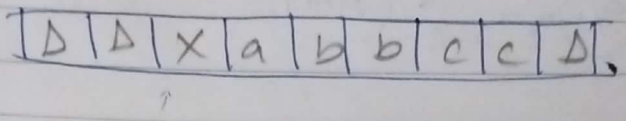








ii) 



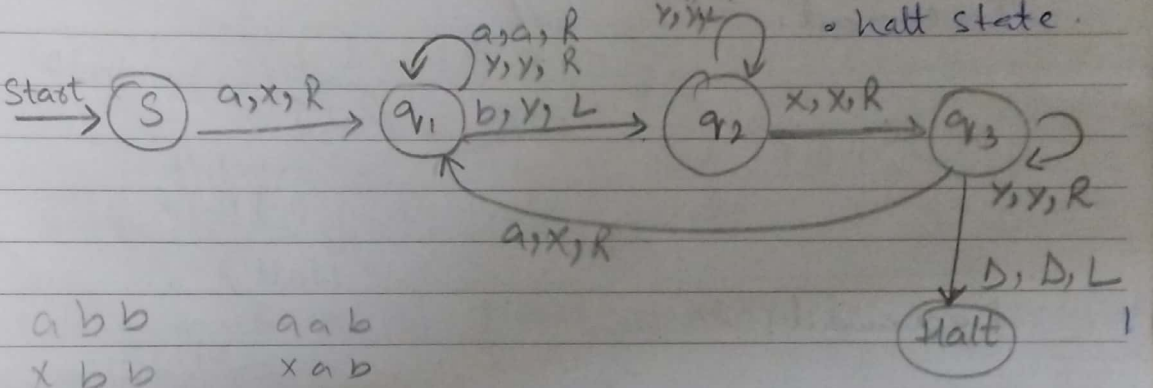
* Every algo has Turing machine & vice versa.

$L = a^n b^n$

$L = \{ab, aabb, aaabbb, \ldots\}$.

Σ => Sigma set valid chars of language.

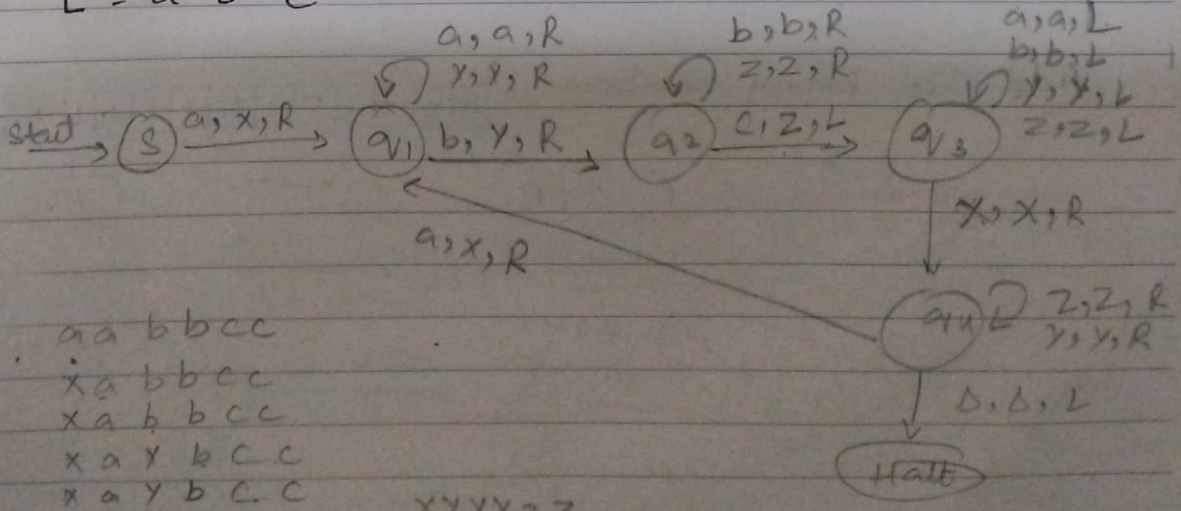
γ => Tape symbol
• halt state.



a b b
X b b
X Y b
X Y b

a a b
X a b
X a b
X a Y
X a Y
X X Y

$L = a^n b^n c^n$



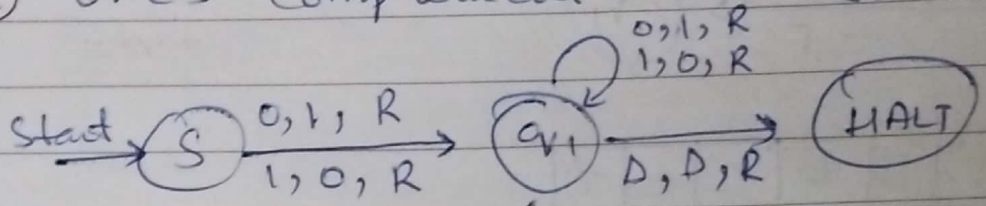
a a b b c c
X a b b c c
X a b b c c
X a Y b C C
X a Y b C C
X a Y b Z C
X a Y b Z C
X X Y b Z C
X X X Y Z C

X X Y Y Z Z

22-05-23

Turing Machine.

