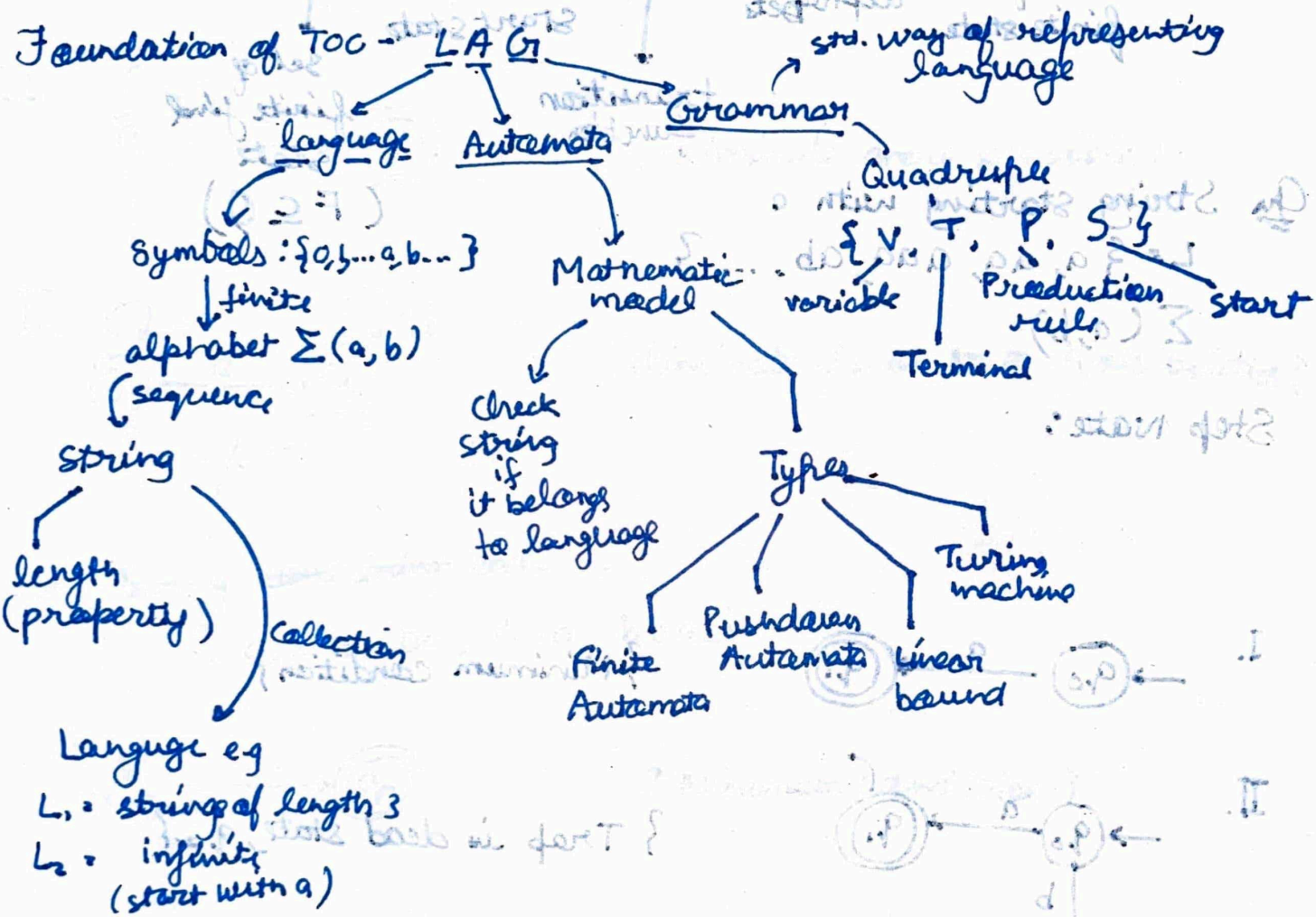
Theory of Computation

Syllabus:

* Regular language and finite automata (80%)
* Context free language and push down automata (10%)
* REC, RE and turing machine (10%)
* Closure properties and undecidability



Power of Σ

Σ^0 = set of string with length 0
= λ, ε (Null string)

⋮

Σ^* (Kleen Closure)
set of all string with all length possible

$(a+b)^*$ } Infinite lang }

Σ^+ (positive closure)
= $\Sigma^* - \varepsilon$

DETERMINISTIC FINITE AUTOMATA (DFA):

One input, per one state

TUPLES: $(Q, \Sigma, \delta, q_0, F)$

- Q: set of finite state
- Σ: set of alphabets
- δ: transition function
- q_0: start state
- F: set of finite final state ($F \subseteq Q$)

Q String starting with a

$L = \{a, aa, aaa, ab, \ldots\}$

$\Sigma (a, b)$

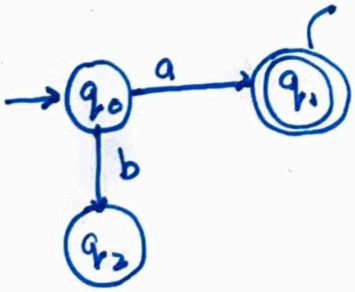
Step Note:

> Always, first design the automata for minimum string

I.

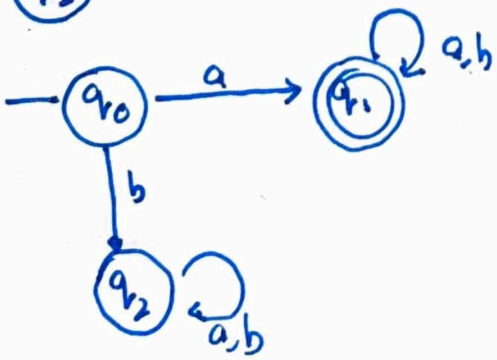
} Minimum condition }

II.



} Trap is dead state, q_2 }

III



Transition

$\delta : Q \times \Sigma \longrightarrow Q$

How To Construct DFA:

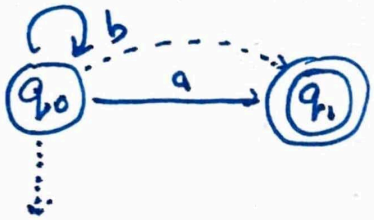
Qu String containing 'a' ; $\Sigma(a,b)$

$L = \{a, aa, aaa, ba, ba, aba \ldots.\}$

I.

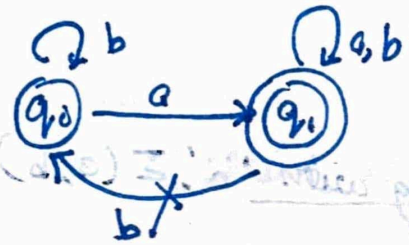
} Minimum condition }

II.



} eliminate every possible }

III.



} eliminate final state possibilities }

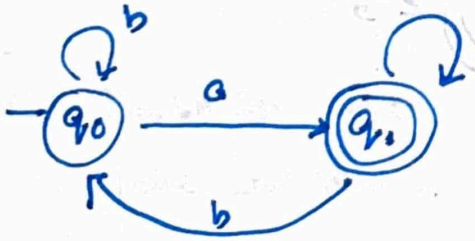
Qu String ~~containing~~ end with 'a' , $\Sigma(a,b)$

$L = \{a, aa, ba, aba, bba \ldots\}$

I.

} Minimum condition }

II



For multiple symbol use loop, otherwise transition

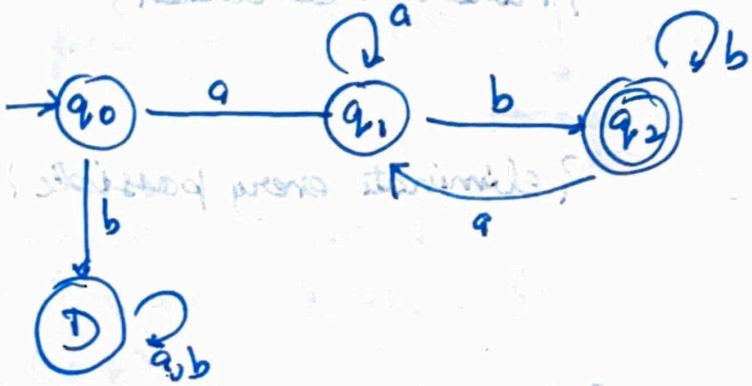
Que Starting 'a' and ending with 'b' ; $\Sigma(a,b)$

L = {ab, abab, abbb, aaab ... }

I.

{Minimal condition}

II.



Que Not starting with 'a' or not ending with 'b'. $\Sigma(a,b)$

A ∪ B

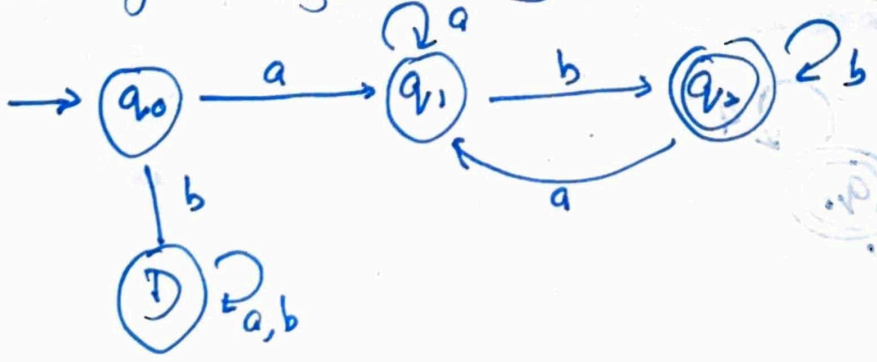
L = { ε, a, b, ba, ... }

Solve as compliment or negate

Use D'Morgan,

$(A \cup B)^c = A^c \cap B^c$

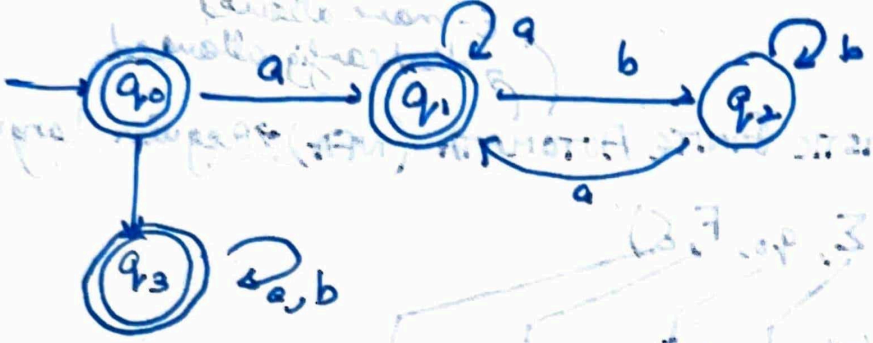
Now, Design DFA for starting with 'a' and ending with 'b'



