

Introduction

Set: (language) : It is a collection of objects.

unordered

(well defined)

members

cardinality

(size of set $\rightarrow$ finite)

distinct

Domain

Set types:

1. Finite and Infinite set.
2. Empty set and Universal set.
3. Countable and Uncountable set.
- 4.

Set Operations:

- Union
- Intersection
- Complement
- Set difference

$$A \cup B = \{x | x \in A \text{ or } x \in B\} \quad \text{let } |A|=m, |B|=n, \max(m,n) \leq |A \cup B| \leq m+n$$

$$A \cap B = \{x | x \in A, x \in B\} \quad 0 \leq |A \cap B| \leq \min(m,n)$$

$$\bar{A} = \{x | x \notin A\} = \{x | x \in U, x \notin A\}$$

$$A - B = \{x | x \in A, x \notin B\} = A \cap \bar{B}$$

Basics of TOC:

Type 0 - (formal languages)

Type 1 - formal languages

Type 2 - formal languages

Type 3
Regular
languages
(FA)

CFLs, PDA, CFGs

CSLs, LBA, CSGs

Recursively Enumerable sets, unrestricted
(Turing machine) grammars

• Set of all regular languages

$\subset$

Set of all context free languages

$\subset$

Set of all context sensitive language

$\subset$

Set of all Recursively Enumerable languages.

• Type 3 $\subset$ Type 2 $\subset$ Type 1 $\subset$ Type 0

• FA $\subset$ PDA $\subset$ LBA $\subset$ TM

• $LS(FA) \subset LS(PDA) \subset LS(LBA) \subset LS(TM)$

Symbols: Raw thing that can not be broken into any other symbol. (It is one length)

Alphabets (Σ): Set of (finite no. of) symbols.

Strings: Sequence of (finite) symbols called as "string"

$\Sigma^0 = \{\epsilon\}$ = set of all zero length string.

Σ^1 = set of all one length strings.

Σ^2 = set of all two length strings.

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

↓ $|w|=0$ ↓ $|w|=1$ ↓ $|w|=2$ ↓ $|w|=3$ ↓
 $\lambda \in \epsilon$ a aa aaa
b ab bba bbb
bb bbb

Language: Set of strings. $|\{\epsilon\}| = |\{\epsilon\}| = 0$ $|\{ \}$ = 0
set ↙ string ↘ size of set

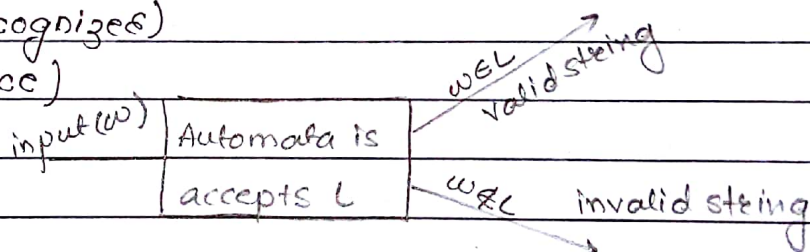
Formal language: It is a language where form of string is known.

Four types of languages have logic to compute.

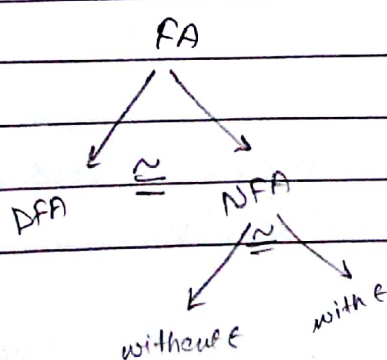
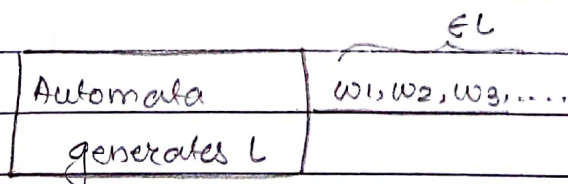
Automata:

1. **Acceptors (Recognizers)**

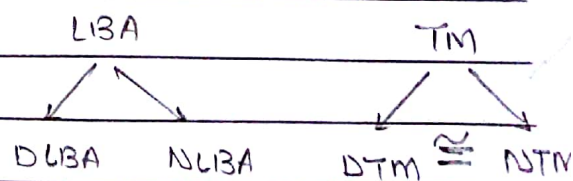
(Universal acceptance)

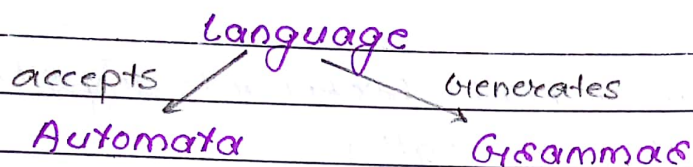


2. **Producers / Generators / Enumerators / Transducers**



PDA
↓ called as
NPDA
 $DPDA \subset PDA (NPDA)$
 $L(DPDA) \subset L(PDA)$
Set of DCFLs $\subset$ Set of CFLs





Grammar = (V, T, P, S)

$V \rightarrow$ Set of Variables (Non-terminals) $V = \{S, A, B\}$

$T \rightarrow$ Set of Terminals $T = \{a, b\}$

$P \rightarrow$ Set of rules (production) $P = \{S \rightarrow AB, A \rightarrow a\}$

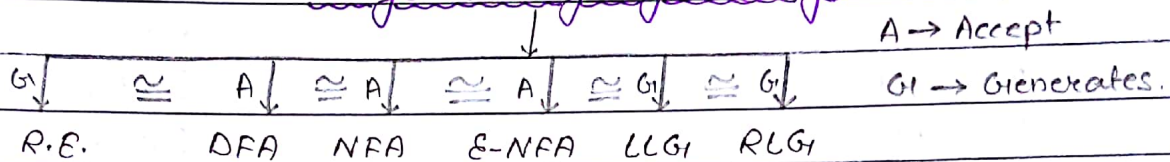
$S \rightarrow$ Start Symbol.

Grammar is set of rules which generates some rules.
Automata is a machine which accepts some language.

Finite Automata

1. Regular Expression
2. Finite Automata (DFA, NFA)
3. Regular Grammars (LLG, RLG)
4. Equivalences ($RE \cong FA \cong \text{Reg. Grammars}$)
5. How to identify a regular languages?
6. Closure properties of regular languages.

Regular Languages (Regular sets)



Regular Expression

- It represents regular language.
 - It generates a regular language.
 $L(R.E) = \text{Regular Language}$
 - It has a declarative way to represent a regular language. program
 - Four operators:

$*$	$+$	$\cdot$	$+$
Unary	Binary		
- $*$ $\rightarrow$ Kleene closure

$+$ $\rightarrow$ positive closure

$\cdot$ $\rightarrow$ Concatenation

$+$ $\rightarrow$ OR.

operator $\downarrow$

Logic $\downarrow$

function $\downarrow$

relation $\downarrow$

set.

1. $+$ (OR)

set.

$$a+b = \{a\} \cup \{b\}$$

$$\therefore ab+bba = \{ab, bba\}$$

2. Concatenation ($\cdot$) symbol.

$$a \cdot b = ab$$

$\rightarrow$ string

$$(aaa) \cdot (bb) = aaabb$$

$$a^* \cdot (b^*) \rightarrow \text{expression}$$

3. Kleene closure ($*$)

$$a^* = a^{\geq 0} = a^{\text{any}}$$

$$a^* = \{a^n \mid n \geq 0\}$$

= set of all strings over $\Sigma = \{a\}$

4. Positive closure (+)

$$a^+ = a^{\geq 1}$$

at least one

$$a^* = a^+ + \epsilon$$

$$= a^* - \{a^0\} = a^* - \{\epsilon\}$$

$\therefore a^* =$ set of all strings except zero length (ϵ).

Regular Expression

1. $\emptyset$
↳ empty exp.

Set
 $\{\} = \emptyset$
↳ empty set (lang)

2. ϵ

$\{\epsilon\}$

3. a

$\{a\}$

4. $a+b$

$\{a, b\}$

5. $a+\epsilon$

$\{\epsilon, a\}$

6. $a.b$

$\{ab\}$

7. $a+ba$

$\{a, ba\}$

8. a^*

$\{\epsilon, a, aa, \dots\} = \{a^n \mid n \geq 0\}$
 $= \{a^n\}$

9. a^+

$\{a, aa, \dots\} = \{a^n \mid n \geq 1\}$
 $= \{a^{n+1} \mid n \geq 0\}$

Q. Simplifying the following Regular Expression.

$$\emptyset \cdot \text{any} = \emptyset$$

$$\text{any} \cdot \emptyset = \emptyset$$

1. $\epsilon + \epsilon = \epsilon$

2. $\epsilon \cdot \epsilon = \epsilon$

3. $\emptyset \cdot \emptyset = \emptyset$

4. $\emptyset \cdot \epsilon = \emptyset$

5. $\epsilon \cdot \emptyset = \emptyset$

6. $a \cdot a = aa$

7. $\epsilon^* = \epsilon$

8. $\emptyset^* = \epsilon$

$$[\text{any}]^0 = \epsilon$$

9. $\emptyset^+ = \emptyset$

10. $\epsilon^+ = \epsilon$

11. $\emptyset + \emptyset = \emptyset$

12. $\emptyset \cdot (a + \epsilon) = \emptyset$

13. $a + a = a$

14. $a \cdot a = aa$

15. $(\epsilon + \emptyset)^+ = (\epsilon)^+ = \epsilon$

16. $(\epsilon + \emptyset)^* = (\epsilon)^* = \epsilon$

17. $(\epsilon \cdot \emptyset)^+ = \emptyset^+ = \emptyset$

18. $(\epsilon \cdot \emptyset)^* = \emptyset^* = \epsilon$

19. $(\epsilon \cdot \epsilon)^+ = \epsilon^+ = \epsilon$

20. $(\emptyset \cdot a)^* = \emptyset^* = \epsilon$

21. $a + \epsilon = a + \epsilon$

22. $a \cdot \epsilon = a$

23. $\epsilon \cdot a = a$

24. $\epsilon \cdot a \cdot \epsilon = a$

25. $a \cdot \epsilon \cdot a = aa$

26. $\epsilon \cdot \epsilon \cdot a = a$

27. $a \cdot a \cdot \epsilon = aa$

28. $a \cdot \epsilon \cdot b = ab$

29. $ab \cdot \epsilon \cdot b = abb$

30. $(a + \epsilon) \cdot b = ab + b$

Properties of operators (' $\circ$ ' and '.')

1. Closed (closure, operation, operator)

(Domain, operation) is closed iff " $\forall x, y \in D \Rightarrow x \circ y \in D$ "

- i. $\circ$ $\Rightarrow$ closed for regular expressions
- ii. concatenation $\Rightarrow$ closed
- iii. kleene closure $\Rightarrow$ closed
- iv. positive closure $\Rightarrow$ closed

2. Associative

$$i. [(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3)]$$

$\xrightarrow{\text{L to R left}} \quad \quad \quad \xleftarrow{\text{R to L right}}$

Eg. $(a+b)+c = a+(b+c)$
 $\{a, b, c\} \quad \{a, b, c\}$

$\circ$ (+) satisfies associative

$$ii. (R_1 \cdot R_2) \cdot R_3 = R_1 \cdot (R_2 \cdot R_3)$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

$$\{a, b, c\} \quad \{a, b, c\}$$

Concatenation (.) satisfies associative.

