Theory of Computation

Date / /

Branch of theoritical CS that deals with study of algorithms & computational complexity. It aims to ask →

→ What can be computed
→ How efficiently
→ What limitations exist in terms of computational power.

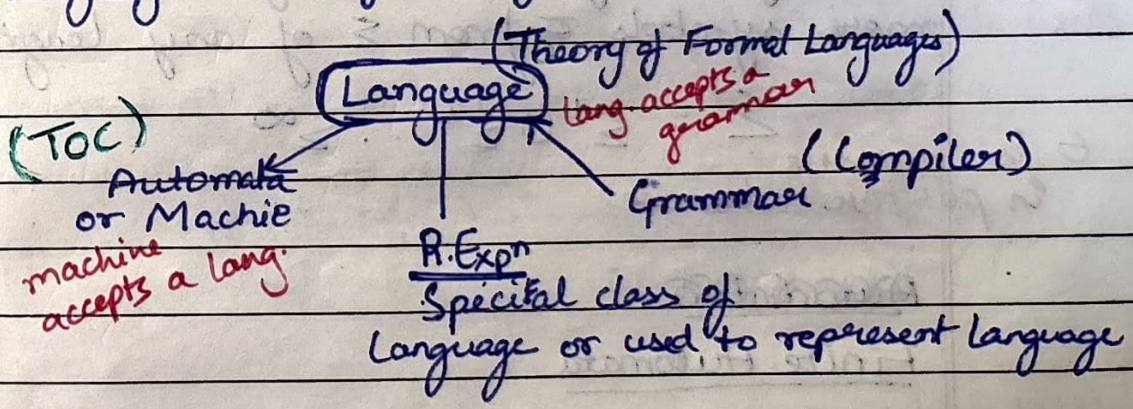
Basic Concepts & Automata Theory

Symbols ÷ Basic building blocks which can be character or token. Ex→ a, b etc

Alphabet ÷ Finite, non-empty set of symbols. Ex→ $\Sigma = \{a, b\}$ set

String ÷ Finite sequence of symbols (which are member of set alphabet) Ex→ aabb, ab, bab etc

Language ÷ Set of strings Ex→ {aabb, ab} set



Studying language is difficult, so we can either study Machine (TOC) or Grammar. (Compilers)

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

$\Sigma^0 = \{\epsilon\}$ → Epsilon = a string with length = 0

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{a, b\} \cdot \{a, b\} = \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

Date / /

Σ^K = Set of all strings from alphabet Σ of length exactly K.

Kleene Closure (Σ^*)

Set of all strings obtained by concatenating 0 or more symbols from Σ of any length.

$$\Sigma^* = \bigcup_{i=0}^{i=\infty} \{w \mid |w| = i\}$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots \Sigma^\infty$$

Ex → $\Sigma = \{a, b\}$

$$\Sigma^* = \{\epsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \cup \ldots \infty$$

Note → Language will be subset of Σ^*

Positive Closure (Σ^+)

Set of strings obtained by concatenating 1 or more symbols from Σ of any length.

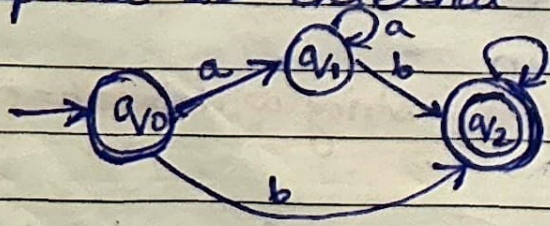
$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \ldots \Sigma^\infty$$

ϵ is missing in positive closure

Finite Automata

Abstract Machine

Model that has finite set of states and its control moves from one state to another in response to external inputs.



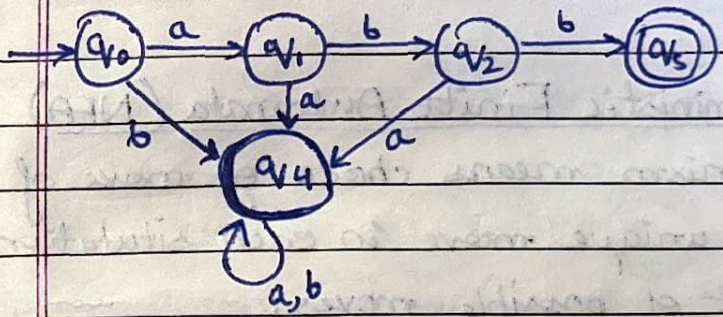
2 Types of FA → without O/P → DFA, NDFA

→ with O/P → Moore, Mealy

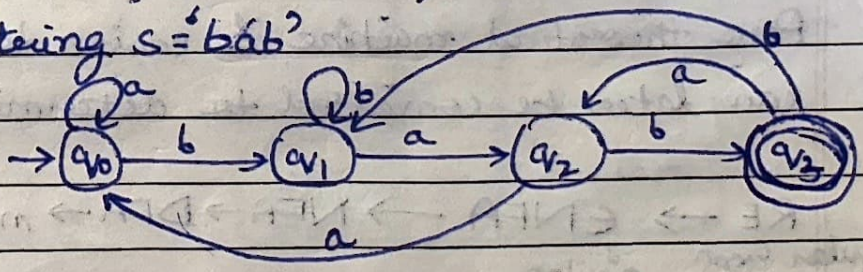
Date / /

→ Deterministic Finite Automata (DFA)