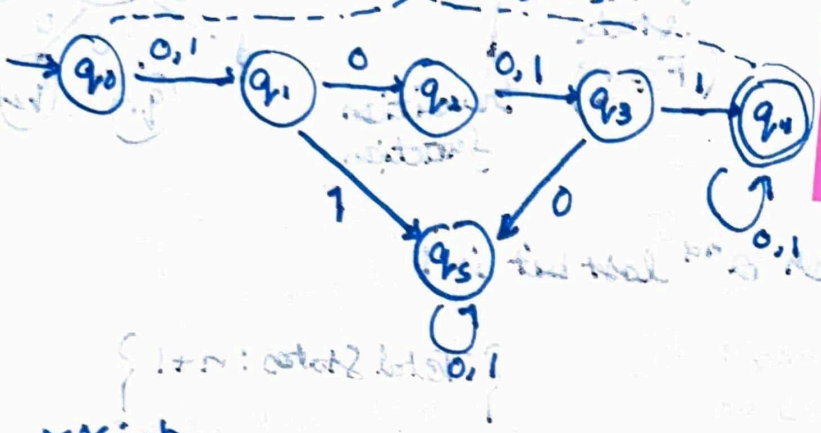
Now, take its compliment

* Final → Non-final states

Non-final → Final



Que String with 2nd symbol is '0' and fourth symbol is '1'



* Minimum Number of states from string of length n
$(n+1)$

**imp.

Que Binary string divisible by 3 . $\Sigma(0,1)$

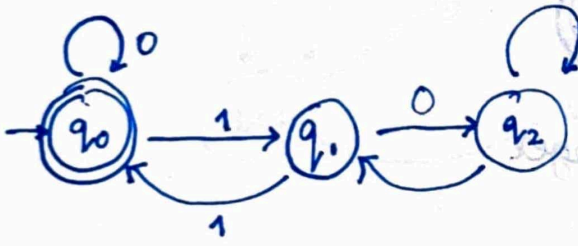
$n = 3$

remainder ⇒ $(n-1)$

0	q_0
1	q_1
2	q_2

000 - 0
001 → 1
010 → 2
⋮

for divisible by n for binary string
Minimum states are n



STEP1: Design for base case q_0, q_1, q_2

STEP2: Take case such as 101 → 5/3 = 2, so it should end at q_2

STEP3: Check for all other case and decide their end ~~States~~ according to remainder

NON-DETERMINISTIC FINITE AUTOMATA (NFA) → Regular Language

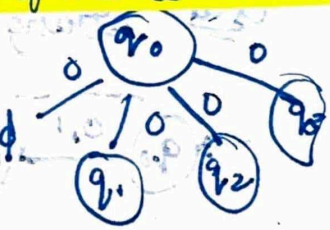
↑ ε-move allowed
↑ dead config allowed

★ Tuples: $(Q, \Sigma, q_0, F, \delta)$

- Q: finite States
- Σ: set of Symbols
- q_0: initial state
- F: final state ($F \subseteq Q$)
- δ: transition function

★ $\delta : Q \times \Sigma \Rightarrow 2^Q$

★ NFA can have different states for one input Symbol
★ Choices of moves



Que Binary string in which 2nd last bit is 1

STEP 1: Write Regex

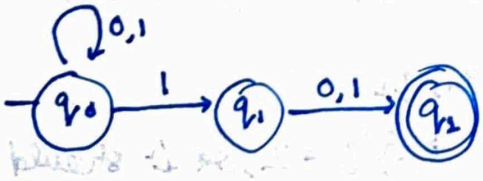
$(0+1)^* 1 (0+1)$

- $(0+1)^*$: any combi
- 1 : 2nd last
- $(0+1)$: last bit

} Total States : n+1 }

STEP 2: Design Minimum NFA

STEP 3: Design according to regex



CONVERT NFA to DFA: ☆☆

STEP 1: Make Transition Table

	0	1
→ q_0	q_0	q_0q_1
q_1	q_2	q_2
(q_2)	—	—

STEP 2: Draw DFA Transition table

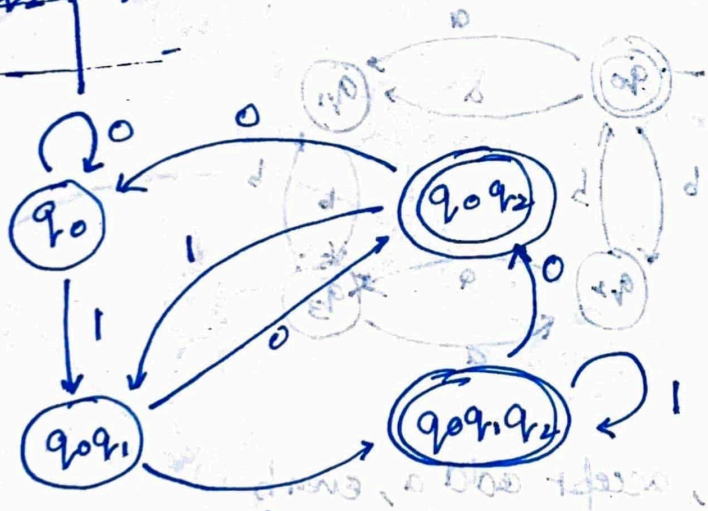
	0	1
→ q_0	q_0	q_0q_1
q_0q_1 ←	q_0q_2	$q_0q_1q_2$
(q_0q_2)	q_0	q_0q_1
($q_0q_1q_2$)	q_0q_2	$q_0q_1q_2$
q_1	q_2	q_2
q_2	—	—

I. Copy NFA & draw same

II. Define transition for undefined states

III. Mark states with prev. final as current final

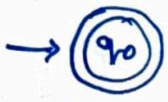
STEP 3: Draw



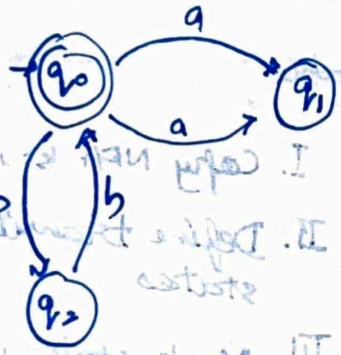
Qu: Design DFA for $L = \{w | w$ has even a's & even b's $\}$