3. Identity

$$\circ: R + \emptyset = \emptyset + R = R$$

$\emptyset$ (empty exp.) is identity for $\circ$.

$$\text{concatenation: } R \cdot \epsilon = \epsilon \cdot R = R$$

ϵ (empty string) is identity for concatenation.

4. Inverse (always depends on identity)

$$\circ: R + [?] = [?] + R = \emptyset \quad \text{"Inverse not exist for 'o'"}"$$

$$\text{concatenation: } R \cdot [?] = [?] \cdot R = \epsilon \quad \text{"Inverse not exist for concatenation"}"$$

(Identity of concatenation, ϵ)

5. Commutative:

o/s: $R_1 + R_2 = R_2 + R_1$ Eg. $a + b = b + a$ $\therefore$ satisfies.

concatenation: $R_1 \cdot R_2 \neq R_2 \cdot R_1$ Eg. $ab \neq ba$ $\therefore$ Not-satisfies

$R_1 \cdot R_2 = R_2 \cdot R_1$ for some R_1, R_2

1. $R_1 = R_2$

4. $R_1 = \epsilon$

2. $R_1 = \emptyset$

5. $R_2 = \epsilon$

3. $R_2 = \emptyset$

6. Distributive

o/s over concatenation: $R_1 + (R_2 \cdot R_3) = (R_1 + R_2) \cdot (R_1 + R_3)$ $\therefore$ Not satisfies.

concatenation over o/s: $R_1 \cdot (R_2 + R_3) = (R_1 \cdot R_2) + (R_1 \cdot R_3)$ $\therefore$ satisfies

o/s cannot be distributed over concatenation but concatenation can be distributed over o/s.

Note: Dominant

$R \cdot [\emptyset] = [\emptyset]$ dominates for any exp. in concatenation
 $R + (\Sigma^*) = (\Sigma^*)$ universal set

Understanding Regular Expression (Language generated by expression)

1. $R = \emptyset$

$L(R) = \{ \}$

2. $R = \epsilon$

$L(R) = \{ \epsilon \}$

3. $R = a^*$

$L(R) = \{ a^0, a^1, \dots \}$

$= \{ \epsilon, a, aa, \dots \}$

$=$ set of all strings over a 's.

4. $R = a^+$

$L(R) = \{ a, a^2, a^3, \dots \}$

$=$ set of all strings

over a 's except empty string.

5. $R = a+b$

$L(R) = \text{At least one 'a' followed one 'b'}$

write all possible equivalent regular expressions for $(a+b)^*$

$R = (a+b)^* = \text{Universal language (set) over } \Sigma = \{a, b\}$

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

$= (a+b)^* + \text{any}$

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

$= (a+b+(\epsilon))^* \rightarrow \text{all strings covered in } (a+b)^*$

$= (a+b+\epsilon)^*$ *redundant*

$= (a+b+\epsilon)^+$

$= (a+b+(aaba^*b))^* \rightarrow \text{redundant term}$

Note: $a^* = (a+\epsilon)^*$, $(a+\epsilon)^+$, a^+

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

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

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

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

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

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

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

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

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

These can be strings (substrings) of length $1=n$, $2=n-1$, $3=n-2$ strings of length $n-1=2$.

Total no. of maximum sub-strings
 $= \frac{n(n+1)}{2}$

Total no. of maximum sub-strings of different lengths = n .

Note:

Any given RE can have contain infinite equivalent regular expression.

Write regular expressions for the following regular language.

4. Set of all strings over $\Sigma = \{a, b\}$

$(a+b)^*$

2. In Q.1 all strings except empty string.
 $(a+b)^+$

3. set of all strings over $\Sigma = \{a, b\}$ where each string starts with 'a'.
 $a(a+b)^*$

4. Ends with 'b'. $\rightarrow (a+b)^*b$

5. starts with 'a' and ends with 'b' $\rightarrow a(a+b)^*b$

6. contain 'ab' as a substring $\rightarrow (a+b)^*ab(a+b)^*$

7. 2nd symbol from beginning is 'a'. $\rightarrow (a+b)^*a(a+b)^*$
starts with 'aa' or 'ba' $\rightarrow (aa+ba)(a+b)^*$

8. 2nd symbol from ending is 'b'. $\rightarrow (a+b)^*b(a+b)$

9. 2nd symbol is 'a' and 4th symbol is 'b'
 $(a+b)a(a+b)b(a+b)^*$

Ends with 'ba' or 'bb' $\rightarrow (a+b)^*(ba+bb)$

$(aaab, aabb, baab, babb) \sim$ starts.

10. Empty language - $\emptyset$

11. $\{w/w \in (a+b)^*, w \text{ starts with } aa \text{ or } bb\}$
 $aa\alpha + bb\alpha = (aa+bb)\alpha = (aa+bb)(a+b)^* \quad \text{min} = aa \text{ or } bb$

12. $\{w\alpha/w \in (a+b)^*, \alpha \in (aa+bb)\}$ (ends with aa or bb)
 $\alpha(a+b)^* = (a+b)^*(aa+bb)$

13. Each string starts with 'aa' or ends with 'bb'.
 $aa(a+b)^*bb + (a+b)^*bb$

14. Each string starts with 'aa' and ends with 'bb'.

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

15. Each string starts with 'a' or contains 'b'.

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

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

16. Each string starts with 'a' and contains 'b'.

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

17. Each string starts with 'a' or contains 'a'. (contains 'a' only)

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

18. Each string starts with 'a' and contains 'a'.

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

19. Each string ends with 'a' or contains 'b'. $(a+b)^*a + (a+b)^*b(a+b)^*$

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

20. Each string ends with 'a' and contains 'b'. $(a+b)^*b(a+b)^*a$

21. Each string ends with 'a' or contains 'a'. $\rightarrow (a+b)^*a(a+b)^*$

22. Each string ends with 'a' and contains 'a'. $\rightarrow (a+b)^*a$

ends with 'a'.

23. Each string has 2nd symbol is 'a' and 3rd symbol is 'b'.

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

24. Each string has 2nd symbol is 'a' or 3rd symbol is 'b'

$$(a+b)a(a+b)^* + (a+b)^2b(a+b)^*$$

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

fa

25. Each string contain 'aa' or 'bb'.

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

26. Each string contain aa and contain bb.

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

27. Length of each string is exactly 3.

$$(a+b)(a+b)^2 = (a+b)^2(a+b) \text{ or } (a+b)(a+b)(a+b)$$

28. Length of each string is atleast 3. $\{w/w \in (a+b)^*, |w| \geq 3\}$

$$(a+b)(a+b)^2(a+b)^* \text{ or } (a+b)^+(a+b)^2$$

29. Length of each string is atmost 3. $\{w/w \in (a+b)^*, |w| \leq 3\}$

$$\epsilon + (a+b) + (a+b)^2 + (a+b)^3 \text{ or } (\epsilon + a + b)^3$$

30. Length of each string is even. (not odd, divisible by 2, 0 modulo 2, multiple of 2)

$$\{w/w \in (a+b)^*, |w| = 2n, n \geq 0\}$$

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

31. Length of each string is ~~th~~ divisible by 3.

$$((a+b)^3)^*$$

32. Length of each string is odd. (not even, 1 modulo 2, $2n+1$)

$$((a+b)^2)^*(a+b)$$

33. Length of each string is 2 modulo 3. $(3n+2) n \geq 0$.

$$(a+b)^2((a+b)^3)^*$$

34. Length of string is not divisible by 3.

$$\{w \mid w \in (a+b)^*, |w| = (3n+1) \text{ or } (3n+2), n \geq 0\}$$

$$|w| \neq 3n$$

$$(a+b)((a+b)^3)^* + (a+b)^2((a+b)^3)^*$$

35. Each string has exactly two a's.

$$b^*ab^*ab^*$$

36. Each string has at least two a's.

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

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

37. Each string has at most two a's.

$$b^* + b^*ab^* + b^*ab^*ab^* \text{ or } b^*(a+\epsilon)b^*(a+\epsilon)b^*$$

38. Each string starts and ends with different symbols.

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

39. Each string starts and ends with same symbol.

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

40. $\{w \mid w \in (a+b)^*, w \text{ starts with 'a' and 'b'}\} \Rightarrow \emptyset$

41. Each string ends with both 'a' and 'b'. $\Rightarrow \emptyset$.

42. Each string contains both 'a' and 'b'.

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

43. Each string contains 'aaa' as a substring.

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

44. Each string not containing 'aaa' as a substring.

$\Rightarrow FA \Rightarrow RE$

45. Each string not starts with 'a'.

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

46. Each string not ends with b.

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

47. Even no. of a's. $(b^*ab^*ab^*)^* + b^*$ or $(b^*ab^*ab^*)^*b^*$
 $b^*(b^*ab^*ab^*)b^*$
 $b^*(ab^*ab^*)^*b^*$
 $b^*(bab^*a)^*b^*$

48. $\{w \mid w \in (a+b)^*, n_a(w) = \text{even or } n_b(w) = \text{odd}\}$

$$[(b^*ab^*ab^*)^* + b^*] + a^*ba^*[a^*ba^*ba^*]^*$$

49. Odd no. of a's. $b^*ab^*(b^*ab^*ab^*)^*$

$$= (b^*ab^*ab^*)^*b^*ab^* = (bab^*ab^* + b)^*ab^*$$

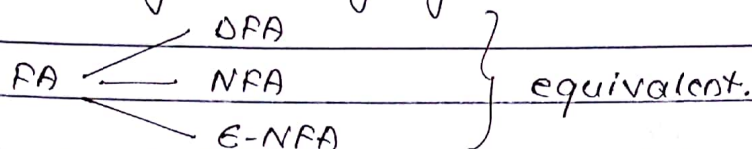
Languages over one symbol:

50. $(11+1111)^*$, $\Sigma = \{1\}$ $= \{\epsilon, 12, 14, 16, \dots\}$
 $= (11)^*$

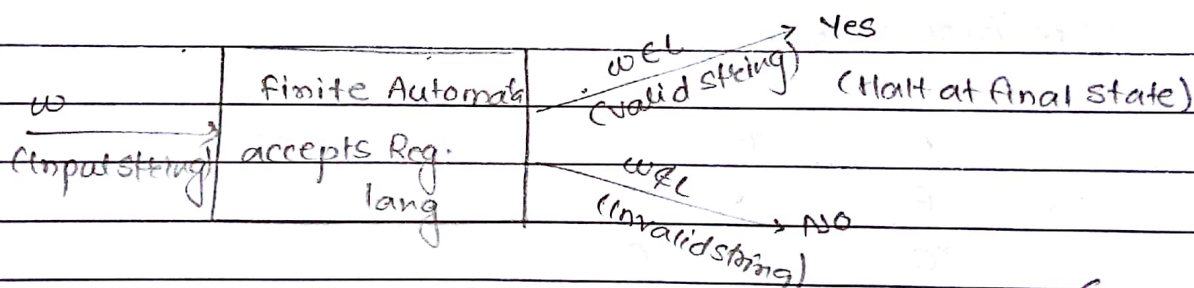
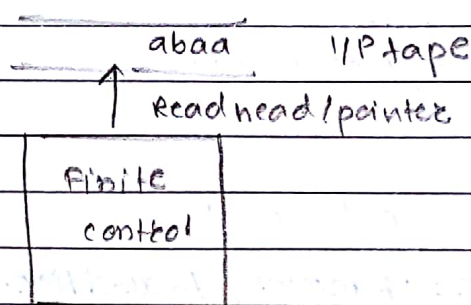
51. $(11+111)^*$, $\Sigma = \{1\}$ $= \{\epsilon, 12, 13, 14, 15, \dots\}$
 $= \epsilon + 111^* = \epsilon + 11^* = \epsilon + 1^*1 \quad \{n \mid n \neq 1\}$

Finite Automata:

- Do not require any memory.
- It accepts regular language.