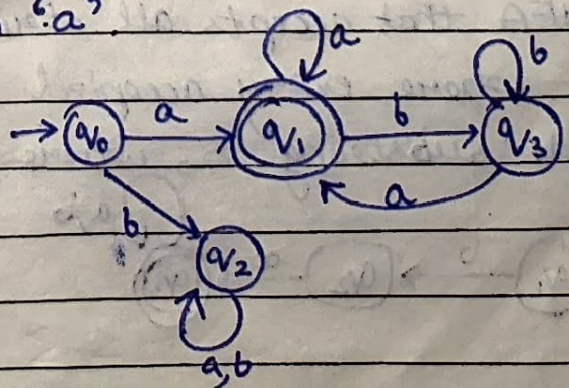
Q Design DFA that accepts all strings over alphabet $\Sigma = \{a, b\}$, where every accepted string 'ω' starts with substring s = 'abb'.



Q Design DFA over $\Sigma = \{a, b\}$ where 'ω' ends with substring s = 'bab'



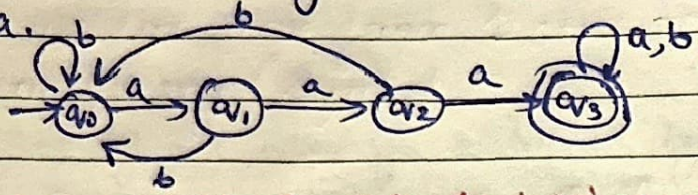
Q Design DFA over $\Sigma = \{a, b\}$ where 'ω' starts & ends with 'a'.



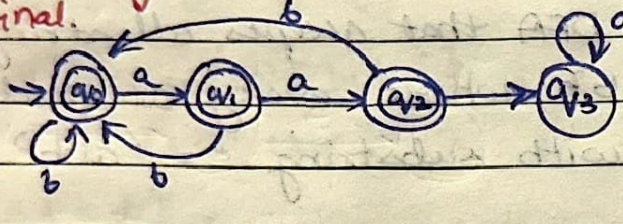
Date / /

Complement of DFA

Q Accept all strings that doesn't accept substring aaa.



Convert all final state to ~~non~~ normal & normal to final.



→ **Non Deterministic Finite Automata (NFA)**

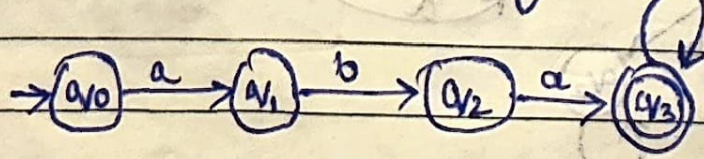
Non determinism means choice of move of automa
Rather than unique move in each situtation we allow a set of possible moves.

Are theoritical machine & easier to design than can later be converted to deterministic machine.

RE → ∈NFA → NFA → DFA → minimal DFA

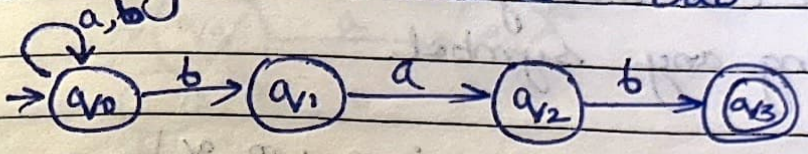
Regular Expn Epsilon

Q Design a NFA that accepts all strings over alph $\Sigma = \{a, b\}$, where every accepted string 'w' starts with substring 's', where s = 'aba'



Date / /

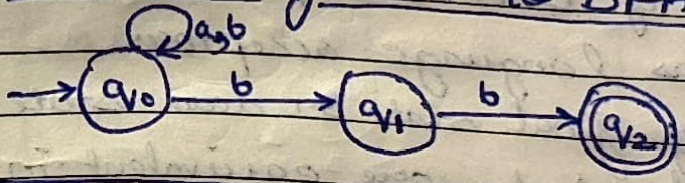
Q Design NFA that accepts all strings over $\Sigma = \{a, b\}$ where every accepted string 'w' ends with substring 's' where s = 'bab'



Note → Every DFA is also an NFA.

Note → Acceptance in NFA just means that there exists at least 1 transition for which we start at initial state & ends in any one of the final state, then string 'w' is accepted by NFA.

Conversion of NFA to DFA



	a	b
→ q_0	q_0	q_0, q_1
q_1	ϕ	q_2
(q_2)	ϕ	ϕ

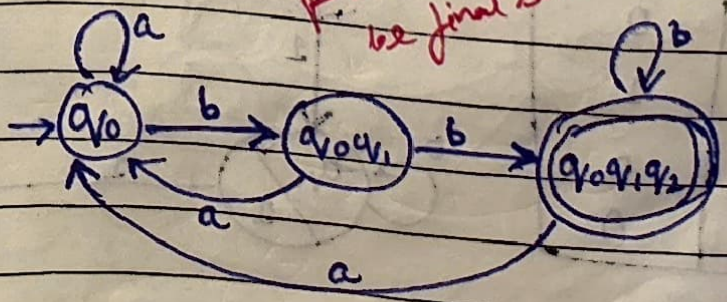
	a	b
q_0	q_0	$\{q_0, q_1\}$
$\{q_0, q_1\}$	q_0	$\{q_0, q_1, q_2\}$
($\{q_0, q_1, q_2\}$)	q_0	$\{q_0, q_1, q_2\}$

write line 1 as it is & treat mult. as a set.