$L = \{\varepsilon, aa, bb, aabb, abab, baab \ldots\}$

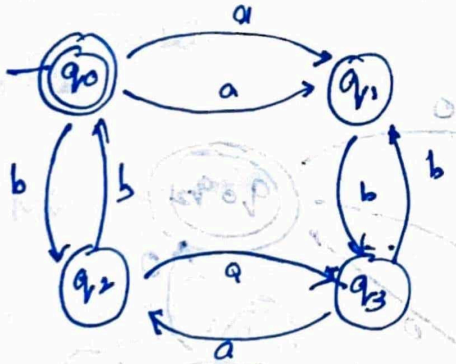
STEP 1: Design for minimum



STEP 2: Now, add more strings



STEP 3: Check for edge cases

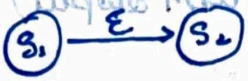


if $q_1 \rightarrow$ final, accept odd a, even b

$q_2 \rightarrow$ final, accept even a, odd b

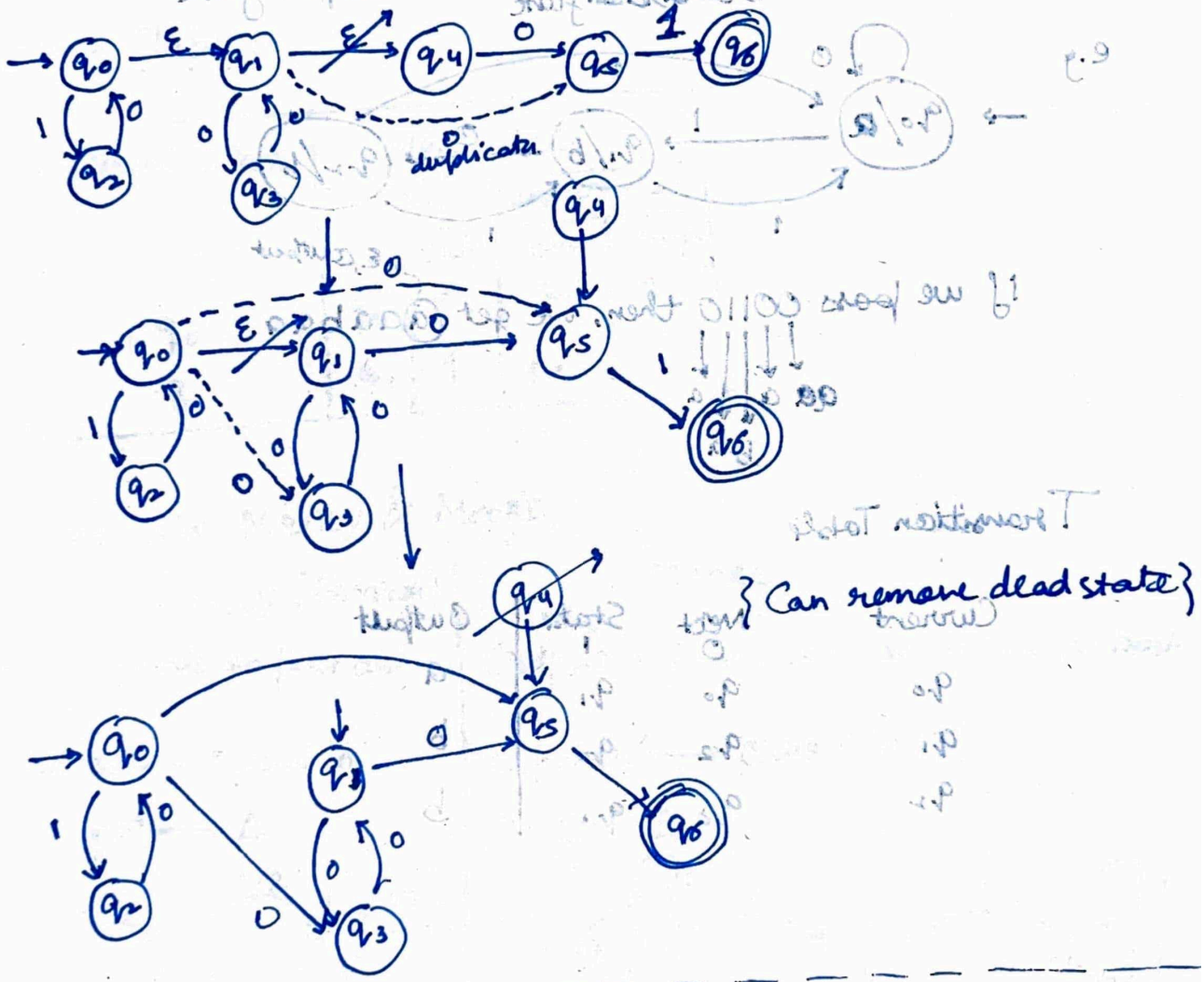
$q_3 \rightarrow$ final, accept odd a, odd b

ε-NFA (Epsilon NFA) → Eliminating ε-moves



i) find all edges starting from S_2
ii) Duplicate all edges to S_1 without changing labels
iii) if S_1 is initial state, make S_2 initial also
iv) if S_2 is final, make S_1 final also

e.g



} Can remove dead state }

Limitation of Finite State Automata

- → Limited Memory
- → String without comparison
- → Linear power

Applications

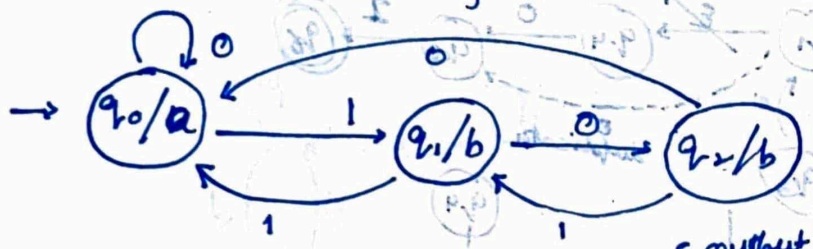
- → Word processor
- → Digital logic
- → Lexical analyser
- → Switching circuit
- → Text editors
- → Game design

MOORE MACHINE: (Finite Automata with output) {gives n+1 output}

Tuple = $\{Q, \Sigma, \delta, q_0, \Delta, \lambda\}$

- Q — set of states
- Σ — alphabet
- δ — output transition func
- q_0 — start state
- Δ — output symbol
- λ — output func

e.g



If we pass 00110 then, we get ε, output a a a b a a

a a a b a

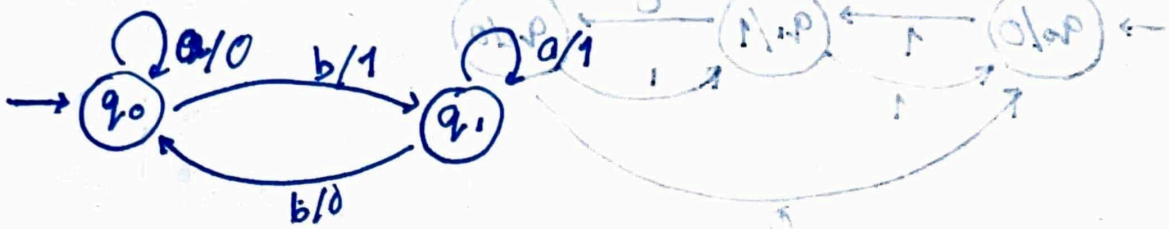
Transition Table

Current	Next State	
	0	1	Output
q_0	q_0	q_1	a
q_1	q_2	q_0	b
q_2	q_0	q_1	b

MEALY MACHINE (I/P/Output) [I/O]

Tuples : $\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ [Same as moore machine]

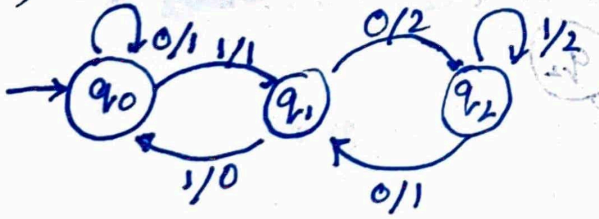
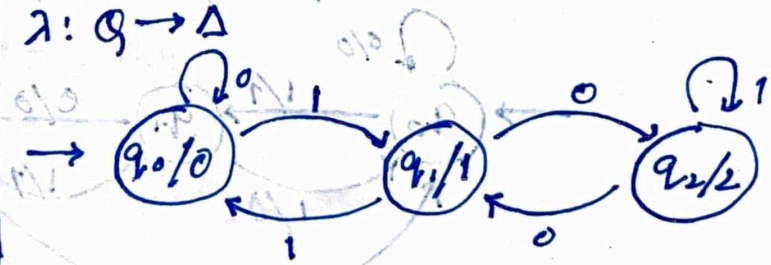
e.g



Transition Table :

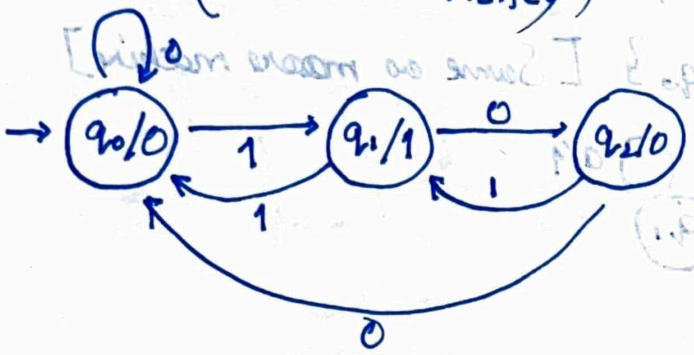
Current State	Next State			
	INPUT - 'a'		INPUT - 'b'	
	State	Outp.	State	Outp.
q_0	q_0	0	q_1	1
q_1	q_1	1	q_0	0

DIFFERENCE B/W MEALY & MOORE

MEALY (FASTER)	MOORE
i) Output depends on past and present	Output depends only on present or past
ii) Input 'n' → Output 'n'	Input 'n' → Output 'n+1'
iii) $\lambda : Q \times \Sigma \longrightarrow \Delta$	$\lambda : Q \longrightarrow \Delta$
	
iv) Output is asynchronous	Output is synchronous with clock {Generate output if time signal applied}

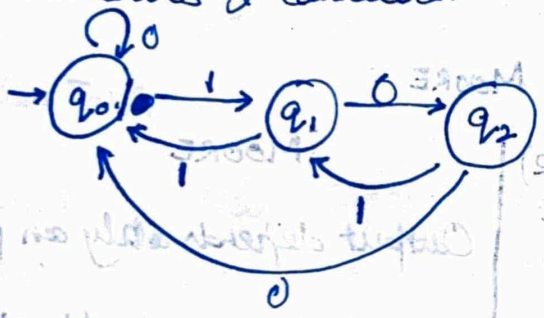
CONVERSION (MOORE TO MEALY)

e.g

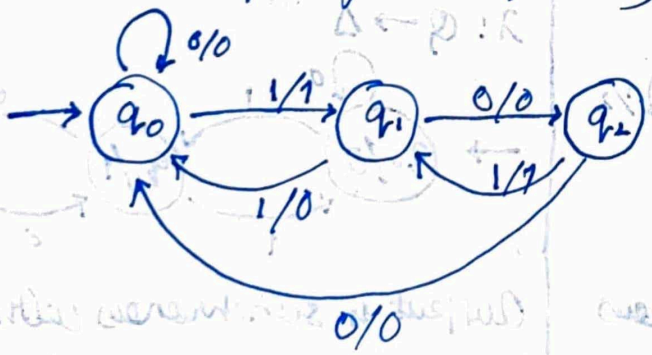


States	0	1	Output
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_0	q_1	0

STEP 1: Make States & Connection } FOR DIAGRAM }



STEP 2: Mark output of arrow pointing



STEP 3:

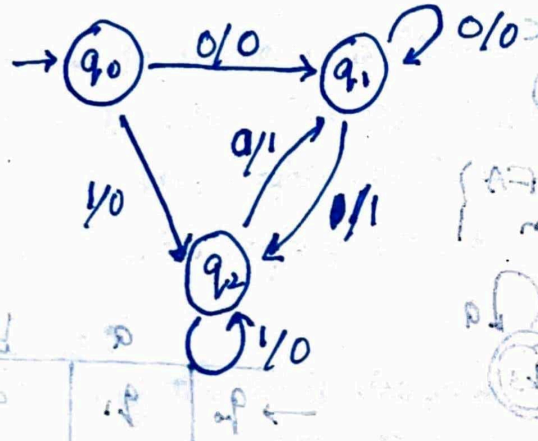
Current	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_0	q_1