- $L(\text{DFA}) = L(\text{NFA}) = L(\text{E-NFA}) = \text{Regular language}$
 - FA only depends on states to accept a language (regular)
- FA configuration: only from left to right direction



$$FA = (Q, \Sigma, \delta, q_0, F)$$

where

Q = Set of states

Σ = Set of symbols

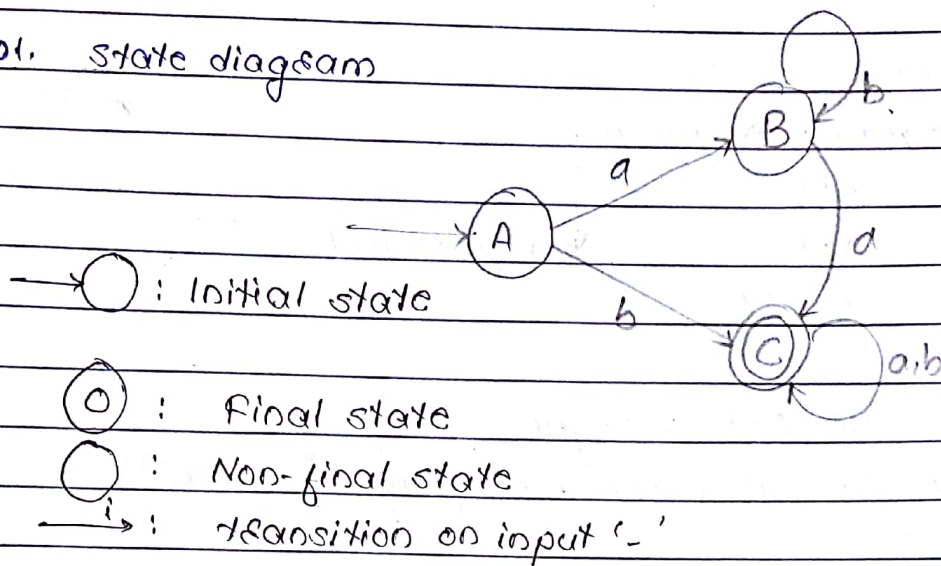
δ = Transition function

q_0 = start state $q_0 \in Q$

F = state of final states $F \subseteq Q$

Representation of Finite Automata:

01. State diagram



How to run a string on machine?

(w)

abba:

Run: $A \xrightarrow{a} B \xrightarrow{b} B \xrightarrow{b} B \xrightarrow{a} C$

Sequence of moves/transitions/path

02. Transition Table

S	a	b
→ A	B	C
B	C	B
*C	C	C

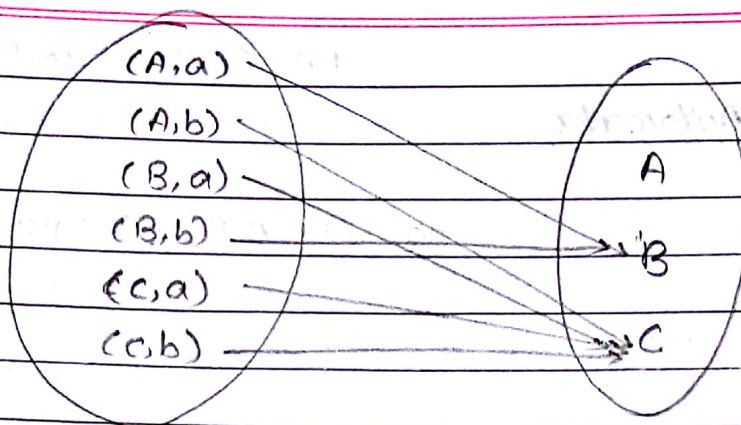
03. Mathematical model (set / relation / function)

(transition function)

$Q = \{A, B, C\}$, $\Sigma = \{a, b\}$, $q_0 = A$, $F = \{C\}$

$FA = (\{A, B, C\}, \{a, b\}, \delta, A, \{C\})$

δ : $\delta(A, a) = B$ $\delta(B, b) = B$
 $\delta(A, b) = C$ $\delta(C, a) = C$
 $\delta(B, a) = C$ $\delta(C, b) = C$



(may change as per type of FA codomain.)

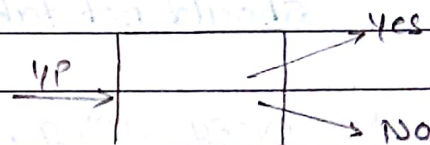
codomain

$$S = \{ (A,a), B, (A,b), C, (B,a), C, (B,b), B, (C,a), C, (C,b), C \}$$

Finite Automata, Finite machine, Finite state machine are same.

Finite Automata (without Output)

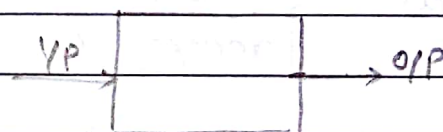
— Accepts Regular language (Recognize)



Finite Automata (with output)

deterministic Moore machine
• Mealy machine

} produce output for input.



Types of Finite Automata (without output)

- Deterministic Finite Automata
- Non-Deterministic Finite Automata without ϵ -moves
- Non-Deterministic Finite Automata with ϵ -moves.

$$FA = (Q, \Sigma, \delta, q_0, F)$$

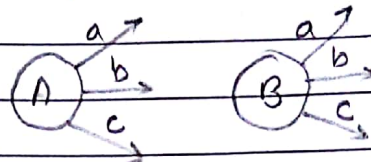
Deterministic Finite Automata

01. $\delta: Q \times \Sigma \rightarrow Q$

Eg. $\delta(A, a) = B$

$$A \in Q, a \in \Sigma, B \in Q$$

02. $Q = \{A, B\}, \Sigma = \{a, b, c\}$



$$|S| = |Q| * |\Sigma|$$

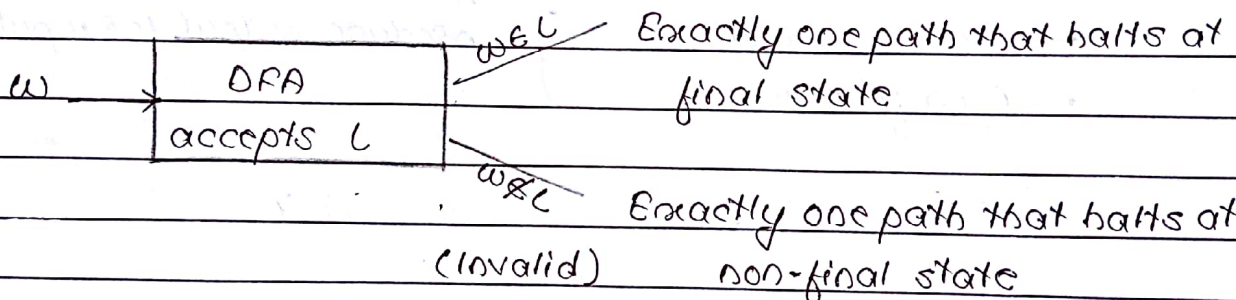
From each state, for every I/P symbol, exactly one transition to the next state.

Note 01: Transition should not miss from any state for any input symbol.

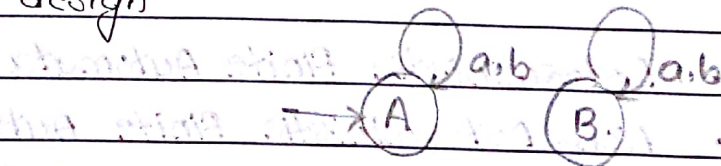
Should not taken more than one transition for same input.

03. For every string, No. of paths = 1.

04. (valid)



DFA → Easy to understand
Difficult to design



Complete DFA: Every state is reachable from initial state. Not-complete DFA

"The number of states in the DFA is m , where: $1 \leq m \leq 2^n$ "

Non-Deterministic Finite Automata (NFA)

(without ϵ -moves)

"Every DFA is NFA but NFA need not be DFA."

01. $\delta: Q \times \Sigma \rightarrow 2^Q$

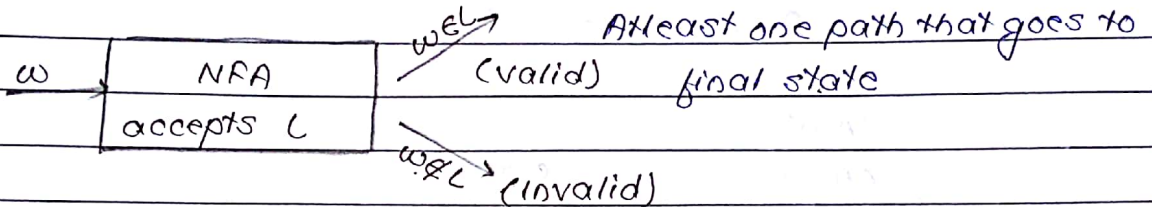
$2^Q = P(Q)$

Eg. $Q = \{A, B\}$, $2^Q = \{\emptyset, \{A\}, \{B\}, \{A, B\}\}$

02. From each state, for every input state, zero or more transition

03. For every string, No. of paths ≥ 0 .

04.

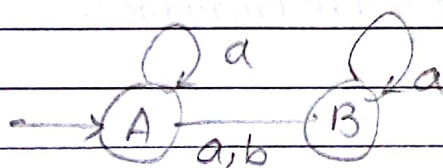


No path halts at final.

No path or every path goes to non-final.

05. For every valid strings, # paths ≥ 1

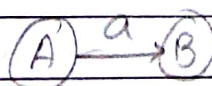
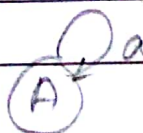
for every invalid string, # paths ≥ 0 .



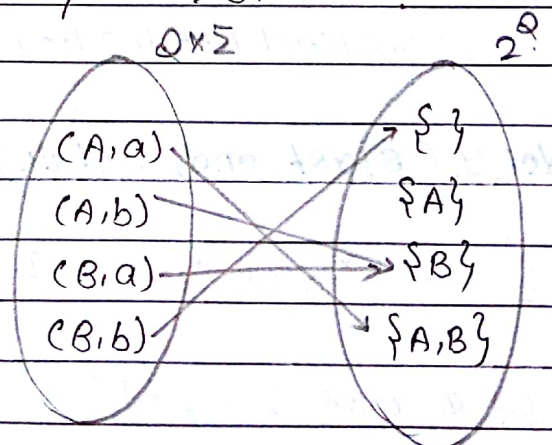
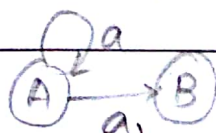
$\delta(A, a) = \{A\}$ $\delta(A, a) = \{B\}$



$\delta(A, a) = \{A\}$

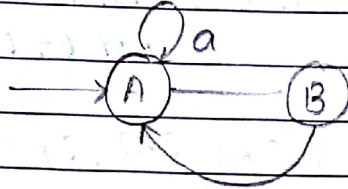


$\delta(A, a) = \{A, B\}$



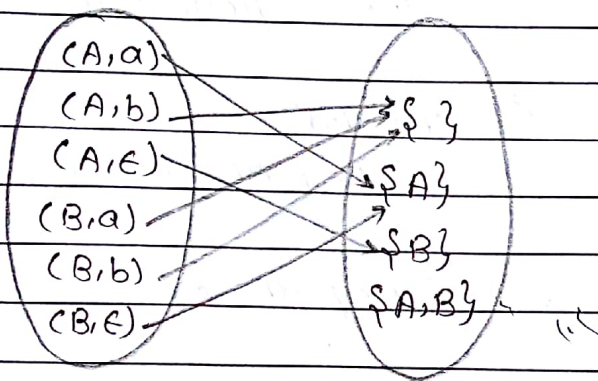
- E-NFA (Non-Deterministic Finite Automata - with E-moves)

$$S: Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$$



- without reading symbols . by reading " ϵ "
- From each state, for every input symbol, zero or more transitions and input symbol.

$$Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$$

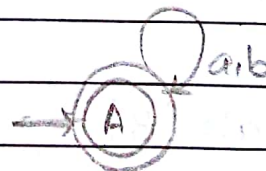


Construct NFA:

Construct NFA for the following regular languages.