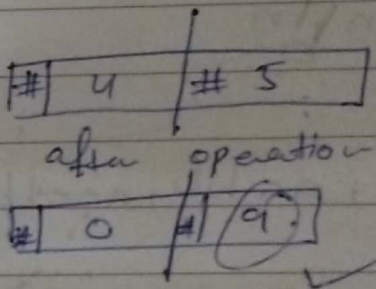
① One's Complement or Toggle Machine.



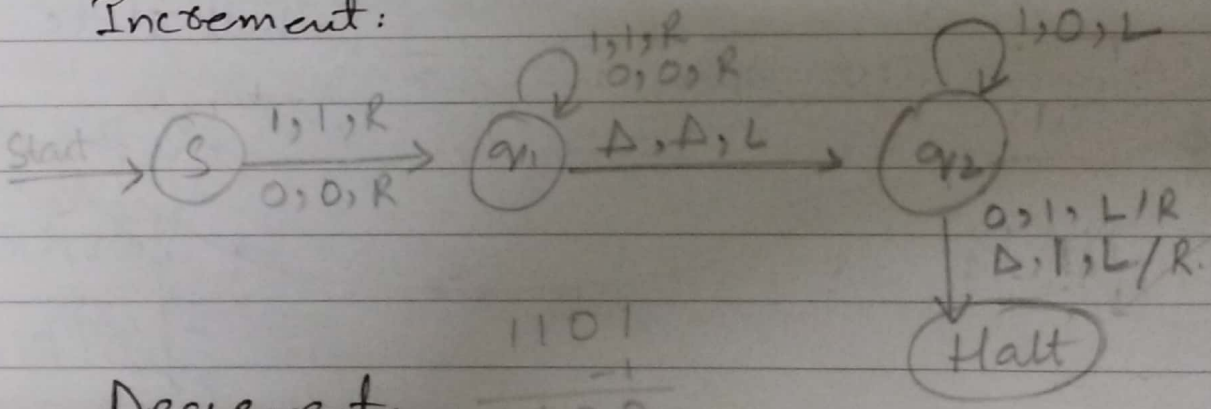
② Add two binary numbers.

4 + 5 = 9.

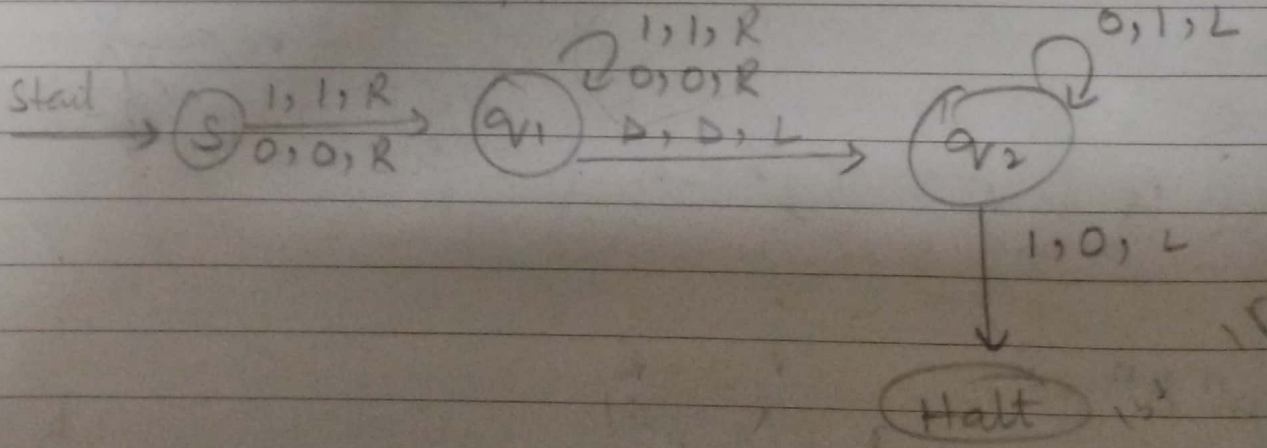
Decrement	Increment
3	6
2	7
1	8
0	9



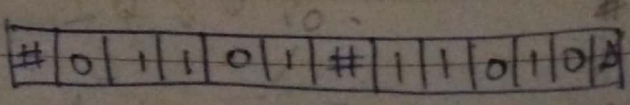
Increment:



Decrement:



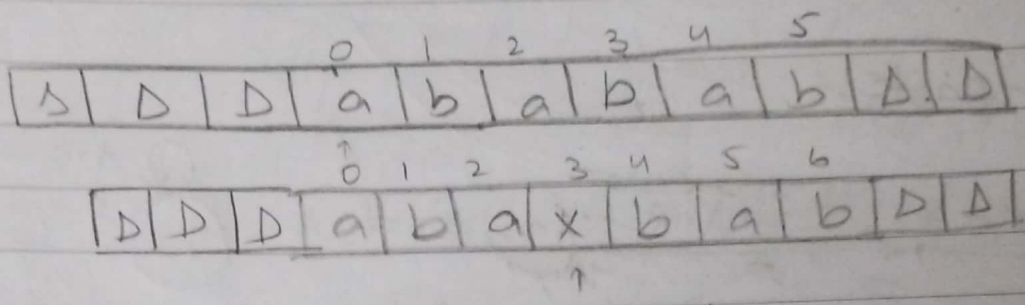
Add two binary numbers.



0,1,R 0,0,L 1,0,L
1,1,L

23-05-23

Insert.