} Copy everything except output

STEP 2: Refer original table for output of every 0 & 1 states

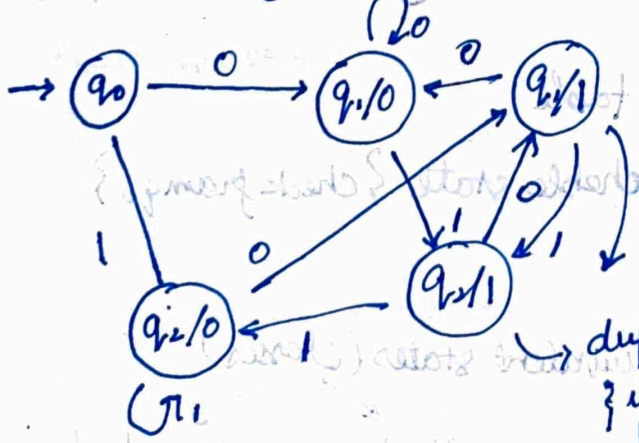
Current	0	1
q_0	$q_0, 0$	$q_1, 1$
q_1	$q_2, 0$	$q_0, 0$
q_2	$q_0, 0$	$q_1, 1$

CONVERSION (MEALY TO MOORE) { m state, n output Mealy → mxn states Moore



STEP 1: Make diagram for every transition { always start with q_0 }

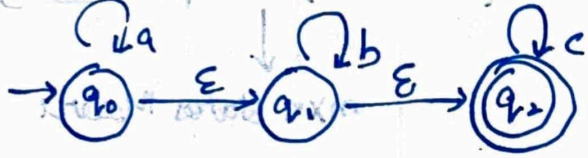
STEP 2: Draw transitions for other states



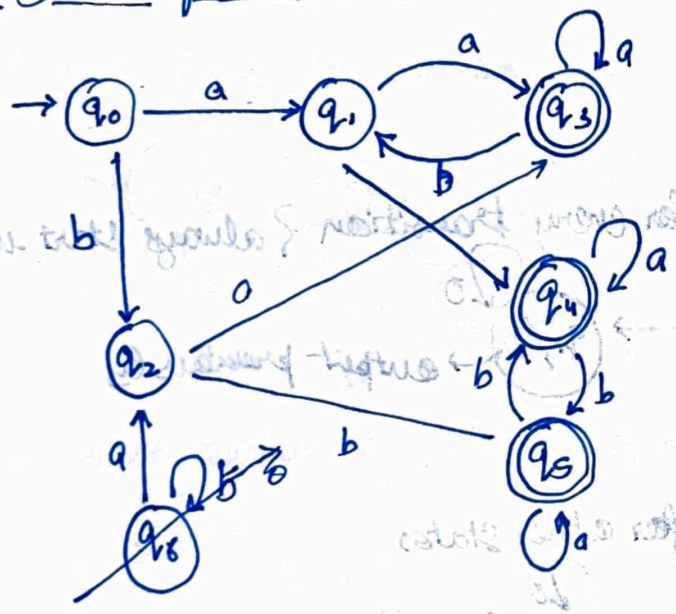
ε-NFA (Epsilon NFA) { When current state unknown }

Q - set of states
Σ - input symbols
q_0 - initial state
F - set of final states
δ - transition function

$\delta: Q \times (\Sigma \cup \varepsilon) \to 2^Q$

eg 

Minimization of DFA: { NFA → DFA n → 2^n }



	a	b
→ q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
* q_3	q_3	q_1
* q_4	q_4	q_5
* q_5	q_5	q_4
q_6	q_5	q_6

STEP 1: Make Transition table

STEP 2: Remove unreachable state { check from q_0 }
In this case (q_6)

STEP 3: Convert into equivalent states (classes)

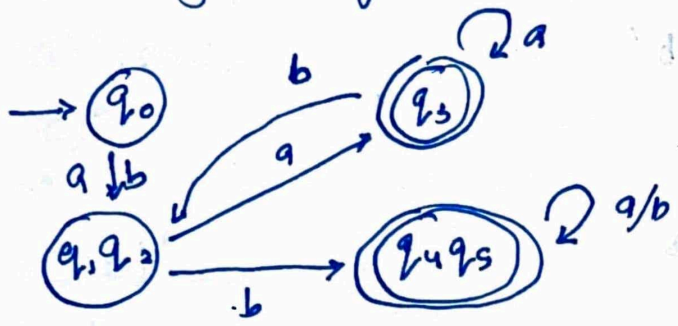
$\Pi_0 = \{ q_3 q_4 q_5 \}$ (final) $\quad \{ q_0 q_1 q_2 \}$ (non final)
G_1 $\quad$ G_2

NOTE: Check if state transition belong to same group, in Π_{n-1}

$\pi_1 = \{q_0\}\ \{q_1 q_2\}\ \{q_3\}\ \{q_4 q_5\}$

$\pi_2 = \{q_0\}\ \{q_1 q_2\}\ \{q_3\}\ \{q_4 q_5\}$ { Check groups in π_1 }

STEP 4: Make final diagram



REGULAR EXPRESSIONS:

→ Method to represent a language (L)

→ Let 'R' be a regular expression over Alphabet Σ if R is:

i) 'ε' denotes empty set {ε}

ii) ϕ, denotes empty set { }

PROPERTIES:-

i) Union of Two RE is also regular, $R_1 \cup R_2$ = Regular

ii) Concatenation of Two RE is also regular, $R_1 \cdot R_2$ = Regular

iii) Kleene Closure* of RE is also regular, R_1^* = Regular

iv) if R is Regular, (R) is also regular, (R) = Regular

e.g $R = \varepsilon$, $L(R) = \{\varepsilon\}$

$R = \phi$, $L(R) = \{\ \}$

$R = a$, $L(R) = \{a\}$

Example: { FOR FINITE LANGUAGE }

1) No String { } ϕ
2) Length 0 {ε} ε, λ
3) Length 1 {a, b} (a+b)
4) Length 2 {aa, ab, ba, bb} (aa + ab + ba + bb) → (a+b)(a+b)
5) Atmost 1 Length 0 or 1 {ε, a, b} (ε + a + b)

Example: {FOR INFINITE LANGUAGE} $\Sigma = \{a, b\}$

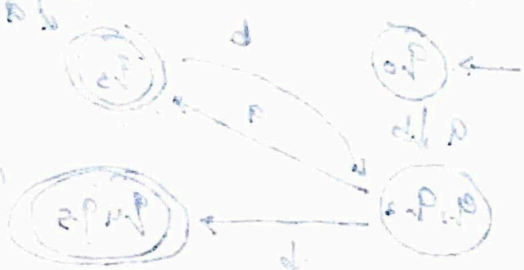
1) All strings having a single 'b'

STEP 1: Write base condition

b

STEP 2: Add remaining cases

$R = a^*ba^*$



2) All strings having atleast one 'b'

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

3) All strings having bbbb as substring

$R = (a+b)^*bbbb(a+b)^*$

4) All strings end with 'ab'

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

5) All strings start with 'ba'

$R = ba(a+b)^*$

6) All strings beginning and end with 'a'

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

7) All strings containg 'a'

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

8) All strings starting and end with different symbol

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

Que Which two of the following four RE are equivalent

i) $(00)^*(\varepsilon+0)$

ii) $(00)^*$

iii) 0^*

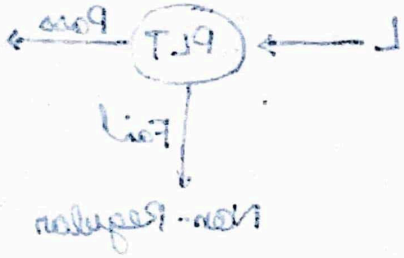
iv) $0(00)^*$

a) (i) & (ii)

b) ii) & (iii)

c) (i) & (iii)

d) (iii) & (iv)



Ans Check option first:

0^n → 'n' number of zeros

$(00)^n$ → even number of zeroes

$0(00)^n$ → odd number of zeroes

$(00)^*(\varepsilon+0)$ → 'n' number of zeroes

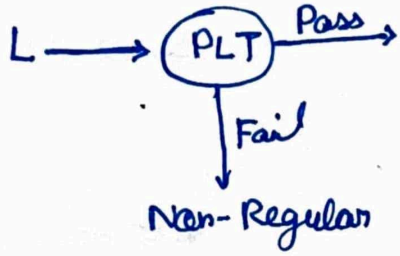
→ Negative test

PUMPING LEMMA: If L is an infinite language then there exists some positive integer 'n' (Pumping Lemma) such that any string $w \in L$ has length greater than equal to 'n' i.e $|w| \geq n$ then w can be divided into three parts, $w = xyz$ satisfy:

i) for each $i \geq 0$, $xy^iz \in L$

ii) $|y| > 0$ and

iii) $|xy| \leq n$



e.g: $L = a^n b^{2n}$, $n \geq 0$

STEP I: Take any string

$w = aabbbb$

$w \in L$

STEP II: Divide w into xyz

$\underbrace{aa}_{x}\ \underbrace{bb}_{y}\ \underbrace{bb}_{z}$

STEP III: Increase $y \to y^{i+1}$

$w' = \underbrace{aa}_{x}\ \underbrace{bbbb}_{y}\ \underbrace{bb}_{z}$

$w' \notin L$, Hence L is not Regular

Closure properties of Regular language:

1) Union ($L_1 \cup L_2$)
2) Concatenation ($L_1 \cdot L_2$)
3) Closure (*) L^*
4) Complementation $\overline{L} = \Sigma^* - L$
5) Intersection $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
6) Difference $L_1 - L_2 = L_1 \cap \overline{L_2}$
7) Reversal $(L)^R$
8) Homomorphism
9) Reverse "
10) Quotient operation
11) INIT
12) Substitution
13) Infinite Union

& not closed

⇒ REGULAR LANGUAGES ARE CLOSED UNDER REVERSAL (L^R)

STEP 1: Make initial state of M as final state of M'

STEP 2: Final state of M become initial state of M'