Model 4 (start, end, contain)

1. $L = (a+b)^*$ over $\Sigma = \{a, b\}$

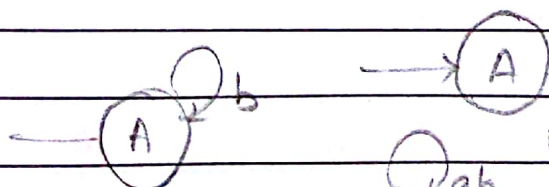


$\epsilon: A$

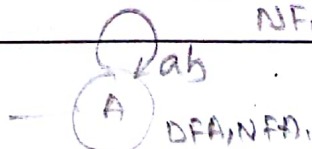
$a:$

$$A \xrightarrow{a} A$$

2. $L = \emptyset$ over $\Sigma = \{a, b\}$



NFA

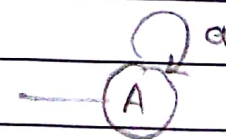


DFA, NFA.

$\epsilon: A$

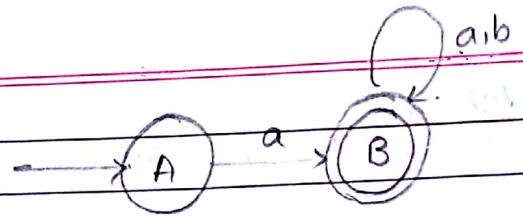
$$a: A \xrightarrow{a} ?$$

NFA but not DFA.

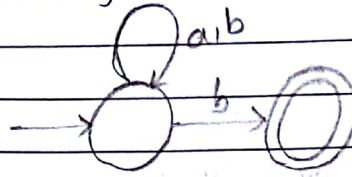


NFA (but not DFA)

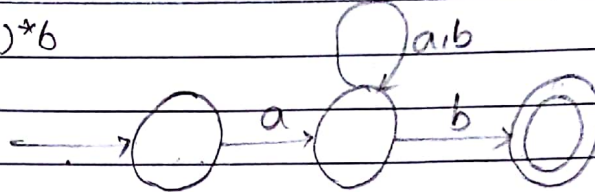
3. $L = a(a+b)^*$
 $= \{aw \mid w \in (a+b)^*\}$



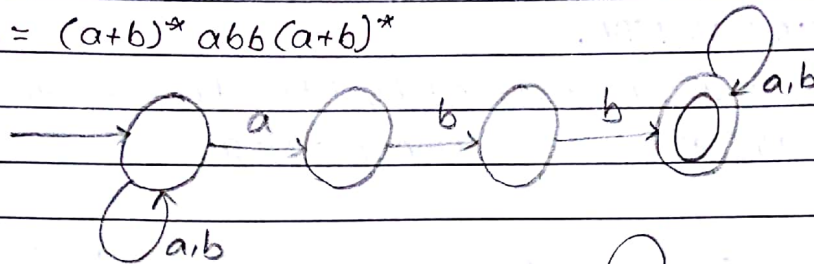
4. $L = (a+b)^*b$



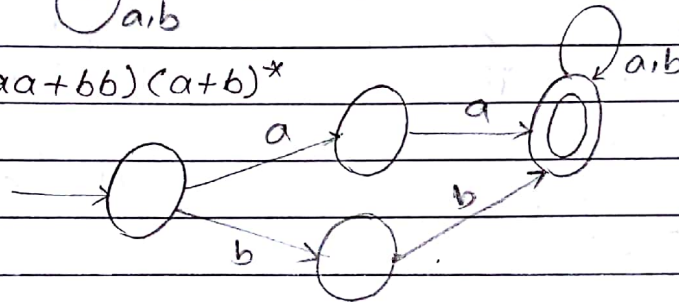
5. $L = a(a+b)^*b$



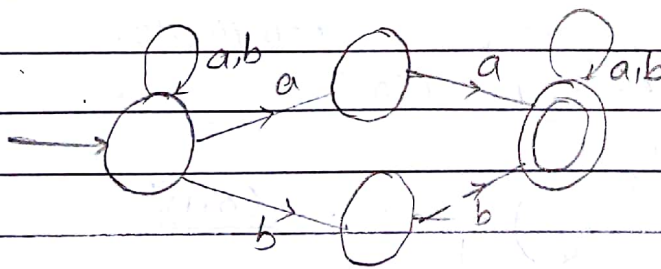
6. $L = (a+b)^*abb(a+b)^*$



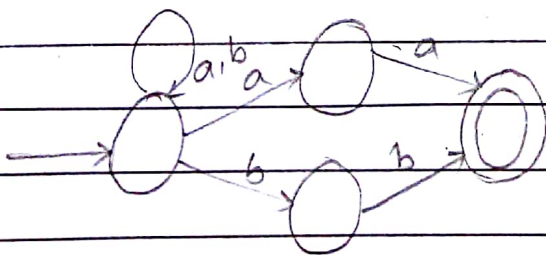
7. $L = (aa+bb)(a+b)^*$



8. $L = (a+b)^*(aa+bb)(a+b)^*$

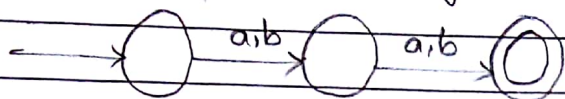


9. $L = (a+b)^*(aa+bb)$

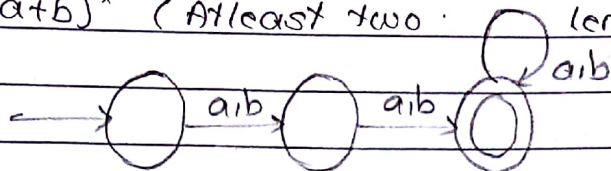


Model 02:

10. $L = (a+b)^2 =$ Exactly two length.

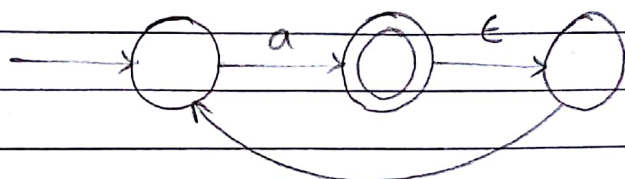


11. $L = (a+b)^2(a+b)^*$ (Atleast two length strings)



Model Questions of NFA.

01. Complement of language accepted by NFA shown below?
 $\Sigma = \{a\}$ and ϵ is an empty string.

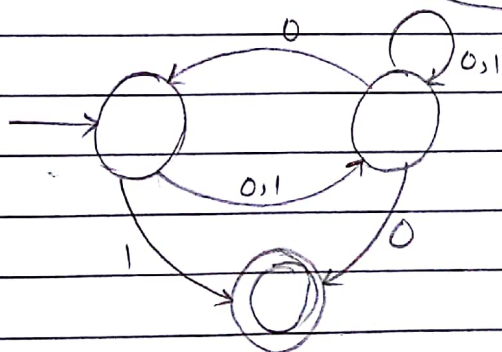


$L = \{a, aa, \dots\}$

$L = \{a^+\}$

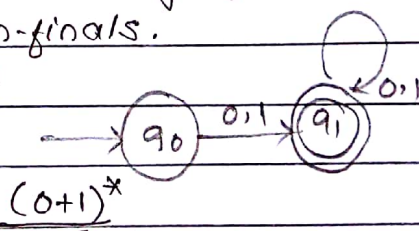
$\therefore L' = \{\epsilon\}$

02.



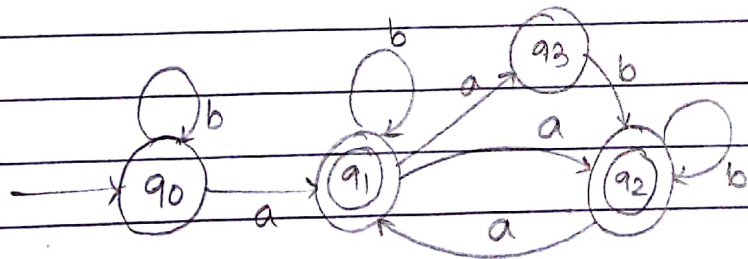
M' obtained by interchange finals & non-finals.

$L(M') = ?$



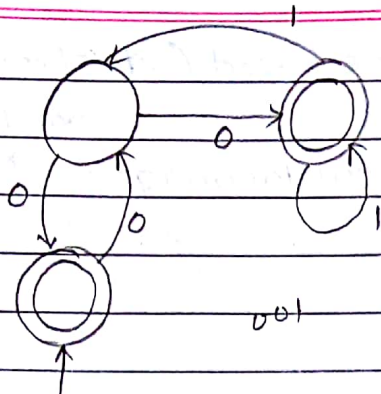
$(0+1)^*$

03.

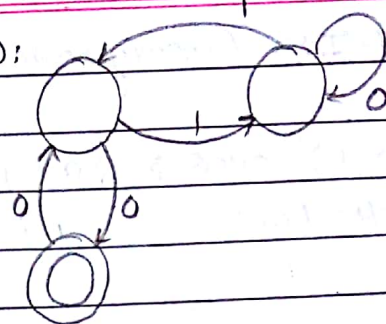


c. $b^*a(a+b)^*$

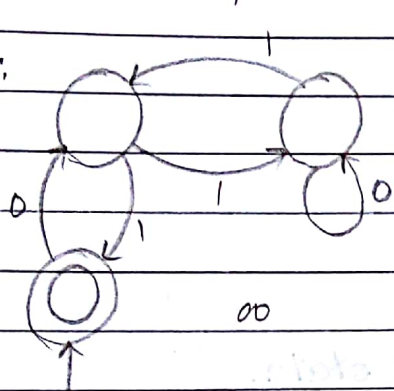
04. (P):



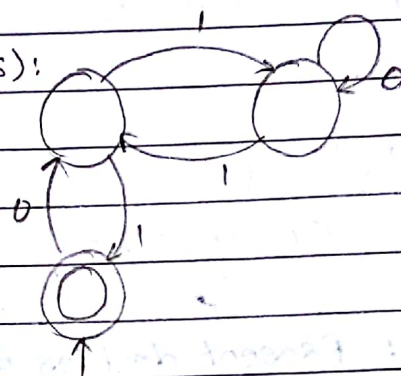
(Q):



(R):



(S):



1. $\epsilon + 0(01^*1 + 00)^*01^*$ - P

2. $\epsilon + 0(10^*1 + 00)^*0$

3. $\epsilon + 0(10^*1 + 10)^*1$ - R

4. $\epsilon + 0(10^*1 + 10)^*10^*$

c. P-1, Q-2, R-3, S-4.