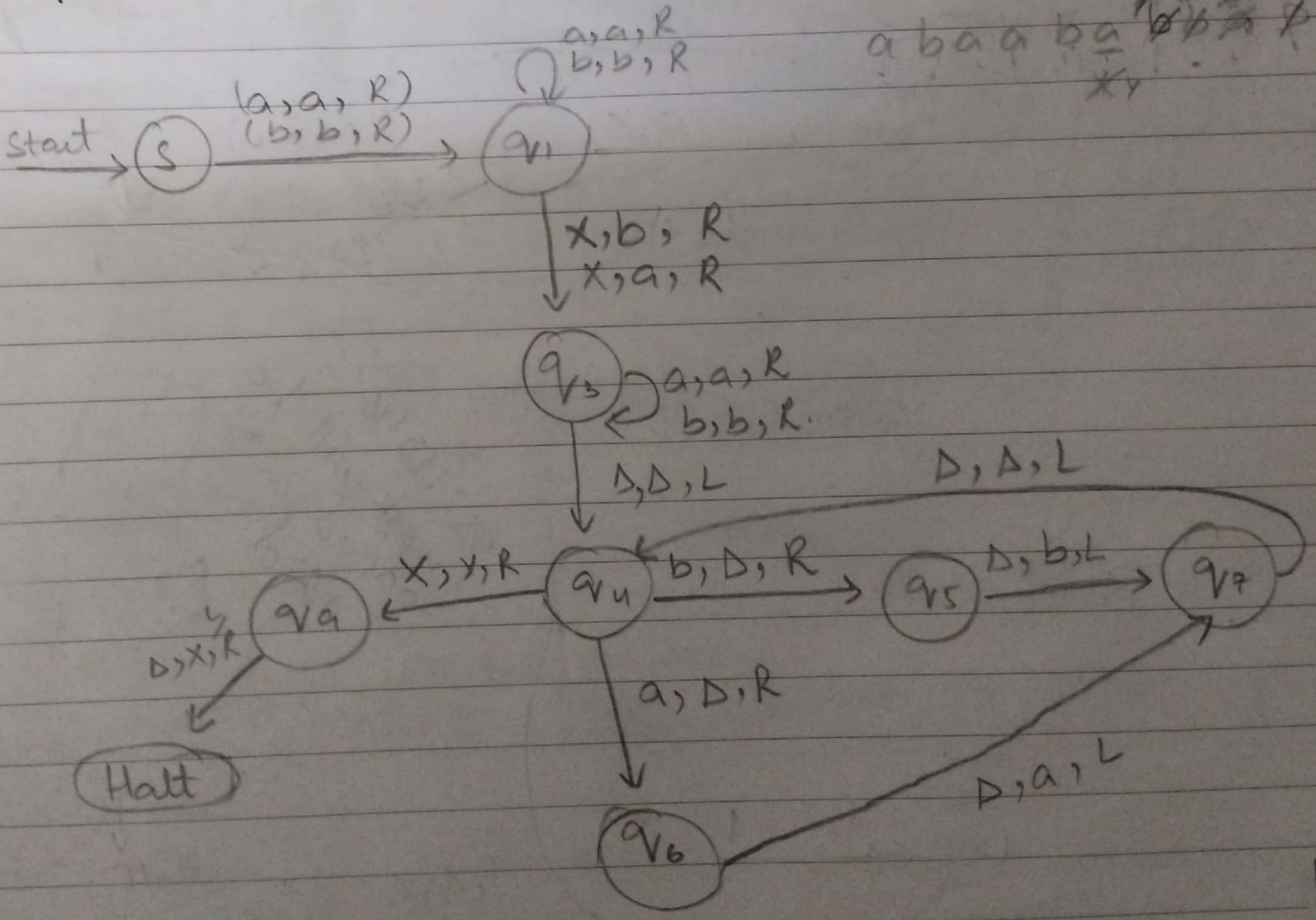
* position
* shift all characters to the right
* The place the new one at posi...

Copy-Paste , Cut-Paste, Delete

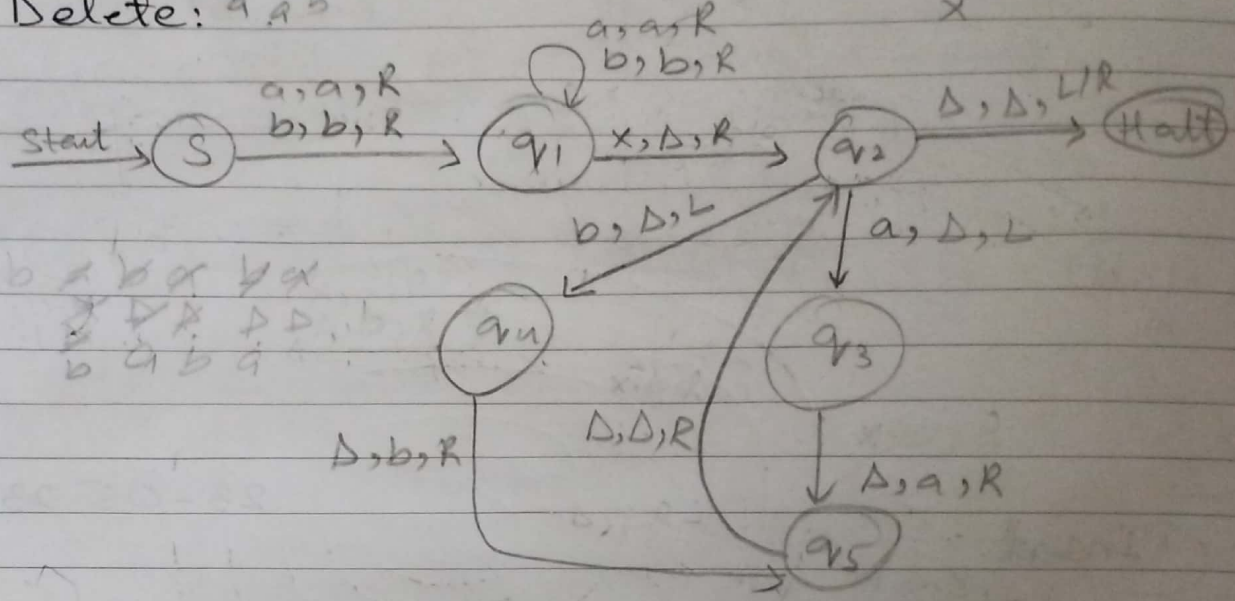
Turing machine to Insert a character at specific position.