$\{q_0, q_1\}$ state 'a' par kha kha gaya = $q_0 \cup \phi = q_0$

Whereever q_2 is present that will be final state

$\{q_0, q_1\}$ state 'b' par kha kha gya $q_0, q_1 \cup q_2$ $q_0 q_1 q_2$

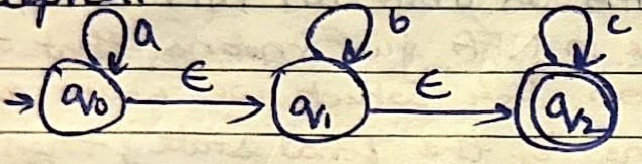


Date / /

→ Epsilon NFA (ϵNFA)

Automata that consists of null transitions.
Or we can move from 1 state to another without consuming any symbol.

Ex→ Design where $\{a^n, b^m c^q \mid n,m,q \geq 0\}$ is accepted.



→ Moore & Mealy Machine

⇒ Special cases of DFA that produce output rather than acting as language acceptors.
⇒ No concept of final state or dead state.
⇒ Both Moore & Mealy are equivalent in power.

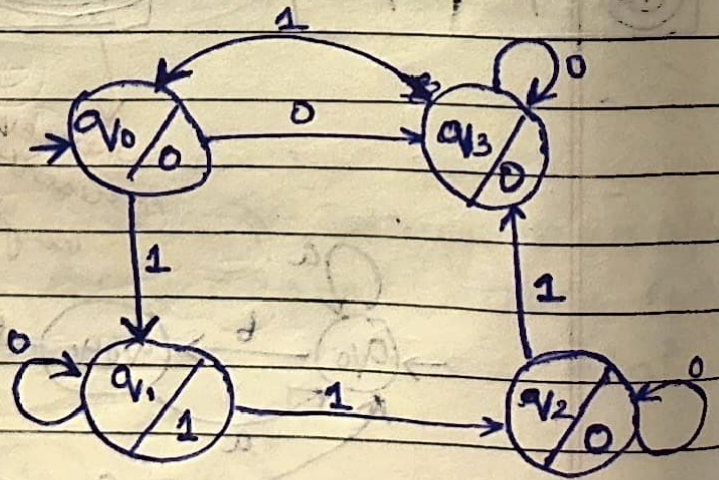
Moore Machine

Output is associated with state.

Ex→

Transition table

Present State	Next State a=0	Next State a=1	O/P or λ
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0



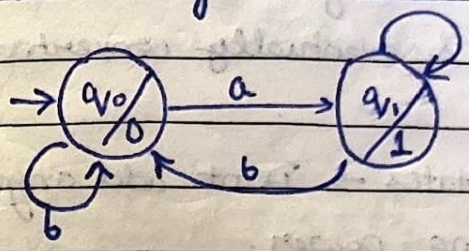
Suppose I/P given to this machine is **01110**
then O/P will be → $q_0$01110

q_0 pr 0 aane pr goes to q_3 (and q_3 ka O/P is 0)

↓↓↓↓↓
0 00100

Date / /

Q Construct a Moore machine that takes a's & b's as I/P and count no. of a's in I/P string in terms of 1's.



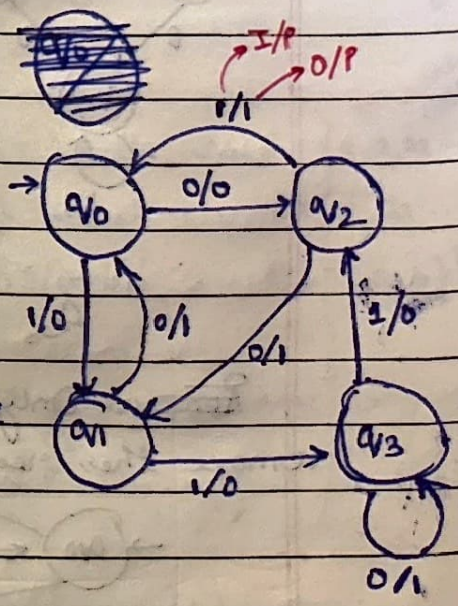
Mealy Machine

Output is associated with transition.

Ex:-

Transition Table

Present State	Next State			
	a=0		a=1	
	State	O/P	State	O/P
→ q_0	q_2	0	q_1	0
q_1	q_0	1	q_3	0
q_2	q_1	1	q_0	1
q_3	q_3	1	q_2	0

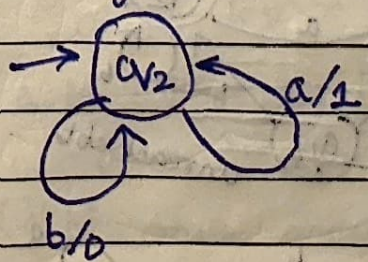


Suppose I/P given to this machine is 01110 then O/P is

0 1 1 1 0

0 1 0 0 1

Q Construct a Mealy machine that takes all strings of a's & b's as I/P & counts no. of a's in I/P string in terms of 1's.



Date / /

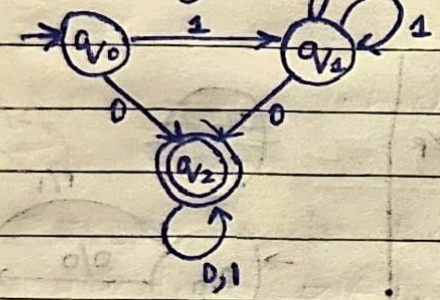
Minimization of Finite Automata

Only DFA are minimized because NFA & ε-NFA are anyways conceptual & eventually converted to DFA.