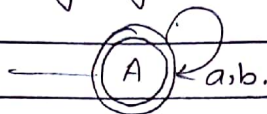
Construction of NFA:

1. All DFA's depends on minimum string.
2. If language has empty string, initial state must be final.
3. If any invalid combination appears, goto dead state.
4. Starting conditions (starts, particular position from begin) can go to dead state.
5. Ending & contain conditions never involve dead state.

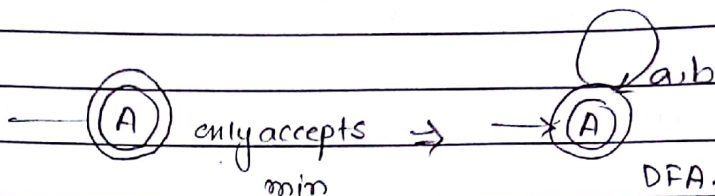
Model I: Purely minimum string based (start, end, contain conditions)

1. $L = (a+b)^*$ over $\Sigma = \{a, b\}$ (universal language = Σ^*)

method 01: $\min = \epsilon$



method 02: $\min = \epsilon$



i. $A \xrightarrow{a} \boxed{}$
 $a \in L$ goto final

Case I: Present decides $\in L$ Goto final state.

ii. $A \xrightarrow{b} \boxed{}$
 $b \in L$ goto final

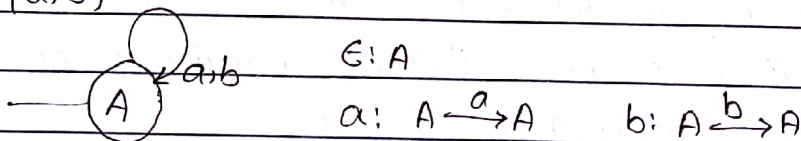
Step 1: Take min string

Step 2: Design a machine that accepts only min string.

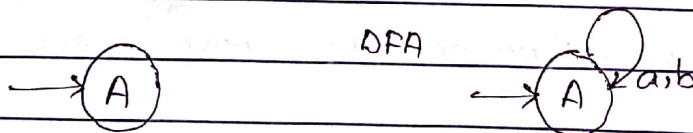
Step 3: Find all missing transitions to make DFA.

2. $L = \emptyset$ over $\Sigma = \{a, b\}$

method 01:



method 02:



i. $A \xrightarrow{a} \boxed{} \Rightarrow \text{final}$
 present future
 $a \in L$

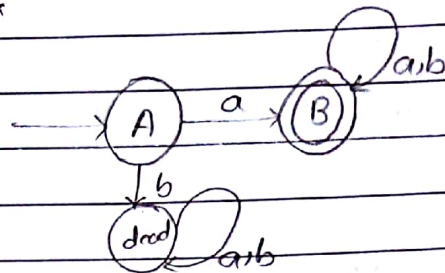
ii. $A \xrightarrow{b} \boxed{?} \Rightarrow \text{final}$
 $b \notin L$ future no future
 exist goto dead state

Case II. If present & future not able to decide $\in L$ then goto Dead state.

3. $L = a(a+b)^*$ over $\Sigma = (a+b)^*$

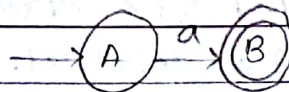
method 01:

min a



method 02:

min a



i. $A \xrightarrow{b} [?] \mid$
 $b \notin L$

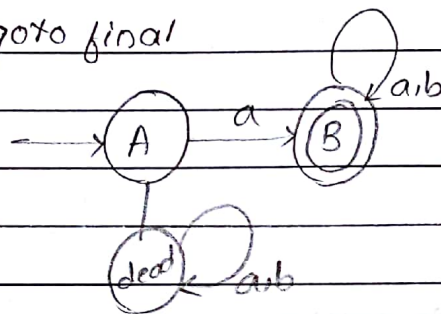
present, future not able to decide goto dead.

ii. $a \xrightarrow{a} B \mid [?] \mid$
 $aa \in L$

goto final

iii. $a \xrightarrow{b} B \mid [] \mid$
 $ab \in L$

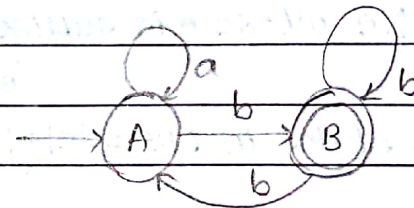
goto final



4. $L = (a+b)^*b$

method 01:

min b



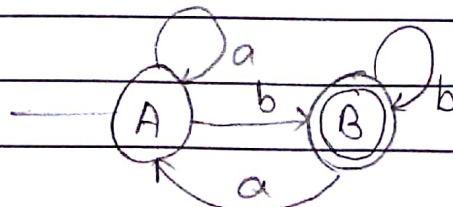
Case III: If present not able to decide but future able to decide then find out who is reading that future and go there.

i. $A \xrightarrow{a} [] \mid$
 $a \in L$

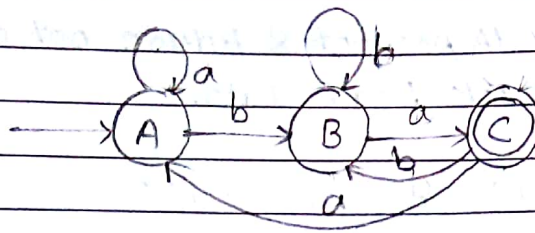
$A \xrightarrow{a} [?] \xrightarrow{b} \text{final}$
 $ab \in L$
 $= A.$

ii. $b \xrightarrow{a} B \xrightarrow{a} [?] \mid$
 $bab \in L$

iii. $b \xrightarrow{b} B \xrightarrow{b} ?$
 $bb \in L.$



5. $L = (a+b)^* aba$.
min: aba



6. $L = abb(a+b)^*$

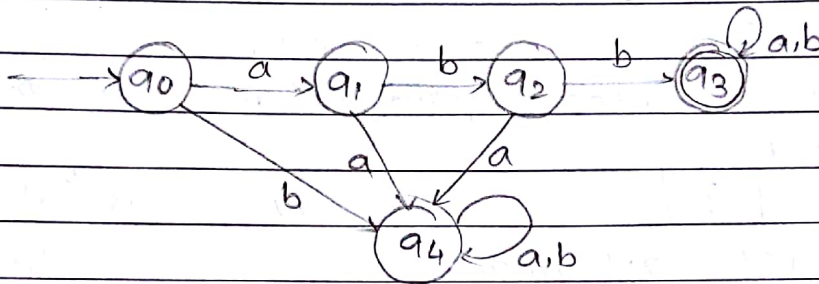
min string: abb

4 states

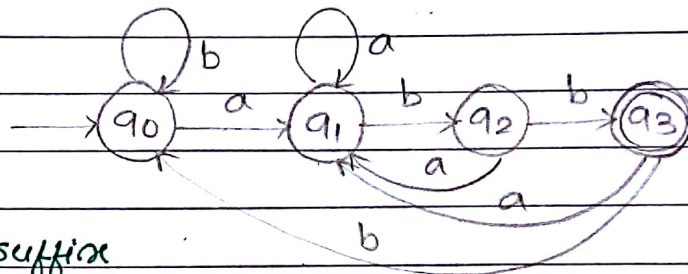
1 dead state

$\therefore$ Total: 5 states

$L = sX$ where s is prefix (start condition)
 $\Downarrow$ No. of states in DFA = $(|s|+1) + 1$ of dead state



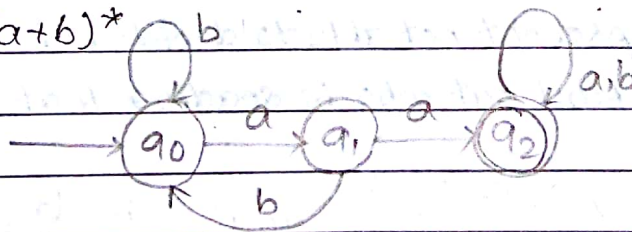
7. $L = (a+b)^* abb$



$L = Xe$ where e is suffix
 $\Downarrow$ (end condition)

No. of states in DFA = $|e|+1$.

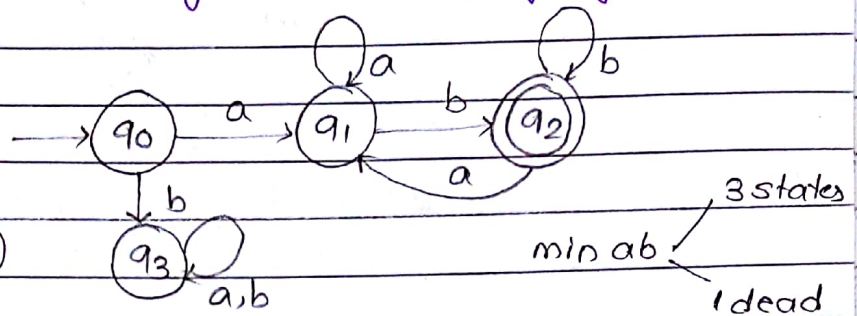
8. $L = (a+b)^* aa(a+b)^*$



Model 2: (start/end combinations) (Depends on min string and overlapping symbols)

9. $L = a(a+b)^*b$

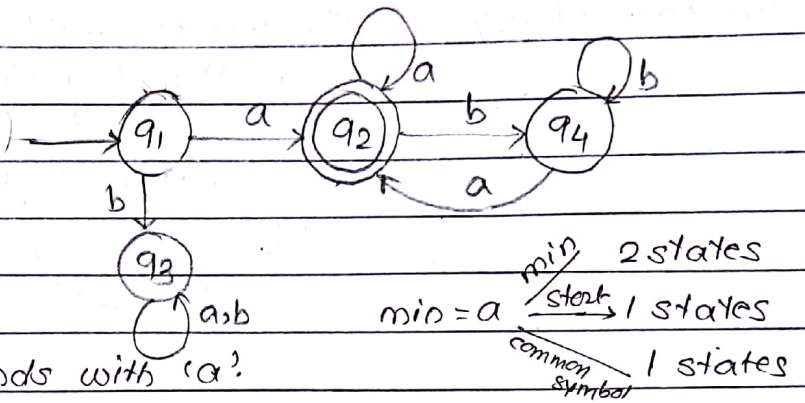
(Each string starts with 'a' and ends with 'b')



Total states: 4.

10. $L = a(a+b)^*a$

min = (a (common symbol))
will be duplicated.



Total states: 4.