x => positio... varia...

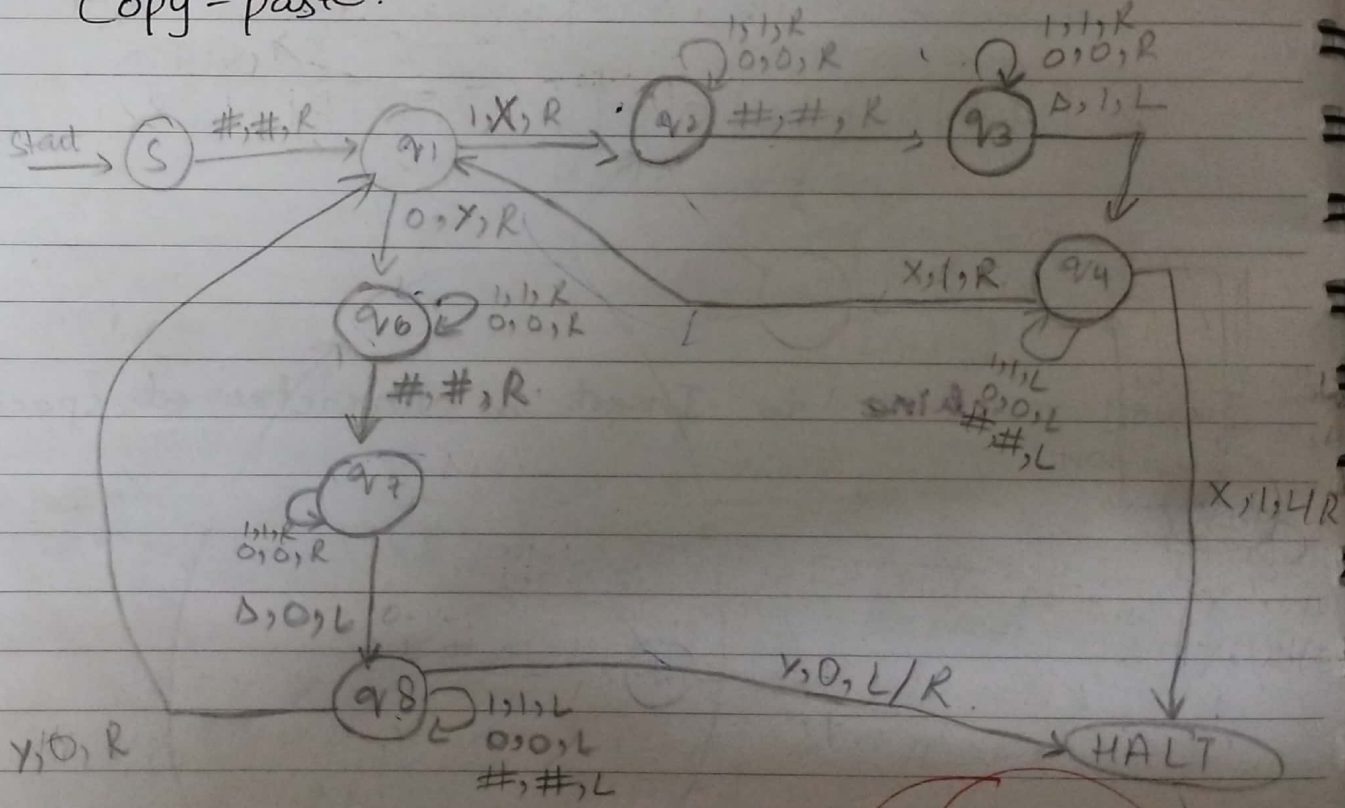
a b a a b a
x



Delete:



Copy-paste.