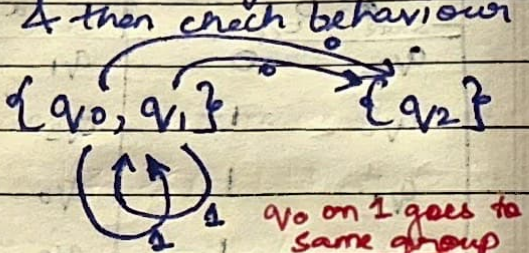
Non Productive States → Dont add anything to language accepting power.

DEAD State, Unreachable State & Equal State

Ex: Minimize the following →



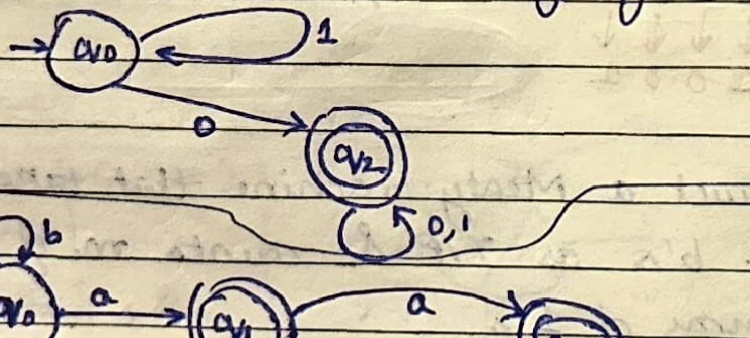
Group all non-final states in 1 set & final in another & then check behaviour



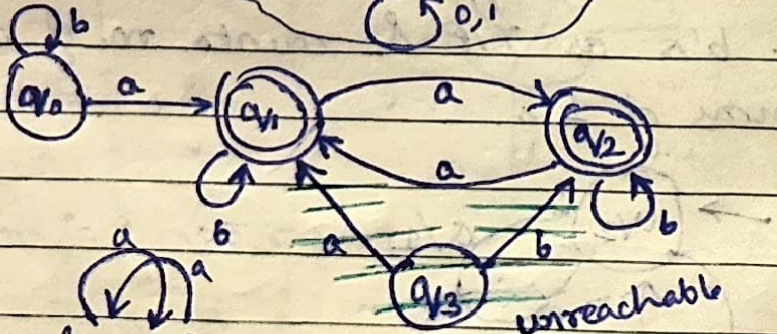
q0 on 1 goes to same group

As Behaviour is same →

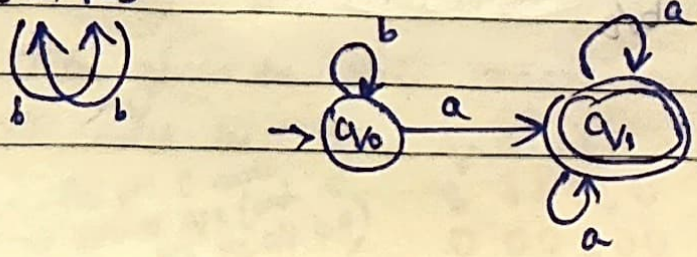
Remove Only keep 1 state from the group & remove the rest & their outgoing transitions.



Ex2



{q0} {q1, q2}



Date / /

Regular Expressions

An expression of strings which represent regular language is called Regular Expression.

A regular exp^n is valid iff it can be derived from primitive regular exp^n by a finite no. of applications of operator.

Any terminal symbol (Σ)
ϵ & ϕ

Regular Language :- Any set (language) represented by a regular exp^n.

Ex → If $a, b \in \Sigma$ then

$R = a$ denotes $L = \{a\}$
$R = a.b$ " $L = \{ab\}$ (concat)
$R = a+b$ " $L = \{a, b\}$ (union)
$R = a^*$ denotes $L = \{\epsilon, a, aa \dots\}$ (Kleent closure)
$R = a^+$ " $L = \{a, aa, \dots\}$
$R = (a+b)^*$ " $L = \{a, b\}^*$

Note → 2 Regex are equal if they represent same language.

Ex → $r_1 = a^*$ $r_2 = a^* + (aa)^*$

$r_1 = r_2$

Q Design Regex that represent language 'L'

① → $L = \{a\}$ over $\Sigma = \{a\}$

a

② → $\Sigma = \{a, b\}$ where accepted string 'w' starts with $s =$ 'abb'

$abb\,(a+b)^*$

③ → $\Sigma = \{a, b\}$ where accepted string 'w' ends with $s =$ 'bab'

$(a+b)^*\,bab$

④ → $\Sigma = \{a, b\}$ where accepted string 'w' contains substring $s =$ 'aba'

$(a+b)^*\,aba\,(a+b)^*$

Date / /

⊙→ $\Sigma = \{a, b\}$ such that accepted string 'ω' starts & ends with a

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

⊙→ $\Sigma = \{a, b\}$ such that accepted string 'ω' starts & ends with same symbol

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

Pumping Lemma for Regular Languages

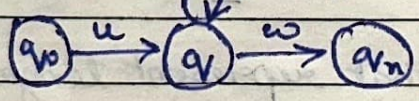
Used as a proof for irregularity of a language.
If there exists atleast 1 string made from this lemma, which is not in L, then L is not Regular

For any Regular Language L, there exists integer n such that $z \in L$ and $|z| \geq n$, then exist $u, v, w \in \Sigma^*$, such that $z = uvw$ and

$|uv| \leq n$

$|v| \geq 1$

for all $i \geq 0 : uv^iw \in L$



string v ko pump karne k baad bhi string remains in L.