11. Each string starts with 'a' and ends with 'ab'.
 $ab + a(a+b)^*ab$

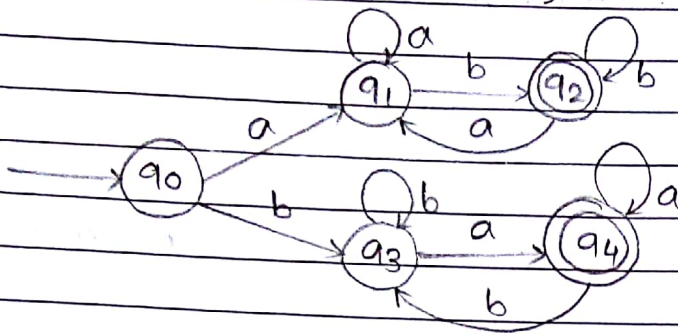
min ab $\begin{matrix} \text{min} \\ \text{start} \\ \text{common} \end{matrix}$ $\begin{matrix} 3 \text{ states} \\ 1 \text{ states} \\ 1 \text{ states} \end{matrix}$

Total: 4 states

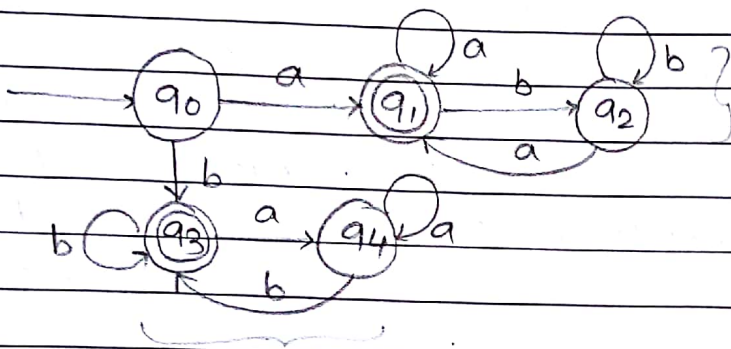
12. Each string starts with 'aba' and ends with 'bab'.
 $abab + aba(a+b)^*bab$

min = abab $\begin{matrix} \text{min} \\ \text{start} \\ \text{ns. (ba)} \end{matrix}$ $\begin{matrix} 5 \\ 1 \\ 2 \end{matrix}$
8

13. Each string starts and ends with different symbols.
 $a(a+b)^*b + b(a+b)^*a$



14. Each string starts and ends with same symbols.
 $a + a(a+b)^*a + b(a+b)^*b + b.$



ends with 'a'

ends with b.

$$\begin{aligned} q_2 &= a \times b \\ q_4 &= b \times a. \end{aligned}$$

Union = Σ^*

$$\begin{aligned} q_0 &= \epsilon \\ q_1 &= a + a \times a \\ q_3 &= b + b \times b. \end{aligned}$$

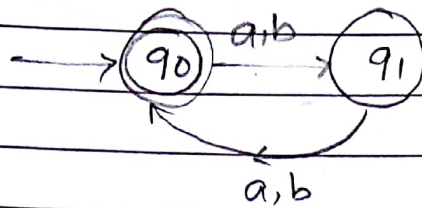
Note:

If DFA contain every state as final then language accepted by such DFA is universal language.

If every state is non-final in DFA, then it accepts empty language.

Model 3: (Length)

15. Each string has even length over $\Sigma = \{a, b\}$
 set of all even length strings
 $\{w \mid w \in (a+b)^*, |w| = \text{even}\}$



$q_0 = \text{remainder } 0$

$q_1 = \text{remainder } 1$

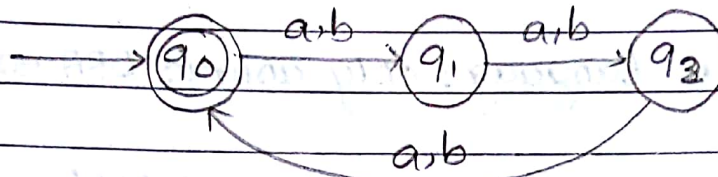
16. Length of $|w| = 0 \text{ modulo } 3$

$\{w \mid w \in (a+b)^*, |w| = 0 \text{ modulo } 3\}$

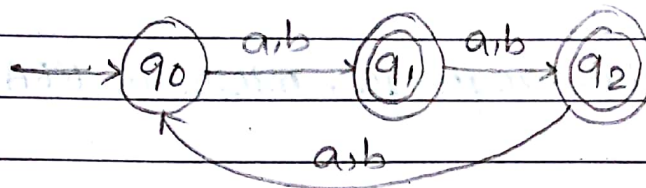
$q_0: \text{remainder } 0$

$q_1: \text{remainder } 1$

$q_2: \text{remainder } 2$



17. $\{w \mid w \in (a+b)^*, |w| \text{ is not divisible by } 3\}$



Any Regular Language (L)

$\Downarrow$

DFA accepts language

$\Downarrow$ Interchange finals and non-finals

modified DFA

$\Downarrow$

Accepts language's complement ($\bar{L}$)

Complement of every regular language is always regular language because both language have DFA.

18. $\{w \mid w \in (a+b)^*, |w| = 2\}$

Set of all 2 length strings.

If $|w| = n \Rightarrow$ No. of states in DFA = $n+2$
(minimum)

$\Rightarrow$ No. of states in (min) NFA = $n+1$

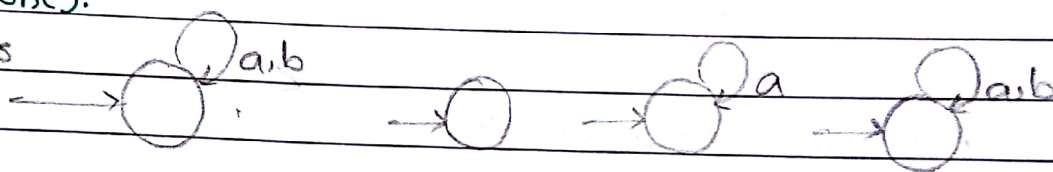
Note:

For every Regular Language only unique DFA exist.

Regular Language but only one mini $\left\{ \begin{array}{l} \rightarrow \text{Infinite equivalent DFA's} \\ \rightarrow \text{Infinite equivalent regular expression} \\ \rightarrow \text{Infinite equivalent NFA's} \end{array} \right.$

Note: For a given regular language, minimum NFA need not be unique. (one).

minimum NFA's



• No. of states in Finite Automata $\Rightarrow$ No. of states in DFA

• No. of states in min. Finite Automata $\Rightarrow$ No. of states in min. DFA
Minimum no. of states in DFA.

19. $\{w \mid w \in (a+b)^*, |w| \leq 2\}$

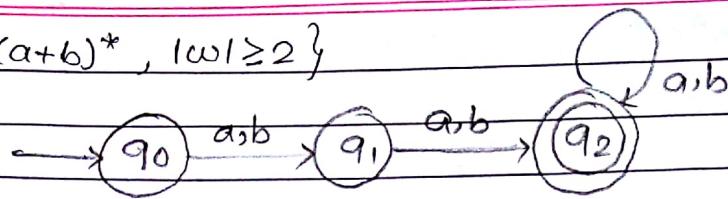
length of each string is at most 2.



If $|w| \leq n \Rightarrow$ in DFA : $(n+2)$ states

in NFA : $(n+1)$ states

20. $\{w \mid w \in (a+b)^*, |w| \geq 2\}$



If $|w| \geq n \Rightarrow$ In DFA : $(n+1)$ states

In NFA : $(n+1)$ states

Note:

If $|w|$ is divisible by $n \Rightarrow$ In DFA, n states.

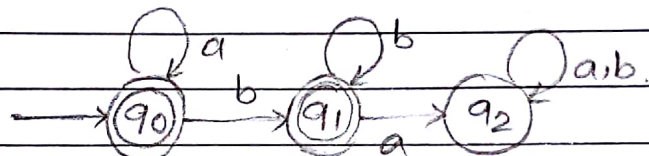
Model 4: Sequence

21. $L = a^*b^*$ (Any no. of a's followed by any no. of b's)

$[A] = a^*$

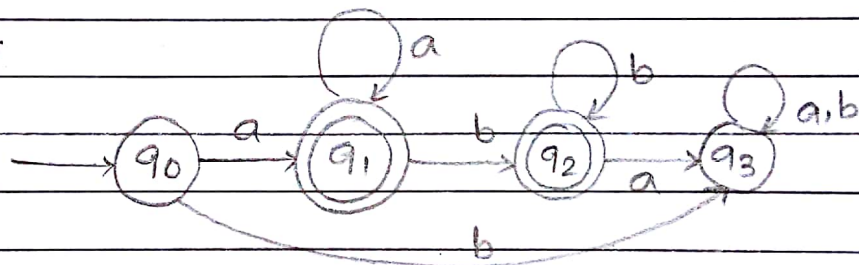
$[B] = b^*$

Dead state: $a^*bb^*a(a+b)^*$

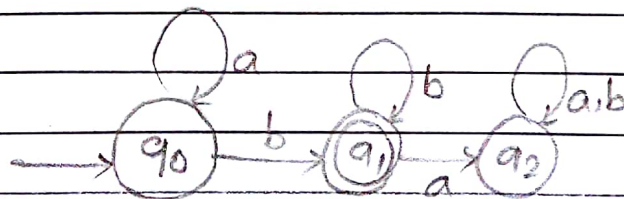


22. $L = a^+b^*$

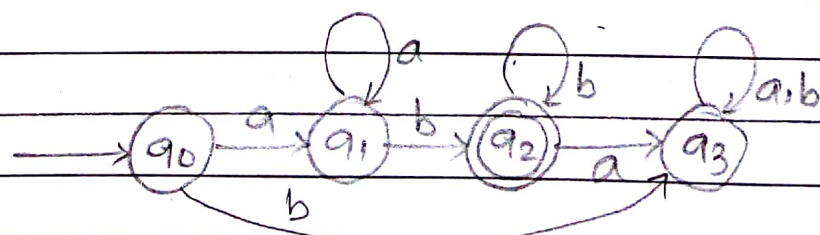
min = a



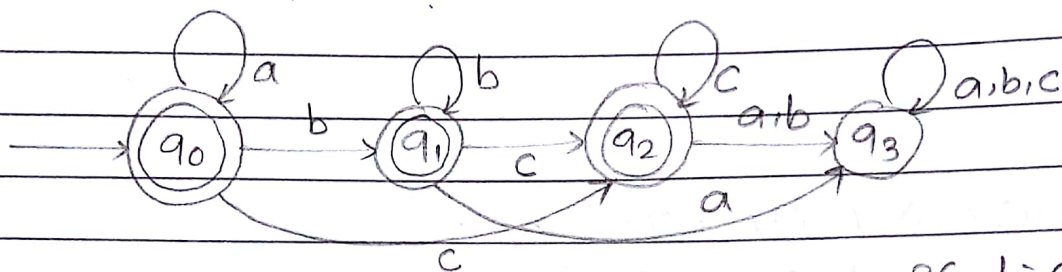
23. $L = a^*b^+$



24. $L = a^+b^+$



25. $L = a^*b^*c^* \quad \{\epsilon, a, b, c, ab, bc, ac, \dots\}$



26. $L = a^+b^*c^*$

27. $a^*b^+c^*$

28. $a^*b^*c^+$

29. $a^+b^+c^*$

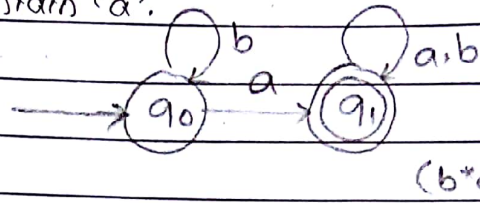
30. $a^+b^*c^+$

31. $a^*b^+c^+$

32. $a^+b^+c^+$

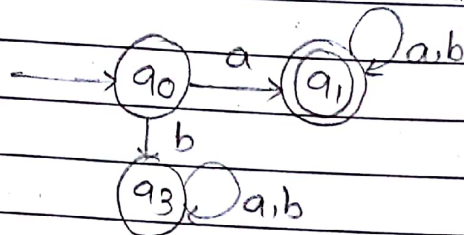
Model 5: (Subset / Super set) (Two conditions will be converted to one condition)

33. Each string starts with 'a' or contain 'a'.
(contain 'a' only)

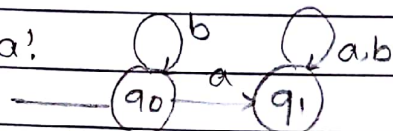


$(b^*a(a+b)^*)$

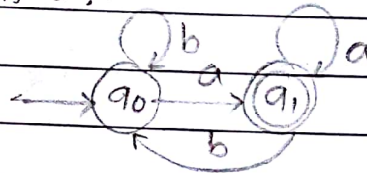
34. Each string starts with 'a' and contain 'a'.