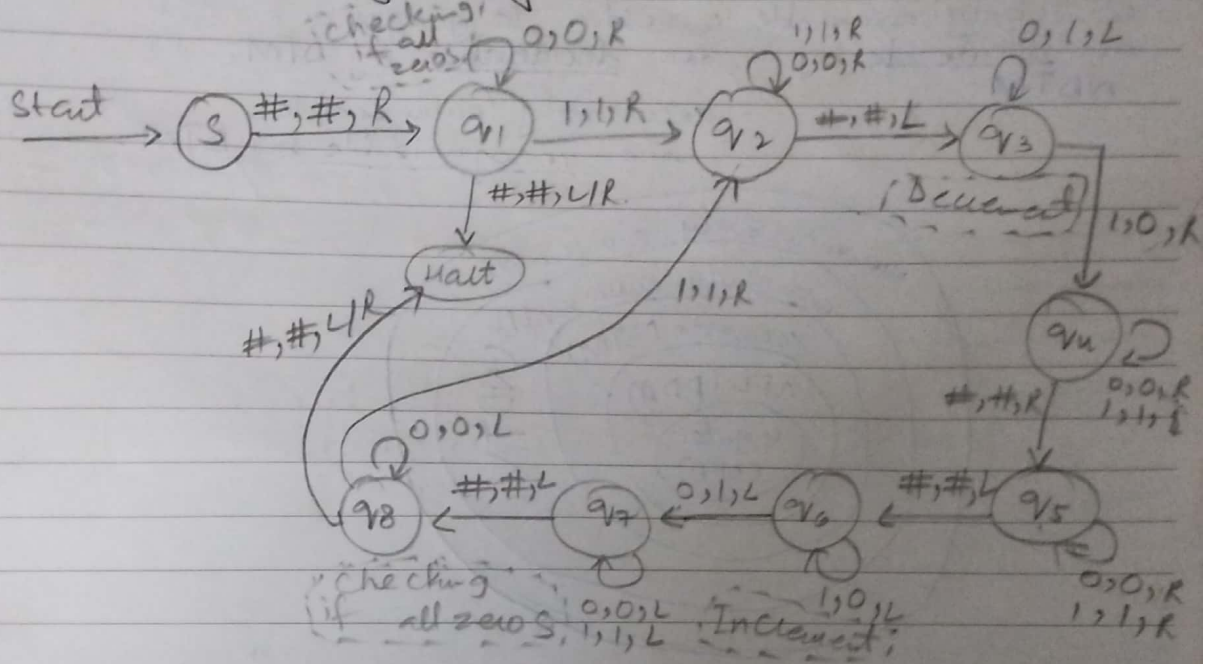
#abbb#ΔΔΔΔ#

#1010#1010#

CS211246 I mark will be added in Mid.

30-05-23

Turing Machine to Add two Binary Numbers without checking overflow condition



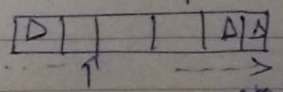
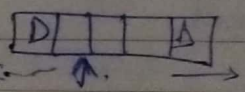
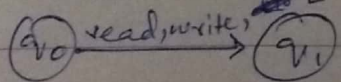
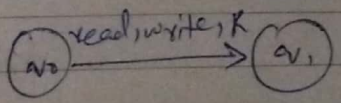
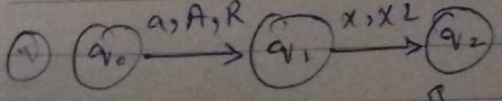
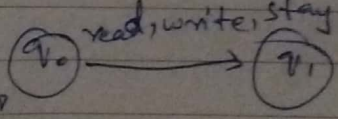
05-06-23

VARIATIONS OF TURING MACHINE

1. Standard T.M. (Already seen).
2. Stay option T.M.
3. Two/Multitape Tape T.M.
4. Multi-track T.M.
5. Semi-infinite T.M
6. Off-line TM.
7. non-deterministic TM.

Information Retrieval. (How search engine works)

Stay Option Machine

Standard Model	Stay Option
1. Input Tape 	Input Tape. 
2. 	" "
3. 	" "
4. 	1. more. 

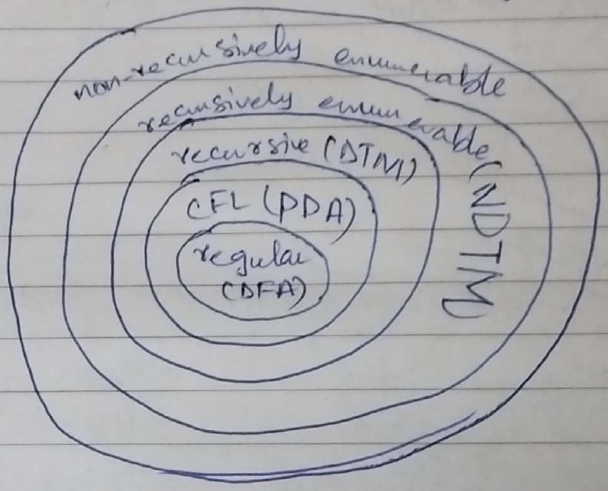
same.

NDTM ≠ DTM (not same power).

↓ recursively enumerable are decideable by NDTM

↓ recursive languages are decideable by DTM.

recursively enumerable are recognizable by DTM.



map

Solvable

NP Problem

Solvable by NDTM in a polynomial time.

NP Non-Deterministic Polinomial Time.

NP-Complete

The problem ones which all other problems can be mapped / reduced

NP-Hard

Can't be solved by NDTM in polynomial time.

Unsolvable

NP Hard

How to check the valid sequence of a language?

int i ; → valid
i int ; → invalid

How to check which one is valid?

→ Write rules to validate it.
→ How to write those rules formally.
We make grammar

Grammar = Set of rules.
rules = variables / constants.
variables → non-terminal (capital letters)
constants → terminal (small letters).
• rules are called productions

Productions
Exactly one non-terminal → terminal / non-terminal / combination of both.

e.g. (read as → can be)
① X → a B
② B → a
③ C → D
④ D → b.