Q $L = \{a^m b^n \mid m = n\}$ is non-regular?

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

$z \in L \quad z = a^k b^k = \underbrace{a^{k-1}}_{u}\ \underbrace{a}_{v}\ \underbrace{b^k}_{w}$

For $i=1 \rightarrow uv^iw \rightarrow a^{k-1}ab^k$

$i=2 \rightarrow uv^iw \rightarrow a^{k-1}a^2b^k$

$= a^{k+1}b^k$

= does not belong to Language.

Date / /

Regular and Non-Regular Grammar

Formal Grammar

Grammar is 4 tuple (V_n, Σ, P, S) — V_n: Variables, Σ: terminals, P: Production Rules, S: Start Symbol

V_n = Finite non-empty set whose elements are called variables.

Σ = Finite non-empty set whose elements are terminals.

S = Start Symbol

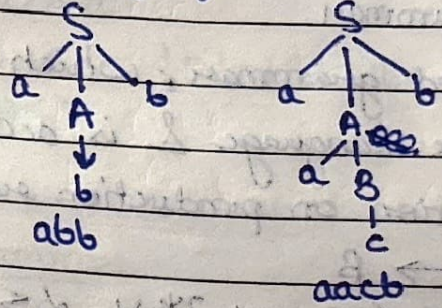
P = Finite set whose elements are $\alpha \to \beta$. α has atleast 1 symbol from V_n & $\beta \in \{\Sigma \cup V_n\}^*$

Q Consider the following grammar & identify its language

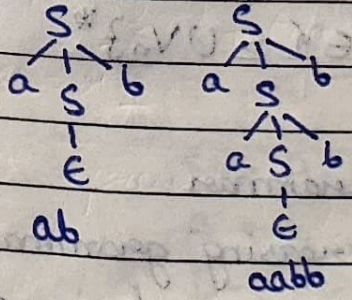
→ $S \to aAb$

$A \to aB/b$

$B \to c$



→ $S \to aSB/\epsilon$

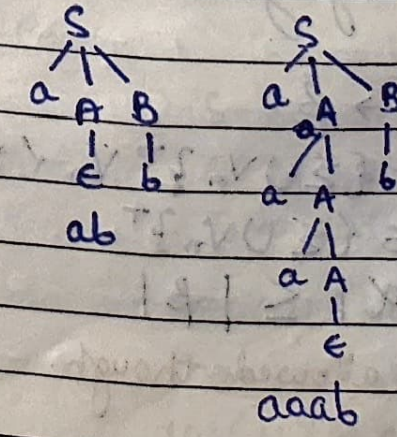


$L = \{a^n b^n \mid n \geq 0\}$

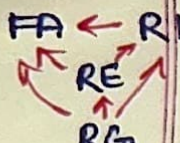
→ $S \to aAB$

$A \to aA/\epsilon$

$B \to b$

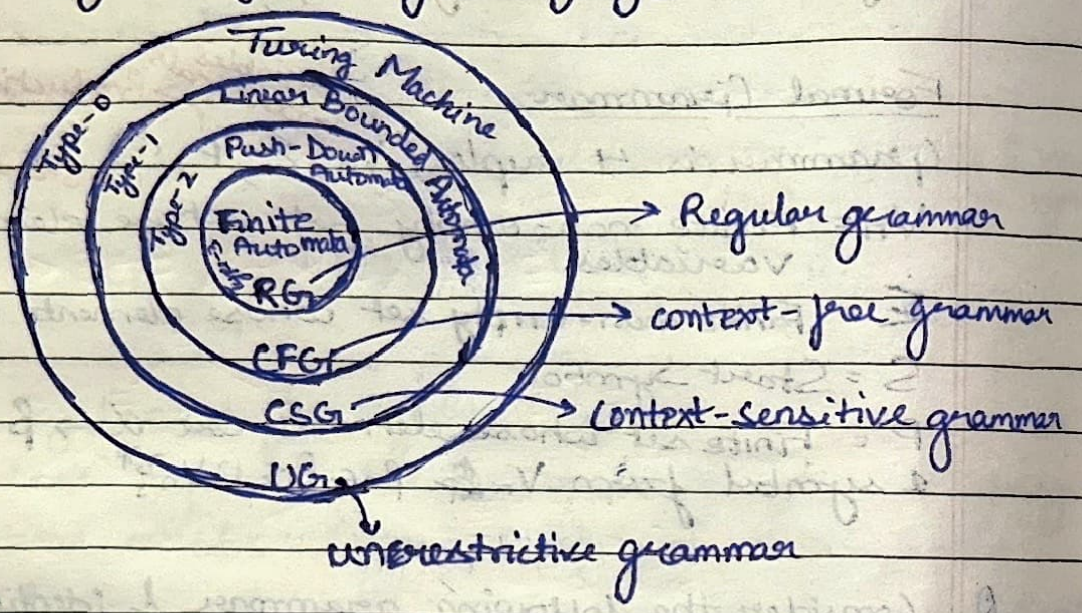


$L = \{a^n b \mid n \geq 1\}$



Date / /

Chomsky Classification of Languages



→ Type-0 Grammar

Unrestricted grammar, which generates recursively enumerable language L is accepted by Turing M/C.