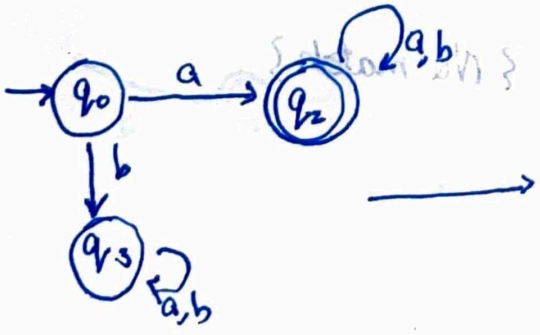
STEP 3: Reverse the direction of edges of M to make M'

STEP 4: No change in loop and remove unnecessary states

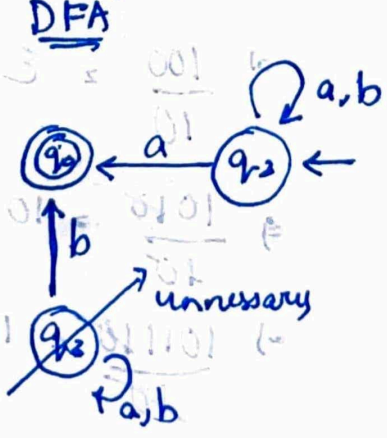
e.g L_1 = { set of all strings over (a,b) staring with a }

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

DFA



DFA'



QUOTIENT OPERATION → Left quotient (Cut Prefix)
→ Right quotient (Cut Suffix)

NOTE: L_1 and L_2 must have same symbol, Σ 0,1

$\frac{L_1}{L_2} = \{ x \mid xy \in L_1 \text{ for some } y \in L_2 \}$ Right quotient (Cut right)

$\frac{L_1}{L_2} = \{ y \mid xy \in L_1 \text{ for some } x \in L_2 \}$ Left quotient (Cut left)

e.g L_1 { 10, 100, 1010, 101110 }

L_2 = {10}

$\frac{L_1}{L_2}$ (Left) = $\frac{10}{10} = \varepsilon$ } Match left

$\Rightarrow \frac{100}{10} = 0$

$\Rightarrow \frac{1010}{10} = 10$

$\Rightarrow \frac{101110}{10} = 1110$

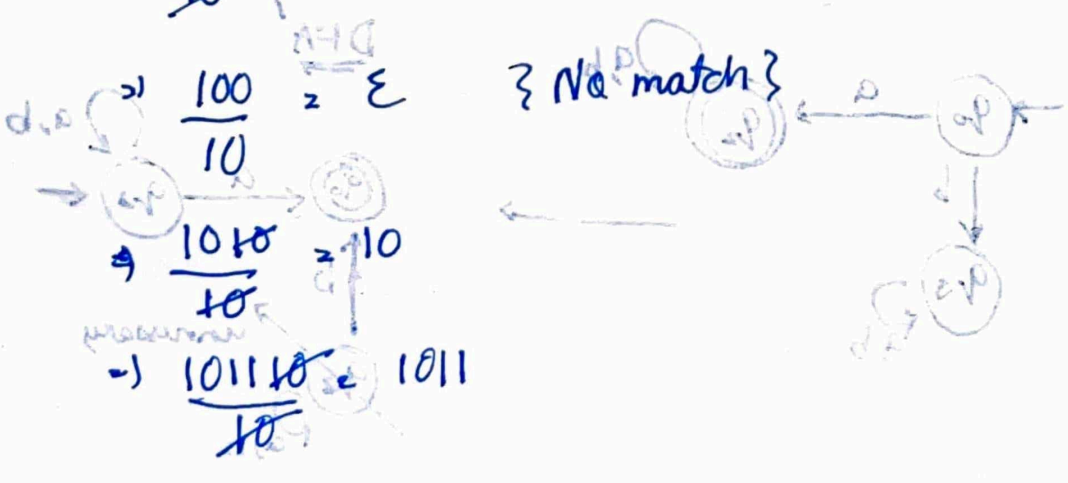
$\frac{L_1}{L_2}$ (Left) = { ε, 0, 10, 1110 }

$\frac{L_1}{L_2}$ (Right) = $\frac{10}{10} = \varepsilon$

$\Rightarrow \frac{100}{10} = \varepsilon$ } No match

$\Rightarrow \frac{1010}{10} = 10$

$\Rightarrow \frac{101110}{10} = 1011$



e.g $L_1 = a^*b = \{b, ab, aab, aaab, aaaab, \ldots\}$?Open first?

$L_2 = ab^* = \{a, ab, abb, abbb\}$

$\frac{L_1}{L_2}$ (Right) $= \{\varepsilon, \varepsilon, \varepsilon \ldots$	Yes	No	Yes			$\cup$
$\frac{L_1}{L_2}$ (Left) $= \{\varepsilon, ab, aab, \ldots$	Yes	No	Yes			$\cap$
	No	Yes	Yes	No	Yes	$\overline{L}$
				No	Yes	.
				No	Yes	*
				No	Yes	+

⇒ INIT operation (Initial / Prefix)

e.g {Set of all Prefix of $w \in L$}

Let $L = \{ab, ba\}$.

$ab = \{\varepsilon, a, ab\}$

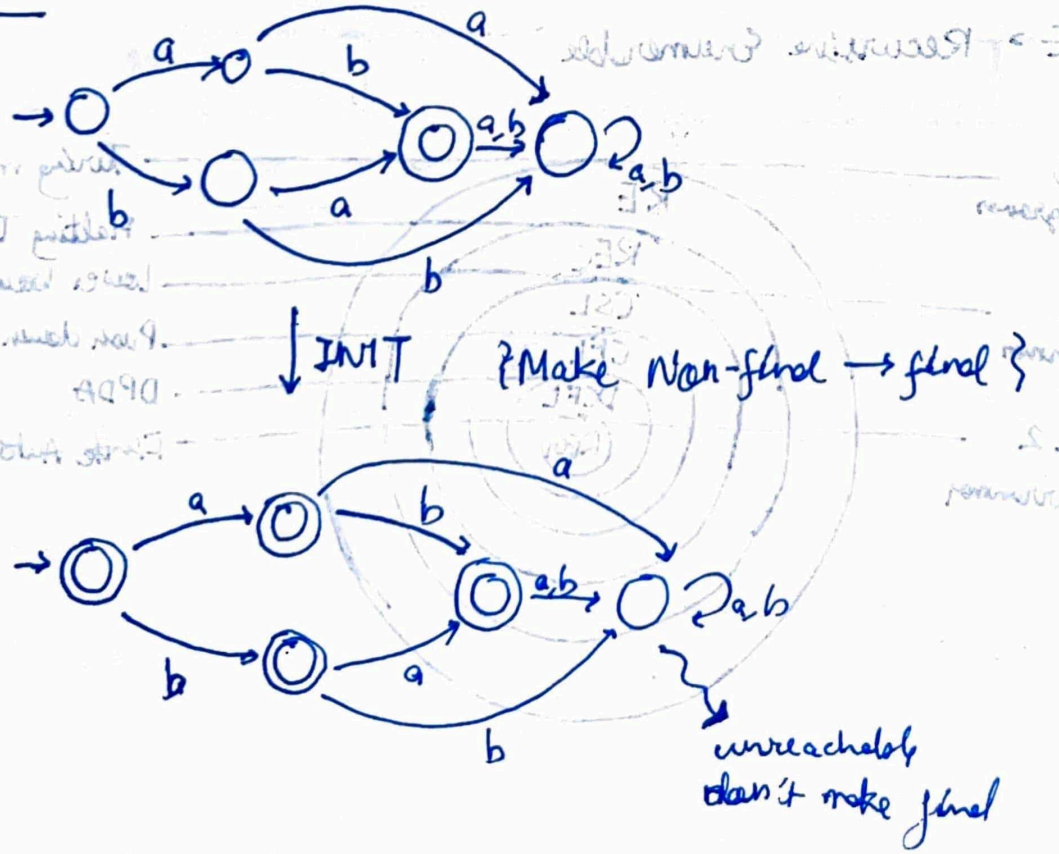
$\cup$

$ba = \{\varepsilon, b, ba\}$

$= \{\varepsilon, a, b, ab, ba\}$

RE are not closed under Infinite Union

DFA for L



Closure Properties of Various Language:

0 comparison → | 1 comparison → | 2 comparison →

	Regular	DCFL	CFL	CSL	REC	RE
$\cup$	Yes	No	Yes	Yes	Yes	Yes
$\cap$	Yes	No	No	Yes	Yes	Yes
L^c	Yes	Yes	No	Yes	Yes	No
$\cdot$	Yes	No	Yes	Yes	Yes	Yes
$*$	Yes	No	Yes	Yes	Yes	Yes
$+$	Yes	No	Yes	Yes	Yes	Yes