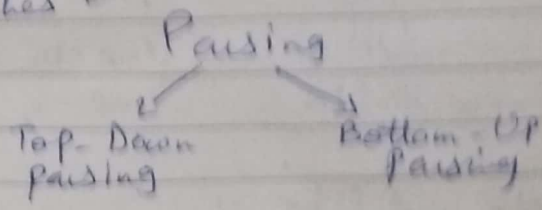
S ⇒ Starting rule.
• In grammar, there can be more than one starting rule.

Example: Grammar G Σ = {a, b}.
① S → aS
② S → bS
③ S → λ

Q) Find whether the string w = w = abab ∈ L(G)?

To check this, w is parsed by Grammar G, if G generate w then we say w ∈ L(G).

Now, how to parse ?
2 approaches ✓

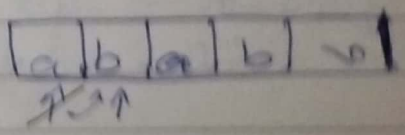


Grammar G

① $S \to aS$
② $S \to bS$
③ $S \to \lambda$
④ $S \to aX$
⑤ $X \to bX\lambda$
⑥ $X \to \lambda$

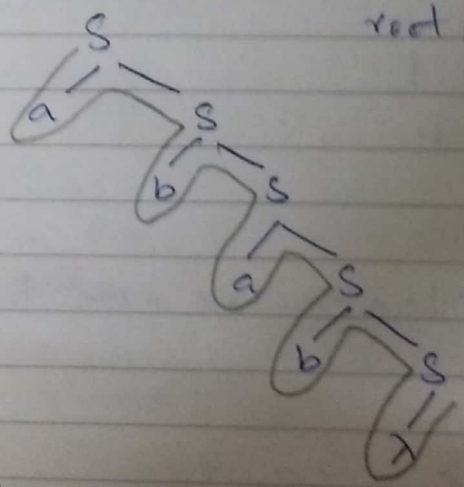
① Top-Down Parsing
↳ Left most derivation
↳ parse tree

Left-most derivation



$S \Rightarrow aS$ (rule 1)
$\Rightarrow abS$ (rule 2)
$\Rightarrow abaS$ (rule 1)
$\Rightarrow ababS$ (rule 1)
$\Rightarrow abab\lambda$ (rule 3)
$\Rightarrow abab$;

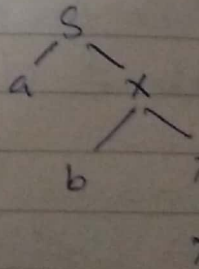
Parse - Tree



root, left, right

a b a b λ
a b a b

Back-tracking a b a b.

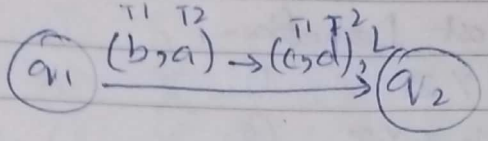


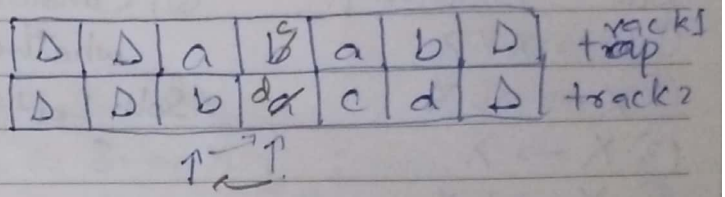
rule 4.

its wrong backtracking

Grammar
$S \to a$
$S \to b$
or
$S \to a/b$.

Multi Track





6-6-23

Grammar:

Regular Grammar

$a^* = \{\lambda, a, aa, aaa, aaaa, \dots\}$.

Grammar:

1. $S \longrightarrow a\ X$ not necessary rule
2. $S \longrightarrow \lambda$ ✓
3. $S \longrightarrow aS$ ✓

Leftmost derivation.

for aaa.

$S \Rightarrow aS$

$\Rightarrow aaS$

$\Rightarrow aaaS$.

$\Rightarrow aaa\lambda$

$\Rightarrow aaa$. ✓

$(ab)^* = \{\lambda, ab, abab, \dots\}$.

Grammar.

① $S \rightarrow abS$

② $S \rightarrow \lambda$.

LMD abab.

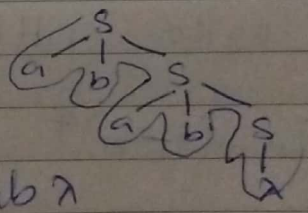
$S \Rightarrow abS$

$\Rightarrow ababS$

$\Rightarrow abab\lambda$

$\Rightarrow abab$. ✓

Parse tree / derivation tree.



abab λ

abab ✓

G_2 $\Sigma = \{a, b\}$.

① $S \to XYZ$
② $X \to abX$
③ $X \to \lambda$
④ $Y \to bY$
⑤ $Y \to a$
⑥ $Z \to bb$
⑦ $X \to aaY$

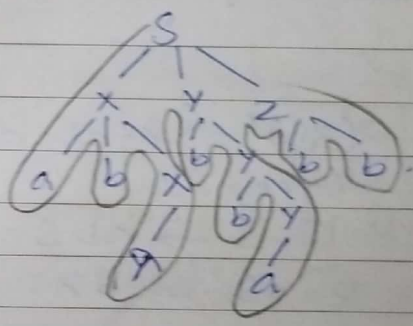
Recent-decent Parsing

Q: Consider the grammar G_2. Find whether $w = abbbabb \in L(G_2)$.