No restriction on production rule.

$$\alpha \rightarrow \beta$$

$$\alpha \in \{\Sigma \cup V_n\}^* V_n \{\Sigma \cup V_n\}^*$$

$$\beta \in \{\Sigma \cup V_n\}^*$$

→ Type-1 Grammar

Length increasing grammar that generates context sensitive language L is accepted by Linear Bounded automata.

$$\alpha \rightarrow \beta$$

$$\alpha \in \{\Sigma \cup V_n\}^* V_n \{\Sigma \cup V_n\}^*$$

$$\beta \in \{\Sigma \cup V_n\}^+$$

$$|\alpha| \leq |\beta|$$

$S \rightarrow \epsilon$ is allowed though.

Date / /

→ Type-2 Grammar

Context Free Grammar that generates Context Free Language L is accepted by Push-Down Automata.

$$\alpha \rightarrow \beta$$
$$\alpha \in V_n \quad |\alpha| = 1$$
$$\beta \in \{\Sigma \cup V_n\}^*$$

→ Type-3 Grammar

Used to generate Regular Language which is accepted by Finite Automata.

Can be of 2 types Left Linear or Right Linear.

Left Linear	Right Linear
$A \rightarrow a/Ba$	$A \rightarrow a/aB$
$A, B \in V_n \quad \lvert A\rvert = \lvert B\rvert = 1$	$A, B \in V_n \quad \lvert A\rvert = \lvert B\rvert = 1$
$a \in \Sigma^*$	$a \in \Sigma^*$

Regex to Grammar

Q $\underset{A}{abb}\ \underset{B}{(a+b)^*}$

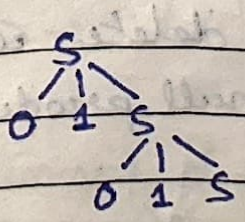
$S \rightarrow AB$
$A \rightarrow abb$
$B \rightarrow aB/bB/\epsilon$

Q $\underset{A}{a} + \underset{A}{a}\underset{B}{(a+b)^*}\underset{A}{a}$

$S \rightarrow A/ABA$
$A \rightarrow a$
$B \rightarrow aB/bB/\epsilon$

Grammar to Regex

Q $S \rightarrow 01S/01$



$(01)^+$

Q $S \rightarrow 011A/101B$

$A \rightarrow 110A/00 \rightarrow (110)^*00$

$B \rightarrow 11B/S \rightarrow (11)^*S$

$011(110)^*00/101(11)^*S$

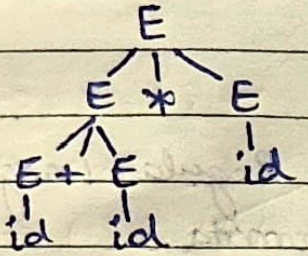
S replaced in this

$101(11)^*011(110)^*00$

Date / /

Derivation - Process of deriving a string.
Syntax/Parse Tree - Graphical representation of derivation.

Ex $E \to E+E / E*E / E=E / id$

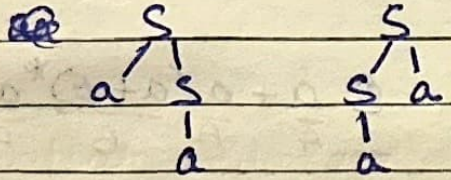


Sentential Form
E
E*E
E+E*id
E+id*id
id+id*id

← Right most derivation

Ambiguous Grammar
CFG is ambiguous if there are more than 1 derivation tree for any string.

Ex $S \to aS / Sa / a$



Note → Grammar which is both left & right recursive is always ambiguous.

Minimization of CFG
To make it more efficient & compiler friendly we delete useless symbol, unit production & null production.

→ Null production ($A \to \epsilon$)

Date / /

Ex 1 $S \to AbB$
$A \to a/\epsilon$
$B \to b/\epsilon$

If we remove $B \to \epsilon$, then what string can be produced with $B \to \epsilon$, add that to S.

e.g. $S \to AbB = Ab$ so seperately add Ab & repeat

$S \to AbB/Ab$
$A \to a/\epsilon$
$B \to b$

$S \to AbB/Ab/bB/b$

Ex 2 $S \to AB/A/B/\epsilon$
$A \to a/\epsilon$
$B \to b/\epsilon$

$S' \to SUE$ → created S' & put epsilon there if lang is such where epsilon is necessity
$S \to AB/A/B$
$A \to a$
$B \to b$

→ Removal of Unit Prod".

Ex 1 $S \to Aa$
$A \to a/Bd$
$B \to d$

Ex 2 $S \to aAb$
$A \to B/a/b/c/d$ ⇒ $S \to aAb$, $A \to a/b/c/d$
$B \to \epsilon/b/c/d$
$C \to d/c$
$D \to d$

→ Removal of Useless Symbol

Ex 1 $S \to aAB$
$A \to a$
$B \to b$
$C \to d$ → not reachable

Ex 2 → $S \to aA/aB$
$A \to b$

useless as B is not present

Compiler Friendly Grammar should only have 1 terminal in 1 rule

Chomsky Normal Form

Simplifying CFGs to make them easier to work with or compiler friendly.