Regular only if no comparison unlike $a^n b^n$

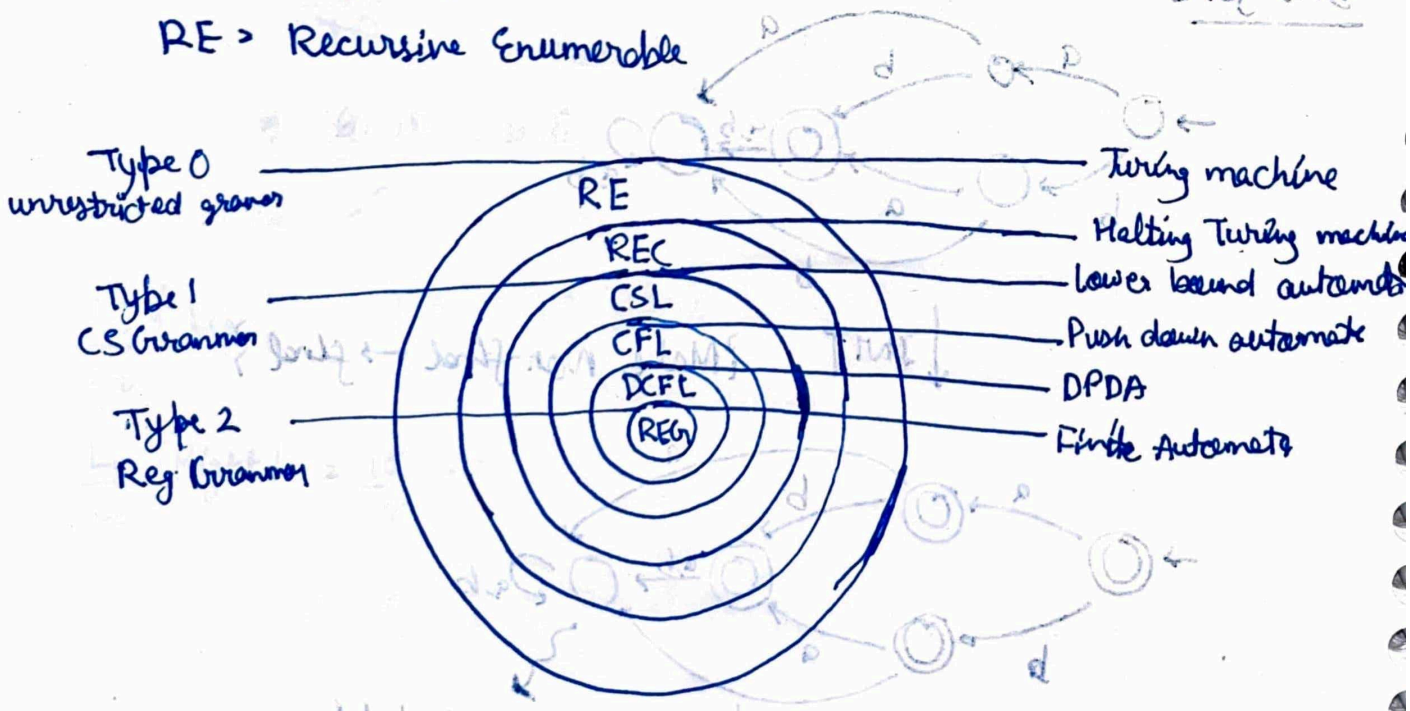
DCFL = Deterministic Context free lang.

CFL = Context free lang.

CSL = Context sensitive

REC = Recursive

RE = Recursive Enumerable



Power of DPDA < NPDA unlike NTM ≡ DTM and DFA ≡ NFA

Que: Which of the following is/are false? [DCFL and CFL] → simple comparison → guess

1) $\{a^n b^n\} \cup \{a^n b^{2n}\}$ is CFL but not DCFL

2) $a^m b^m c^n d^n$ is CFL but not DCFL

3) $\{a^m b^n \mid m, n \geq 0\}$ is DCFL not Regular

Options:

a) I only

b) I and II

c) II and III

d) I, II and III

1) Union of DCFL is not closed so, True

2) $a^m b^m c^n d^n$ is a DCFL, so, False { for every a, you pop b and for every c, you pop d }

3) $a^m b^n$ is a Regular language since no comparison, so false

	REG	DCFL	CFL	CSL	REC	RE
★ Membership problem $w \in Z^*$	✓ (FA)	✓ (PDA)	✓ (CYK)	✓	✓ (Halt-TM)	X
★ Infiniteness prob. $L = \infty$	✓	✓	✓	X	X	X
★ Emptiness prob. $L = \phi$	✓	✓	✓	X	X	X
Equality prob. $L_1 = L_2$	✓	✓	X	X	X	X
L is ambiguous	✓	✓	X	X	X	X
Completeness, $L = \Sigma^*$	✓	✓	X	X	X	X
$L_1 \cap L_2 = \phi$	✓	X	X	X	X	X
if $L_1 \subseteq L_2$ subset problem	✓	X	X	X	X	X

Decidability & Undecidability Table

HOMOMORPHISM (Some) → substitution function

$h(L)$

$h(L) = \{h(w) \mid w \in L\}$

where $h: \Sigma$ to Δ^* is called homomorphism

$\Sigma\{0,1\}$, $\Delta = \{a,b\}$

e.g $h(0) = aa$, $h(1) = bb$

if $L = \{00, 101\}$

Find $h(L)$

$h(L) = \{\underset{0}{aa}\underset{0}{aa}, \underset{\phi}{bb}\underset{0}{aa}\underset{1}{bb}\}$

INVERSE HOMOMORPHISM $h^{-1}(L)$

e.g

Let $h(0) = a$, $h(1) = b$, $h(2) = ab$

$\Sigma = \{0,1,2\}$, $\Delta = \{a,b\}$

Let $L = \{abab\}$

Find $h^{-1}(L)$

Ans $h^{-1}(L) = \{0101, 22, 201, 012\}$ } Check all possibilities

Que $h(0) = aa$, $h(1) = bb$

$\Sigma = \{0,1\}$, $\Delta = \{a,b\}$

$L = \{aa, aabb, baab, ababa\}$

$h^{-1}(L) = \{0, 01\}$

NOTE: $h(h^{-1}(L)) \subseteq L$

CONTEXT FREE LANGUAGE & GRAMMAR (Type 2)

CFG : $\{V, T, P, S\}$

V : final set of Variables (Non-terminal)

T : Set of terminals ($V \cap T = \phi$)

P : Production Rules (Substitution Rules) $\alpha \longrightarrow \beta$

(α : only one variable; β : $(V \cup T)^*$)

e.g

$S = 0S1 \mid \varepsilon$

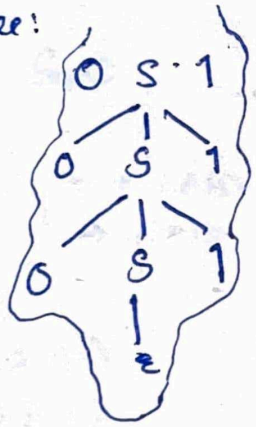
$S \rightarrow 0S1$

$\rightarrow 0S1$

$\rightarrow \varepsilon$

000111

Derivation tree :



$\rightarrow 000111$

$L = 0^n 1^n$

Que Convert CFL → CFG

1) $a^n b^n$, $n \geq 0$ → definitely check condition

STEP 1: Make expression for language

aSb { equal no. of a & b }

STEP 2: Make Start variable and terminal condition

$S \rightarrow aSb \mid \lambda$

2) $a^n b^{n+2}$, $n \geq 0$

$S \rightarrow aSbb$ {Add extra as termination}

3) $a^{2n} b^n$, $n \geq 0$

$S \rightarrow aaSb \mid \lambda$

4) $a^{2n+3} b^n$, $n \geq 0$

$S \rightarrow aaSb \mid aaa$

5) $a^m b^n$, $m > n$, $n \geq 0$

STEP 1: Make rule for $a = b$,

$S_1 \rightarrow aS_1b \mid \lambda$

STEP 2: Make rule for $m > n$

$A \rightarrow aA \mid a$

STEP 3: Concatenate S_1 and A

$S \rightarrow AS_1$

6) $\{w \mid n_a(w) = n_b(w)\}$

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$ { Combine possible 3 ways }

7) $ww^R \cup w(a+b)w^R$ { Palindrome - even, odd }

(ww^R: even; $w(a+b)w^R$: odd)

$S \rightarrow aSa \mid bSb \mid \lambda$ { even }

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda$

8) $a^m b^{m+n} c^n, \; m,n \geq 0$

$S_1 \rightarrow aS_1b \mid \lambda$

$C \rightarrow cC \mid \lambda$

+

$S \rightarrow S_1C$

DERIVATION: → LMD

→ RMD

e.g $S \rightarrow AB$

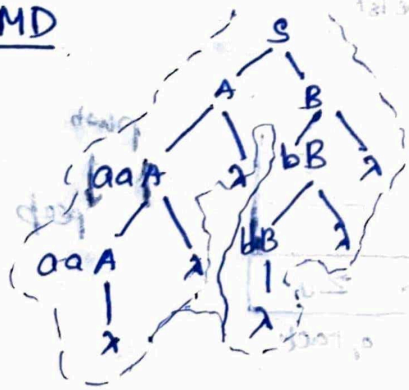
$A \rightarrow aaA \mid \lambda$

$B \rightarrow bB \mid \lambda$

Check whether

$w = aaaabb \in L(G_1)$

: LMD



$w = aaaabb$

Yes, $w \in L(G_1)$

{ First expand left side then, move to right }

: RMD (similar to LMD)

PUSH DOWN AUTOMATA (FA + stack)

$P = \{Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\}$

Q = finite set of states

Σ = Input symbol

Γ = Stack alphabet (Push into stack)

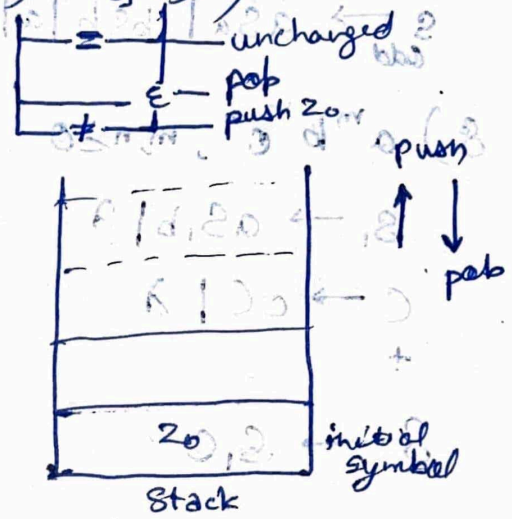
δ = Transition function ($\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \to Q \times \Gamma^*$)

q_0 = Initial state

Z_0 = ~~Final state~~ Stack start symbol

F = Final state

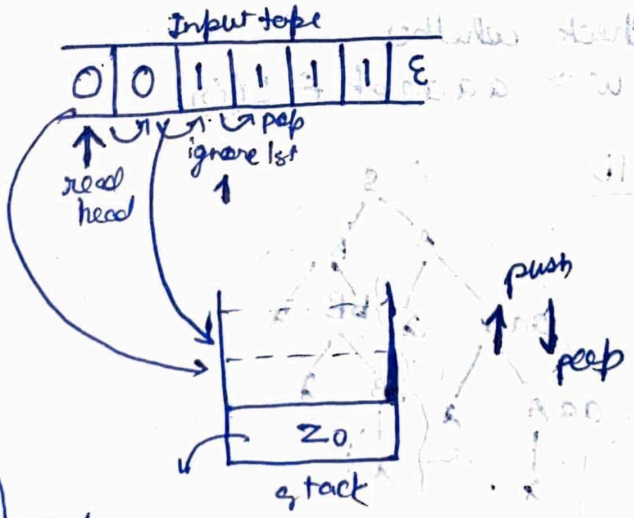
Ambiguity arise due to presence of two or more derivation trees