35. Each string ends with 'a' or contain 'a'.



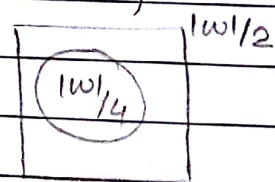
36. Each string ends with 'a' and contain 'a'.



37. $\{w | w \in (a+b)^*, |w| \equiv 0 \pmod{2} \text{ or } |w| \equiv 0 \pmod{4}\}$

$|w| \equiv 0 \pmod{2}$

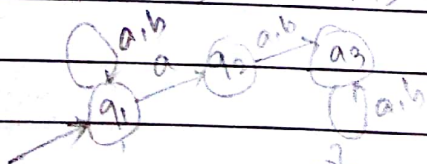
$\therefore$ No. of states: 2



38. $\{w | w \in (a+b)^*, |w| \equiv 0 \pmod{2}, |w| \equiv 0 \pmod{4}\}$
same as $|w| \equiv 0 \pmod{4}$

$\therefore$ No. of states = 4

$$L(|w|/2) \cap L(|w|/4) = L(|w|/4)$$

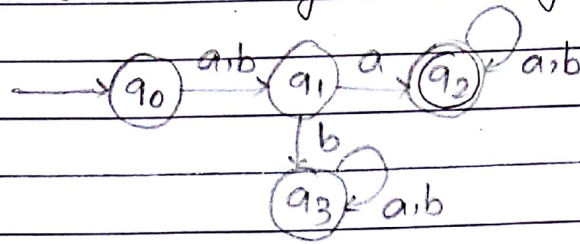


$\{ (a+b)^* a (a+b)^* \}$

Model 6: Position based.

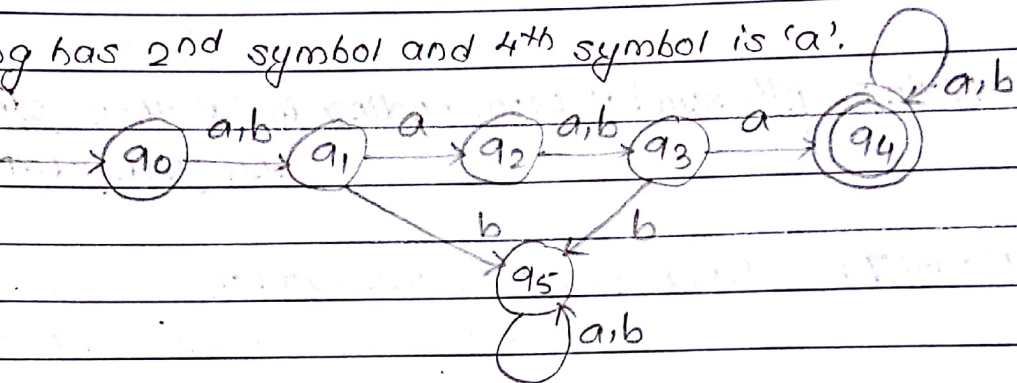
39. Set of all strings where each string has 2nd symbol is 'a'.

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



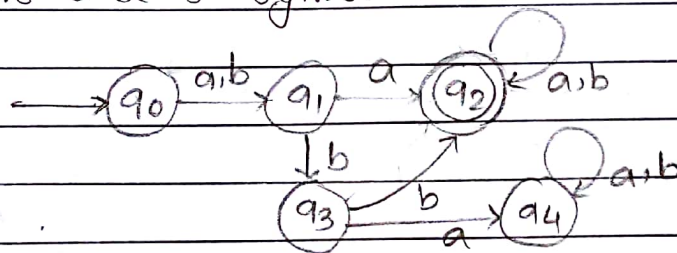
40. Each string has 2nd symbol and 4th symbol is 'a'.

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

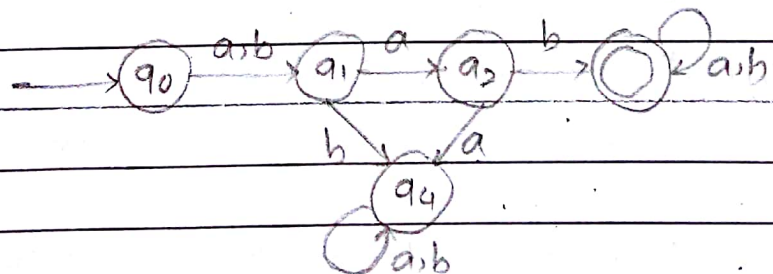


41. 2nd symbol is 'a' or 3rd symbol is 'b'.

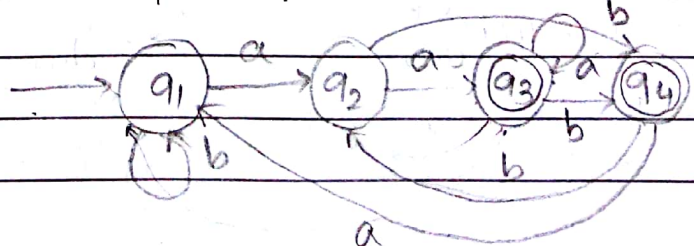
$$a(a+b)^* + a(b+b)^*$$



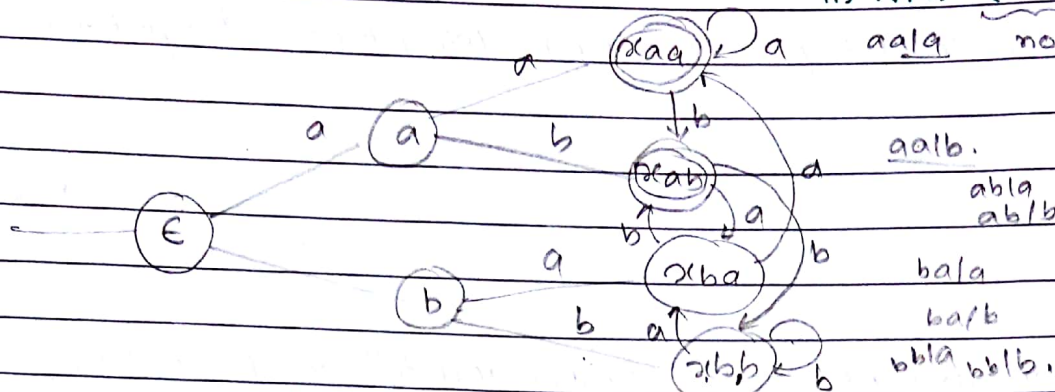
42. Set of all strings where each string has 2nd symbol is 'a' and 3rd symbol is 'b'.



43. 2nd symbol from ending is 'a'. $\{wax \mid w \in (a+b)^*, x \in (a+b)^*\}$



Note: n^{th} symbol from beginning is 'a' $\Rightarrow$ In DFA: $(n+2)$ states
In NFA: $(n+1)$ states

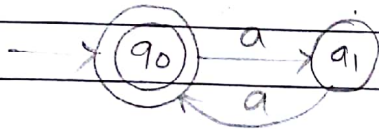


Note: n^{th} symbol from ending is 'a' then 2ⁿ states in DFA
and $(n+1)$ states in NFA.

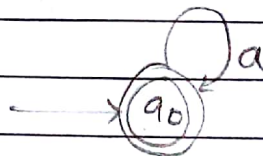
Model 3: (Language over one symbol)

It should form Arithmetic Progression.

44. $L = (aa)^* \{ \epsilon, aa, aaaa, \dots \}$ $\Sigma = \{a\}$
 $(a^2)^*$
 $(a^2)^n$

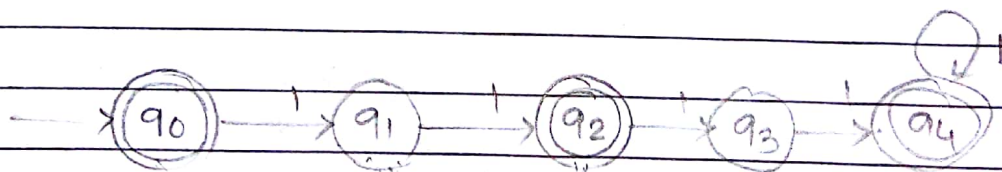


45. $L = a^* \{ a^0, a^1, a^2, \dots \}$

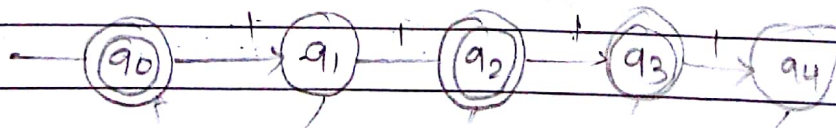


Note: $(a^2)^* \neq (a^*)^2 = a^* \cdot a^* = a^*$

46. $L = (11 + 11111)^* \{ \epsilon, 12, 14, 15, \dots \}$



47. $L = (111 + 11111)^* = \{ 1^0, 1^3, 1^5, 1^6, 1^8, 1^9, 1^{10}, 1^{11}, \dots \}$



48. $L = (11 + 1111)^* = (11)^*$

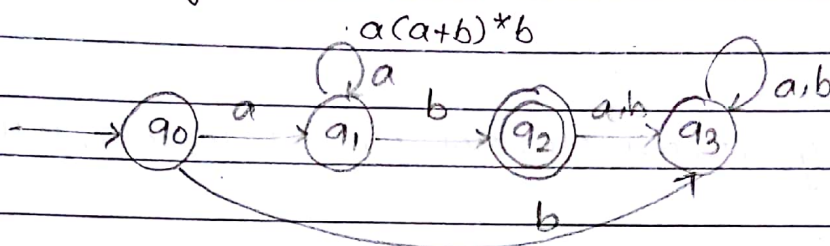
49. $L = (1 + 111)^* = 1^*$ $\rightarrow (q_0) \rightarrow 1$

50. $L = (1111 + 11111)^* = \{ \epsilon, 14, 15, 18, 19, 110, 112, 113, \dots \}$

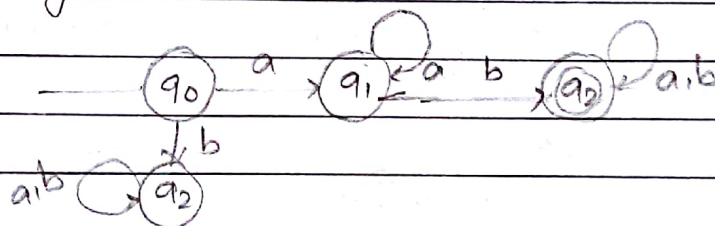
13 states

Model 8: (more conditions but creates sequence)

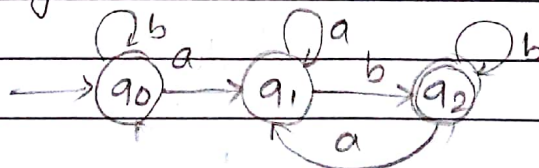
51. where each string ends with 'b' and starts with 'a'.