$A \rightarrow BC / a$

$BC \in V_n$ — Either single terminal or 2 non terminals

$a \in \Sigma$

Ex $S \rightarrow aSb / ab$

Lets say we have $A \rightarrow a$ & $B \rightarrow b$

$S \rightarrow ASB / AB$ → still not allowed

$A \rightarrow a$

$B \rightarrow b$

Lets say we have $S' \rightarrow AS$

$S \rightarrow S'B / AB$

$S \rightarrow AS$

$A \rightarrow a$

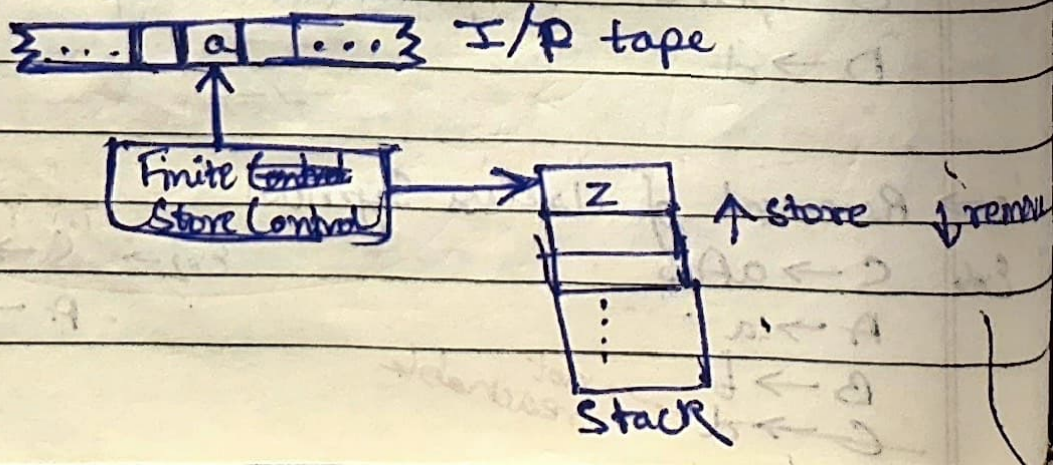
$B \rightarrow b$

} Chomsky Normal Form

Pushdown Automata

Finite automata has limitations, it cannot do infinite comparison between symbols.

Ex: $L = \{a^n b^n \mid n >= 1\}$. This is not regular but can be solved using a auxillary memory, a stack.