Sol:- Left Most Derivation

$S \Rightarrow XYZ$ ①
$\Rightarrow abXYZ$ ②
$\Rightarrow ab\lambda YZ$ ③
$\Rightarrow abYZ$ –
$\Rightarrow abbYZ$ ④
$\Rightarrow abbbYZ$ ④
$\Rightarrow abbbaZ$ ⑤
$\Rightarrow abbbabb$ ⑥ ✓

Parse Tree:-



* Every regular expression can be converted into grammar but not every grammar can be converted into regular expression.

Fama (a tool) ⇒ input: RegEx output: ~~Grammar~~ DFA.
In python: nltk library.

RegEx: $(a+b)^*\, aa\, (a+b)^*$

Grammar

① $S \to X\, aa\, X$
② $X \to aX$
③ $X \to bX$
④ $X \to \lambda$

RegEx: $(a+b)^+\, aa\, (a+b)^+$

Grammar

① $S \to X\, aaX$
② $X \to aX$
③ $X \to bX$
④ $X \to a$
⑤ $X \to b$.

or $X \to aX \mid bX \mid a \mid b$.
↓ short form
no. of rules are same.

RegEx: $a^+ b^*$

Grammar:

① $S \rightarrow aX$ or $S \rightarrow XY$

② $X \rightarrow aX$ $\left.\right\}$ a^+

③ $X \rightarrow a$ $\left.\right\}$ a^+

④ $Y \rightarrow bY$ $\left.\right\}$ b^*

⑤ $Y \rightarrow \lambda$ $\left.\right\}$ b^*

⑥ $X \rightarrow \lambda$

12-06-23

Ambiguous Grammar

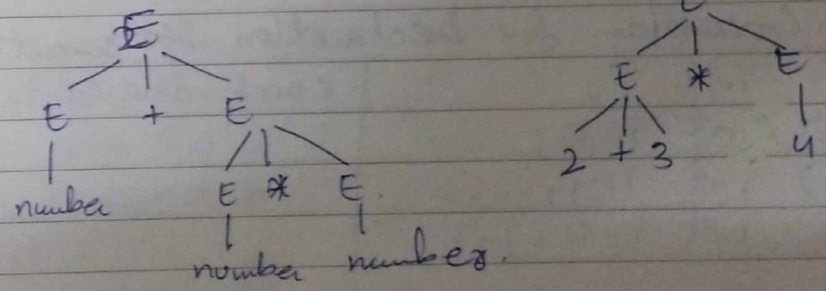
Consider the given grammar.

$E \rightarrow$ number number $\in \mathbb{R}$.

$E \rightarrow E+E$

$E \rightarrow E * E$.

$2 + 3 * 5$.



- a string of a language, if it can be parsed by more than on ways using grammar then this is ambiguous grammar.
- Ambiguity of every grammar is not always resolveable.

Convert into postfix to end ambiguity.

LL Parser
↳ Top down parsing techniques.
↳ Input ko left to right scan krte hain
↳ Leftmost derivation.

Recent - Decent Parser.

① E()
{ E()
match(+)
E()
}

② E()
{ E()
match(*)
E()
}

③ E()
{ return number;
}.

Postfix Solution of Grammar of Previous Questio.

E → number
E → EE+
E → EE*.

Grammar for Declaration Statements of P.L.s.

① int i;
② int i=0;
③ int i, j, ..., k;
④ int i=0, j=1;

<Declaration Statement>.

① <Declaration Statement> → <Datatype><space><ID>;

② <Declaration Statement> → <Datatype><space><ID> = <value>;

Palindromes.

w = reverse (w).

$\Sigma = \{a, b\}$. Ex: abbaabba, babab

Grammar:

$S \longrightarrow a\ S\ a$

$S \longrightarrow b\ S\ b$

$S \rightarrow a$

$S \rightarrow b$

$S \rightarrow \lambda$

① abbbba.

Left-most derivation.

$S \Rightarrow a\ S\ a$.

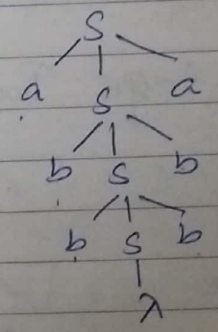
$S \Rightarrow abSba$

$S \Rightarrow abbSbba$.

$S \Rightarrow abb\lambda bba$.

$S \Rightarrow abbbba$.

Parse-Tree.



② babab.

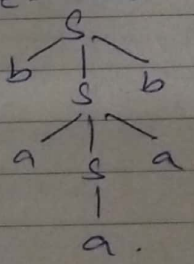
Left-most derivation.

$S \Rightarrow bSb$.

$S \Rightarrow baSab$.

$S \Rightarrow babab$.

Parse-Tree



Equal Equal.

$a^n b^n$

$a^n b^n$

aabb.

Grammar.

$S \rightarrow a\ S\ b$.

$S \rightarrow \lambda$ → if $n \geq 0$.