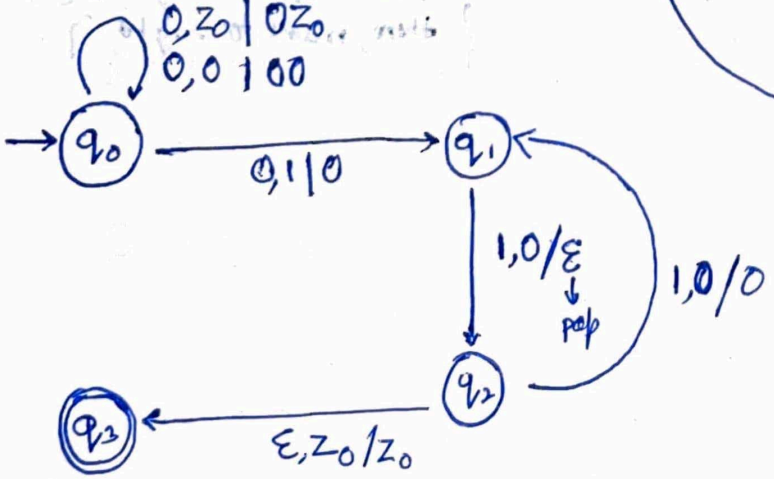


PDA Designing:

Ques $L = \{0^n 1^{2n}, n > 0\}$

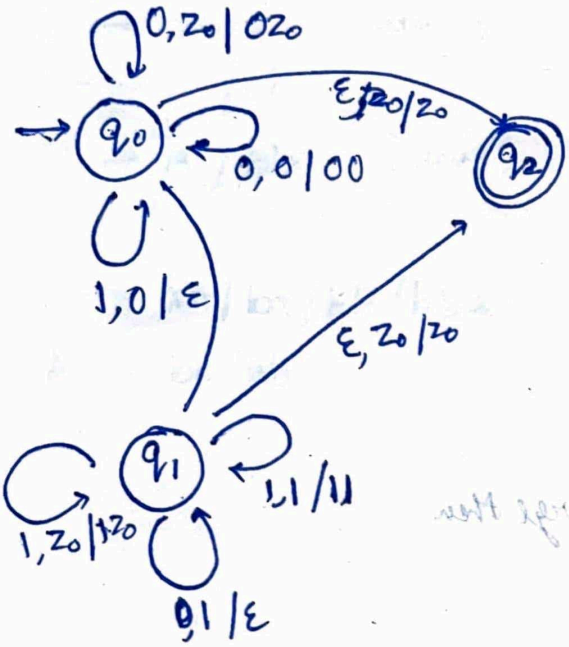
$\{011, 001111, \ldots\}$

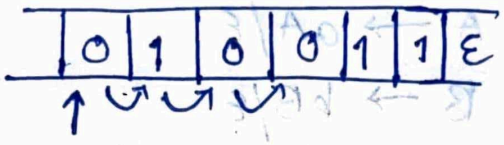


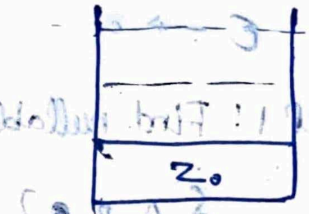


☆ Doubt

Que $L = \{ w \mid n_0(w) = n_1(w) \}$, $\Sigma(0,1)$







CLOSURE PROPERTIES OF CFL

- ★ Union ✓
- ★ Concatenation ✓
- ★ Kleen Closure ✓
- ★ Intersection ✗
- ★ Complementation ✗

Elimination of Null (ε) Production

e.g $S \rightarrow aS / A$

$A \rightarrow \varepsilon$

STEP 1: Identify Nullable rules

$\{A, S\}$

STEP 2: Try to replace the production rules (nullable) with ε and merge

$S \rightarrow aS / a / \varepsilon$

~~$A \rightarrow \varepsilon$~~

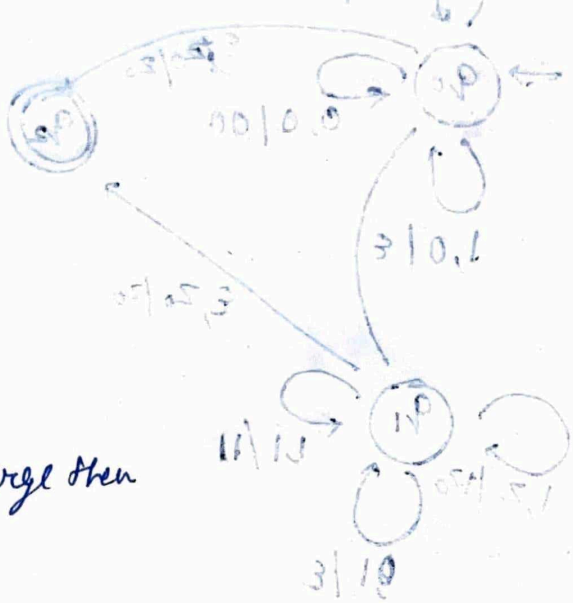
Que Eliminate nullable variables:

$S \rightarrow ABC$

$A \rightarrow aA / \varepsilon$

$B \rightarrow bB / \varepsilon$

$C \rightarrow c$



STEP 1: Find nullable production rules

$\{A, B\}$

STEP 2: Replace nullable with ε and merge them

$S \rightarrow ABC / BC / AC / C$

$A \rightarrow aA / \cancel{\varepsilon} / a$

$B \rightarrow bB / \cancel{\varepsilon} / b$

$C \rightarrow c$

Elimination of Unit Production $\alpha \rightarrow \beta$ $\{\alpha, \beta \in V\}$

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow C / b$ {unit}

$C \rightarrow D$ {unit}

$D \rightarrow E$ {unit}

$E \rightarrow a$

NOTE: Use Transitive property

NOTE: Remove direct unreachable states

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow a / b$

Que Eliminate unit production

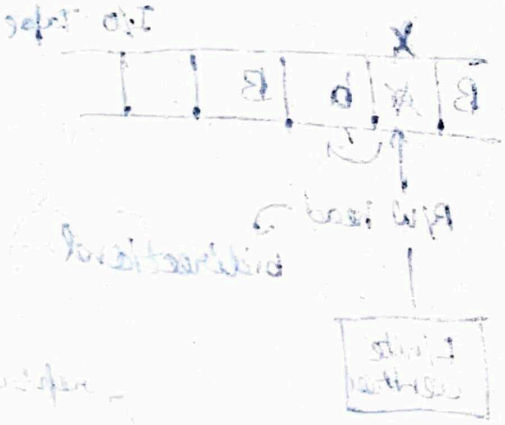
$S \longrightarrow aA/B$ {unit}

$A \longrightarrow ba/bb$

$B \rightarrow A/bba$ {unit} $\Longrightarrow$ $B \rightarrow ba/bb/bba$

$S \rightarrow aA/ba/bb/bba$

$A \rightarrow ba/bb$



TURING MACHINE {More than one comparison}

$a^n b^n c^n$, $n \geq 1$

$M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

$\delta = Q \times \Gamma \rightarrow Q \times \Gamma \times (L, R)$

FA < PDA < LBA < TM
(CSL)

Q: Set of finite states

Σ: Set of symbols

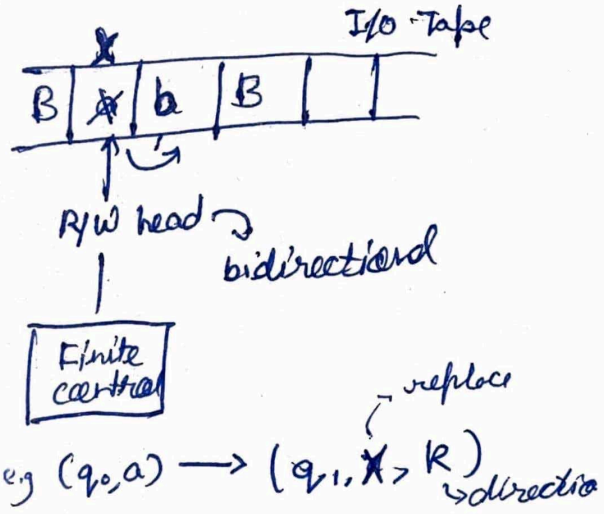
Γ: Set of symbols allowed on input tape, $\Sigma \subseteq \Gamma$

δ: Transition function

q_0: Initial state

B: Blanks

F: Final state's set



★ LBA (Linear Bounded Automata) – Limited tape TM, based on input

Standard Examples:

★★ 1) $L = \{a^n b^n c^n : n \geq 1\}$

2) $L = \{a^n : n \text{ is prime}\}$

3) $L = \{a^n : n \text{ is non-prime}\}$

4) $L = \{a^{n!}, n \geq 0\}$

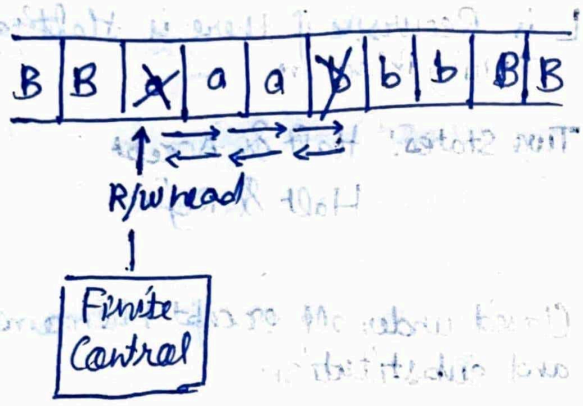
5) $L = \{ww : w \in (a,b)^+\}$

6) $L = \{wwww^R : w \in (a,b)^+\}$

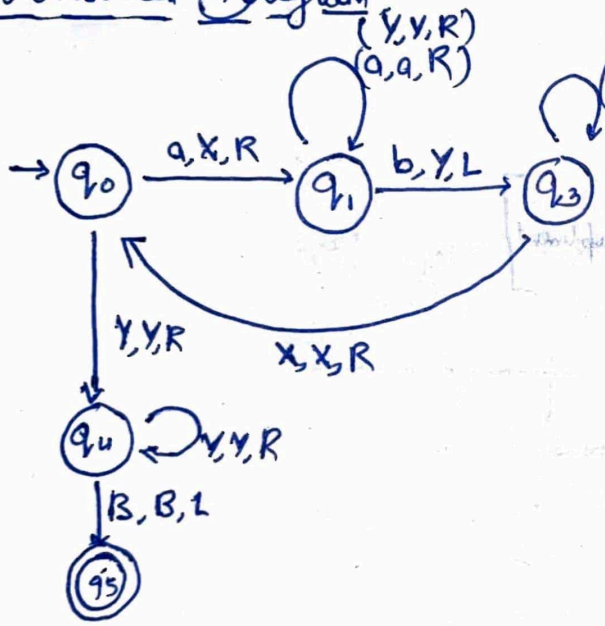
7) $L = \{w^n : w \in \{a,b\}^+, n \geq 1\}$

8) $L = \{a^n : n = m^2, m \geq 1\}$

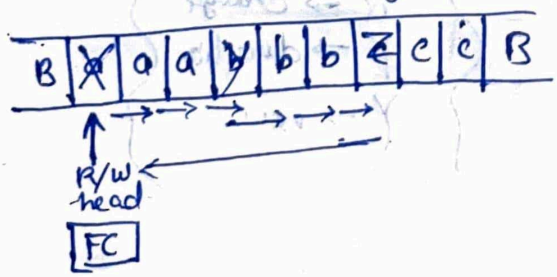
Qu Design Turing Machine for $\{a^n b^n / n \geq 1\}$



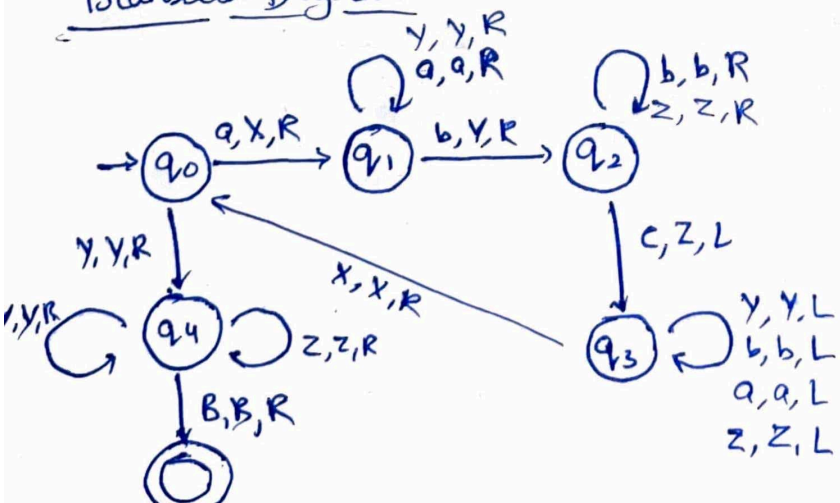
Transition Diagram:



Qu Design Turing Machine for $\{a^n b^n c^n / n \geq 1\}$



Transition Diagram



Recursive Enumerable Language

- ★ L is RE if there is TM
- ★ Three States: Halt & Accept
  Halt & Reject
  Never Halt
- ★ Closed under all except - Complement

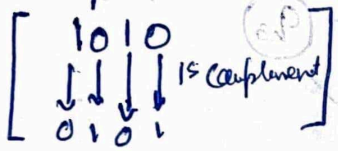
Recursive Language

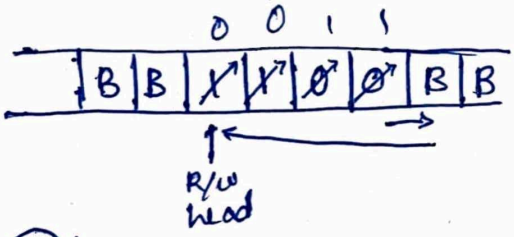
L is Recursive if there is Halting / Total Turing Machine

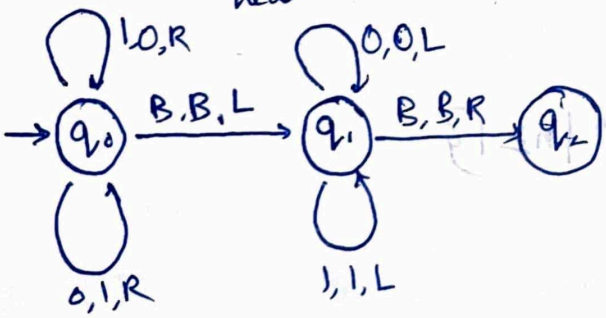
Two States: Halt & Accept
Halt & Reject

Closed under all except Homomorphism and substitution

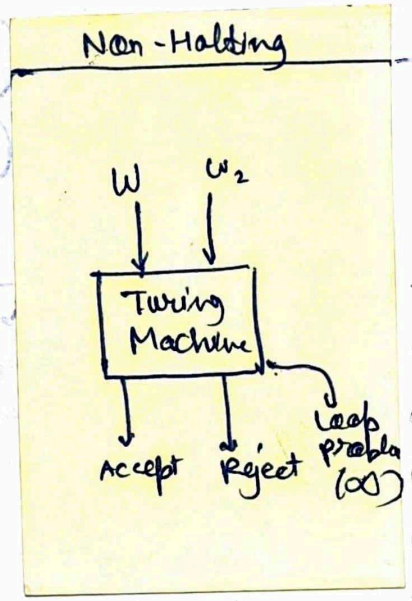
Q ⑪! Turing Machine for 1's complement (work also as transducer (input → output))

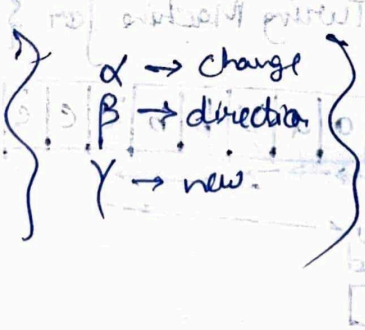






States	0	1	B
q_0	$1Rq_0$ (α β γ)	$0Rq_0$	BLq_1
q_1	$0Lq_0$	$1Lq_0$	BRq_2
q_2	—	—	—





CYK ALGORITHM

Characteristics:

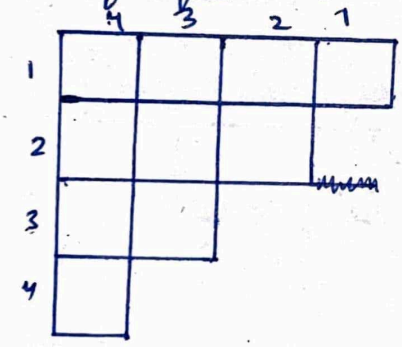
- Applicable only for Chomsky Normal form only
- It is Universal (all CNF)
- Time complexity - $O(n^3)$
- Space complexity $O(n^2)$

CNF : $A \longrightarrow BC$ or $A \longrightarrow a$

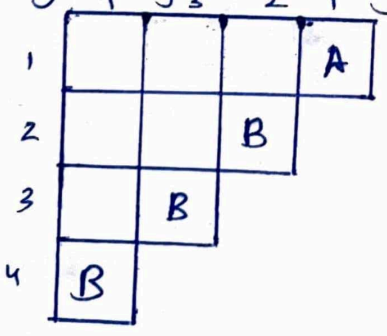
e.g Check whether string 'abbb' is a valid member of following CFG,

$$\begin{cases} S \longrightarrow AB \\ A \longrightarrow BB/a \\ B \longrightarrow AB/b \end{cases}$$

STEP 1: Create half square matrix, with n = len(string)



STEP 2: Fill diagonal first according to coordinate

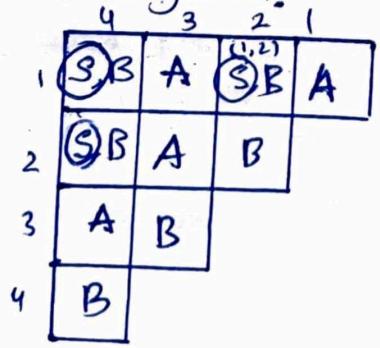


Derived by

a b b b
1 2 3 4
↓ ↓ ↓ ↓
A B B B

If S is in corner block, then string belongs.

STEP 3: Fill Next diagonal according to coordinate



for (1,2), concatenate (1,1) and (2,2)
⇒ AB, which is generated by S, B

Similary, for next block,
⇒ for (1,3), concatenate (1,2) (3,3) ⇒ AA → ϕ
concatenate (1,1) (2,3)
AA ϕ
final ⇒ AA · ϕ : ϕ · A = A

CNF (Chomsky Normal Form)	GNF (Greibach Normal Form)
★ $A \to BC$ OR $A \to a$; $A,B,C \in V$, $a \in T$	$A \to aX$; $X \in V^*$, $a \in T$
★ No. of steps required to generate a string of length 'n' is $2|n|-1$	Steps required = 'n'
★ Used in membership algo (CYK)	Used to convert CFGs to PDA
★ Derivation tree, always binary tree	Not always
★ Length of each production restricted	Not restricted