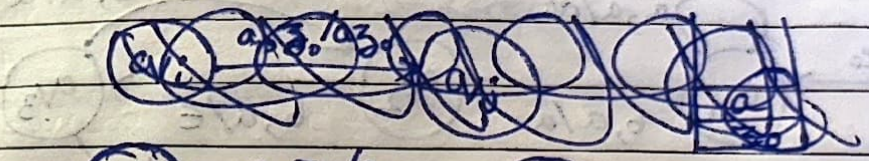


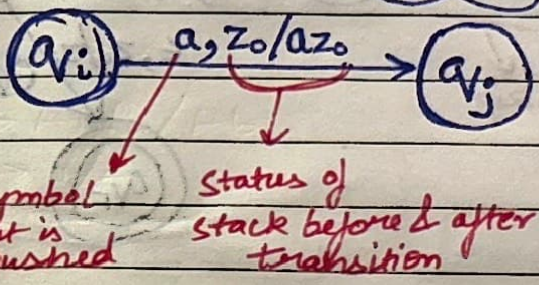
Representations

Date / /

→ PUSH

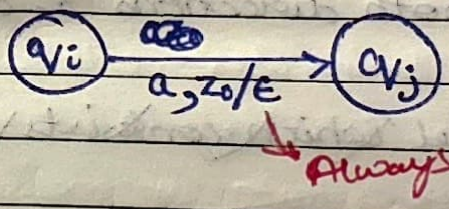
$\delta(q_i, a, z_0) = (q_j, az_0)$





→ POP

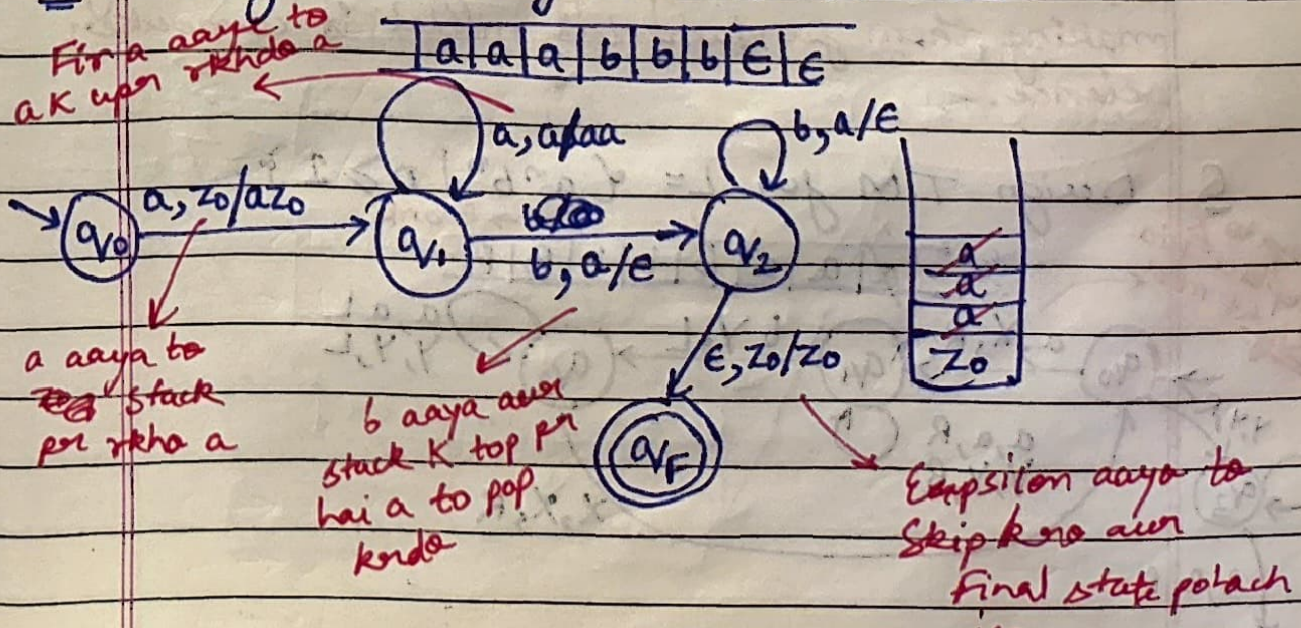
$\delta(q_i, a, z_0) = (q_j, \epsilon)$



→ SKIP

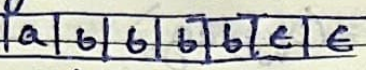
$\delta(q_i, a, z_0) = (q_j, z_0)$

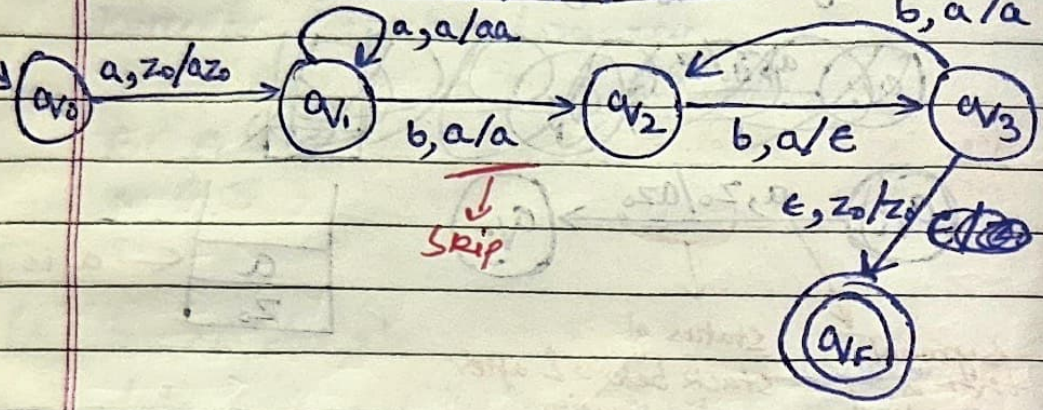
Q Design a PDA for $\{a^n b^n \mid n \geq 1\}$



Date / /

Q Design PDA for $\{a^n b^{2n} \mid n \geq 1\}$





Turing Machine

Infinite size tape & it is used to accept Recursively Enumerable Languages.

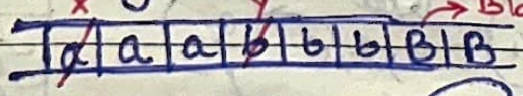
It can move in both directions & doesn't accept ∈.

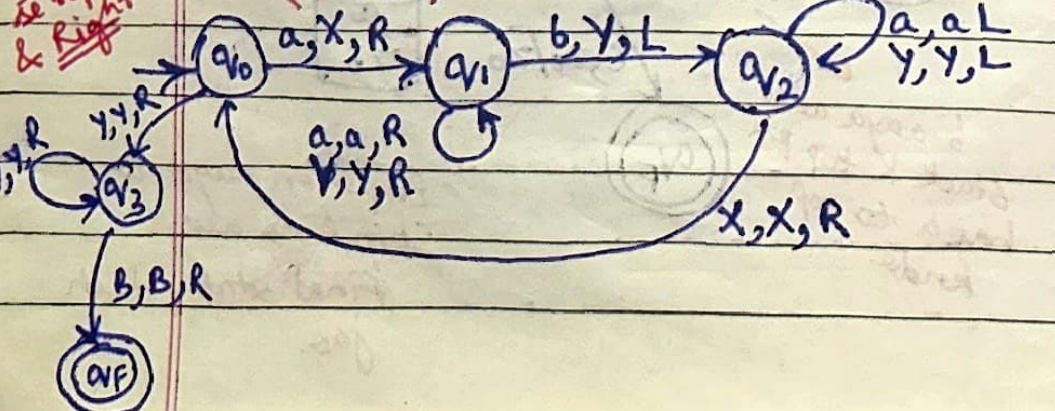
~~Mathematical model which consists of infinite length tape~~

Theoritical / Abstract model of computation introduced in 1936 that understands the limits of what can be computed.

Note → Can simulate logic of any computer algorithm making them fundamental model in computer science.

Q Design T.M for $L = \{a^n b^n \mid n \geq 1\}$





Date / /

→ Decidable Language

Recursive Language → T.M will always Halt.

Recursively Enumerable Language → TM will halt sometimes.

→ Partially decidable language

Undecidable lang. → No TM exist.