$S \rightarrow ab$ → if $n > 0$.

n=2

$S \Rightarrow aSb$

$aaSbb$

aabb. S ⇒ ab

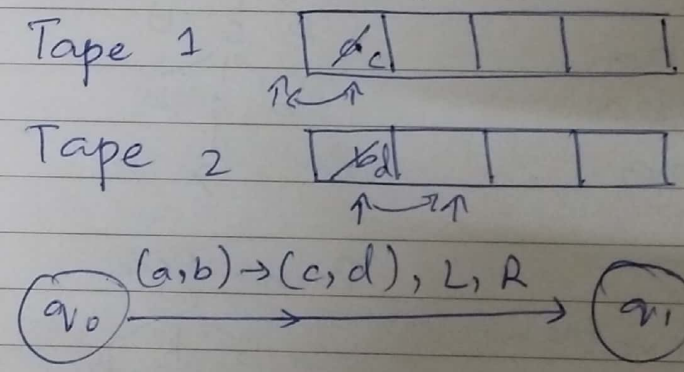
EQual Equal.

Grammar

1. S → a B.
2. S → b A
3. ~~B → aBB~~
4. B → bA
5. A → aB

if else statement ki grammar ambiguous kyun hai?

Two-tape Turing Machine.



Pumping Lemma

→ Is a given language regular or not?
→ Pumping lemma is an algorithm that recognizes that a given language is not regular.
→ 3 properties to know (regular or not), for all infinite strings of a language
→ Hence, if it is not then it'll return not a regular language.

If a language is regular 'L', then there must exist strings $\Rightarrow S \in L$.

$L = \{s_1, s_2, \cdots, s_n\}$.

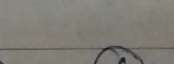
pumping length 'P' (after a unique length)

$S = xyz$ (S can be written as xyz)

$P = 3$; $|S| \geq P$. E.g. $a^* = \{\lambda, a, aa, aaa, aaaa, \cdots\}$.

① $S = x y^{(i)} z$. $(i > 0)$

② $|y| > 0$. ($y \Rightarrow$ loop).



(x before y & z after y).

③ $|xy| \leq P$ (x & y are concatenated).

L is regular if it holds above three conditions.

Examples

① $a^* = \{\lambda, a, aa, aaa, \cdots\}$.

$S = \lambda$ $P = 0$.

$S = xyz$ $\lambda \cdot \lambda \cdot \lambda$

② $S = a$. $S = xyz$. $P = 1$

case no. 1	case no. 2	case no. 3
$x = a$.	$x = \lambda$	$x = \lambda$
$y = \lambda$	$y = a$	$y = \lambda$
$z = \lambda$	$z = \lambda$	$z = a$

$x\ y\ z$

① $S = a \cdot \lambda \cdot \lambda$

② $|xy| = |a \cdot \lambda| = |a|$

③ $|y| \Rightarrow |\lambda|$; $y > 0$ $\quad 1 \leq P$ (True)

(2) $L = a^n b^n (n \geq 0)$ This is not regular.

$L = a^7 b^7$

$S = \underbrace{aa}_{x} \underbrace{aaaaa}_{y} \underbrace{bbbbbbb}_{z} \Rightarrow \in L.$

$S \in L \quad x y^i z \qquad P = 7$

i is a loop variable, agar hum ise multiple times chalayein or resulting string language ka part ho then it is regular.

if $S = x y^2 z.$

$S = \underbrace{aa}_{x} \underbrace{aaaaaa}_{y} \underbrace{aaaaa}_{y} \underbrace{bbbbbbb}_{z}$

- ek string se generate hone wale strings bhi language a part hone chahiyen.

(3) $L = a^* b^* \Rightarrow \{\lambda, a, b, ab, aabb, aaabb, aabbb, \ldots\}$

$S = aaabb$

$x = aa, \quad y = ab, \quad z = b.$ wrong way

$\underbrace{aa}_{x} \underbrace{ab}_{y} \underbrace{b}_{z}$

$aaababb \notin L \quad$ if $i = 2$

we can't take y as different alphabets like a & b.

y should be loop wala either a or b.

so, $S = \underbrace{aa}_{x} \underbrace{a}_{y} \underbrace{bb}_{z}.$

$aaaabb \in L \quad i = 2$ ✓

19-06-23

Insertion, Deletion, Addition
Context free grammar.
Most appropriate machine of a language.
Identify the grammar of a language.
Deriving a string through Parse tree or left-most derivation

Palindrome
Equal Equal.

Equal a's & b's.

Grammar

abbbba

① $S \rightarrow SASBS$
② $S \rightarrow SBSAS$
③ $S \rightarrow \lambda$

Halting Problem (Undecideable Problem)

M ⟶ machine
W ⟶ string.

```
f(m, w)
{
    m()
    {return Y/N}
}
```

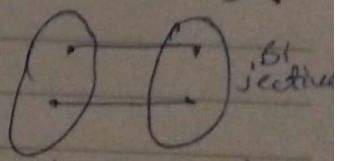
Question: Does M halt on W ?

Universal Turing Machine
↳ It can take a machine as an input.

- Every T.M can be represented as a binary number but not vice versa.

- Binary numbers set = B (superset)
- Turing machines set = T. (subset.)

$B \supset T$



Countably Infinite Set.
Is a set, jis ke corresponding ek bijective function likh sakein.
→ Will take an element as input & assig a unique integer to every element