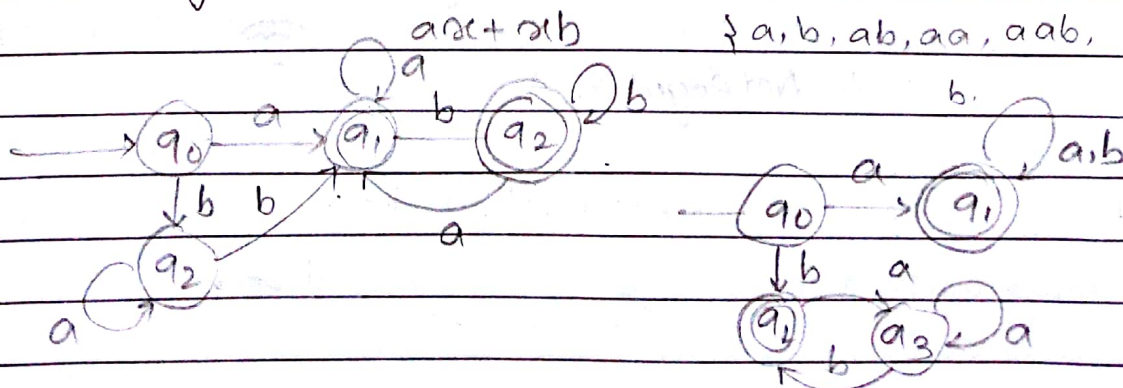
52. where each string starts with 'a' and contain 'b'.



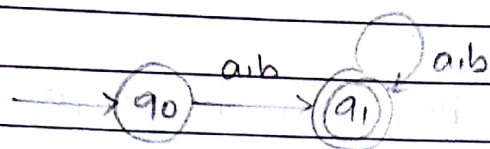
53. where each string ends with 'b' and contain 'a'.



54. where each string starts with 'a' or ends with 'b'.



55. where each string starts with 'a' or contains 'b'. $(a+b)^+$ equal. $\uparrow$
ends with 'a' or contains 'b'.



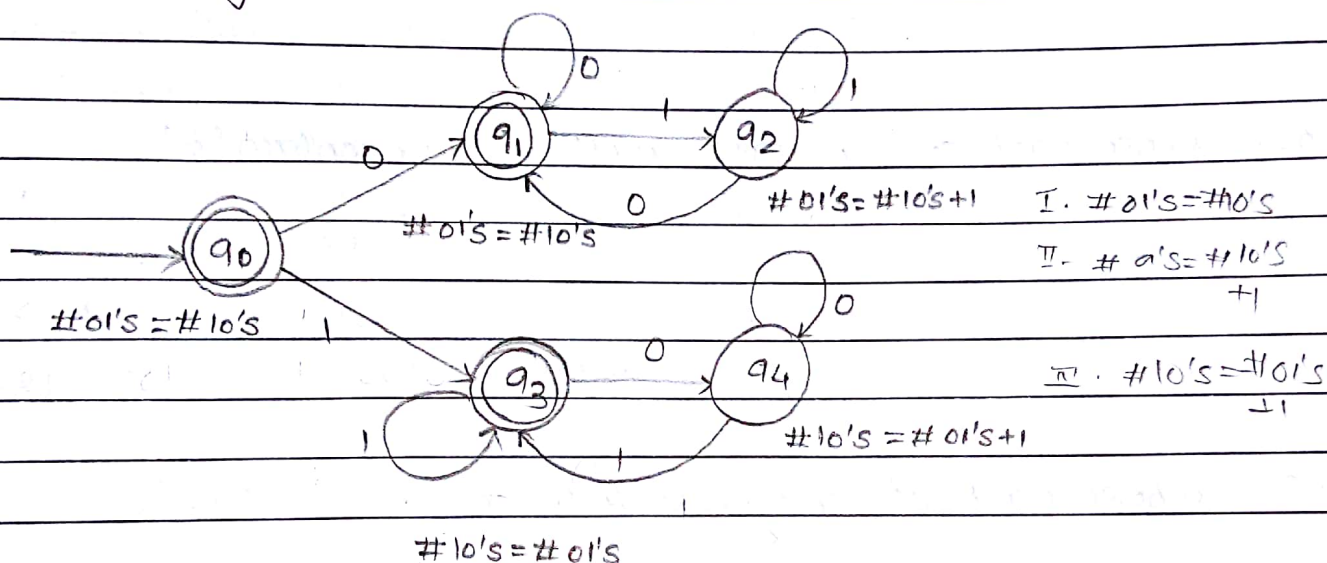
Model 9: Pattern Based

(57) $\{w \mid n_{01}(w) = n_{10}(w); w \in (0+1)^*\}$

57. $\{w \mid n_{01}(w) = n_{10}(w); w \in (0+1)^*\}$

$\{w \mid n_{01}(w) = n_{10}(w), w \in (0+1)^*\}$ — impossible to design FA.

valid string = $\{\epsilon, 0, 1, 010, 0101, 0100001, 1000010, 10111110, \dots\}$



58. $\{w \mid n_{01}(w) = n_{10}(w), w \in (0+1)^*\}$

Impossible to design finite Automata.

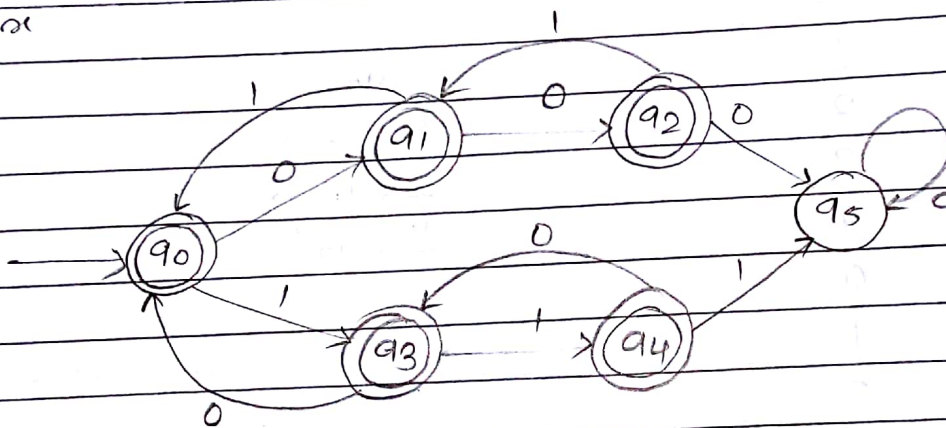
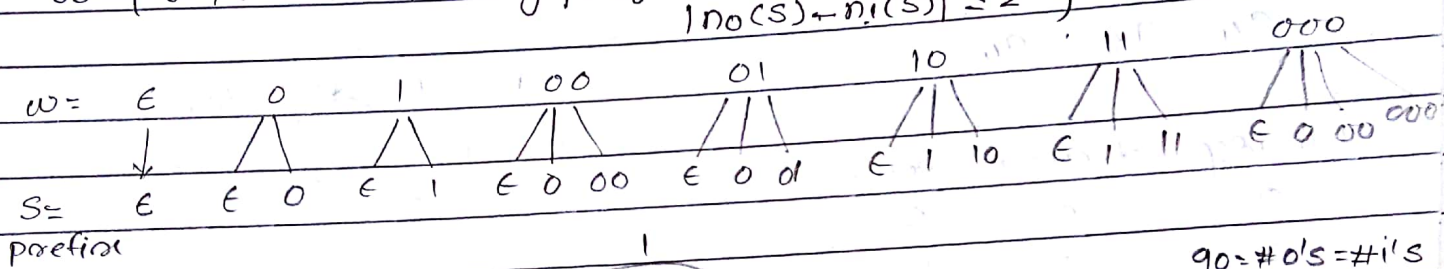
$\therefore$ Not-Regulate.

59. $\{w \mid n_{011}(w) = n_{110}(w), w \in (0+1)^*\}$
 011011, 01100000111, 0111111011, 110110, 1100000110

$\therefore$ Language is regular.

Model 10: Prefix Based.

60. $\{w \mid w \in (0+1)^*, \text{ every prefix } s \text{ of } w \text{ satisfies condition } |n_0(s) - n_1(s)| \leq 2\}$



$q_0: \#0's = \#1's$
 $q_1: \#0's = \#1's + 1$
 $q_2: \#0's = \#1's + 2$
 $q_3: \#1's = \#0's + 1$
 $q_4: \#1's = \#0's + 2$
 $q_5: \text{dead state.}$

61. $\{w \mid w \in (0+1)^*, \text{ every prefix } s \text{ of } w \text{ satisfies condition } |n_0(s) - n_1(s)| = 0\}$

$L = \{\epsilon\}$

62. $\{w \mid w \in (0+1)^*, [n_0(w) - n_1(w)] \leq 2\}$

using Q.58

$$|n_0(w) - n_1(w)| = 0$$

$$n_0(w) = n_1(w)$$

$\therefore$ Not regular,

so no finite Automata exist.

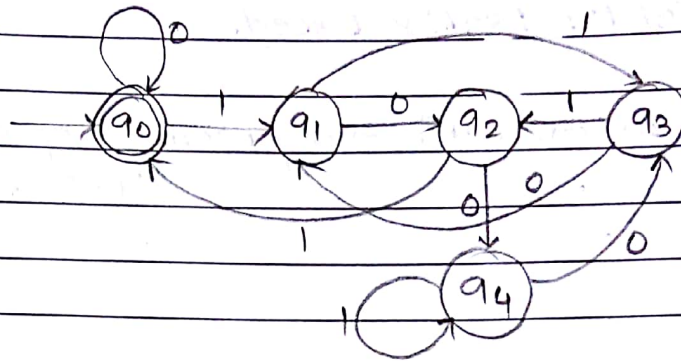
Model 11: [Binary to decimal / Integer conversions and divisible by n]

63. Decimal equivalent of each binary number is divisible by 5.

Binary alphabet.

0 1

remainder	q_0	q_0	q_1
	q_1	q_2	q_3
	q_2	q_4	q_0
	q_3	q_1	q_2
	q_4	q_3	q_4



Binary decimal rem.

0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	0
110	6	1
111	7	2
1000	8	3
1001	9	4

$q_0 \xrightarrow{1} ?$
 $\xrightarrow{1} q_1 \xrightarrow{0} ?$
 $\xrightarrow{1} q_1 \xrightarrow{1} ?$
 $\xrightarrow{10} q_2 \xrightarrow{0} ?$

divisible by k where $k=2^n$

1. Divisible by 2 $\Rightarrow$ ends with 0 $\rightarrow$ 2 states
2. Divisible by 4 (2^2) $\rightarrow$ ends with 00 $\rightarrow$ 3 states
3. Divisible by 8 (2^3) $\rightarrow$ ends with 000 $\rightarrow$ 4 states

n. divisible by $2^n \rightarrow$ ends with n zero's $\rightarrow (n+1)$ state.

divisible by $k (k=2^n) \rightarrow$ ends with $\log_2 k$ zero's $\rightarrow (\log_2 k + 1)$ states

Decimal equivalent of Binary number is divisible by k .

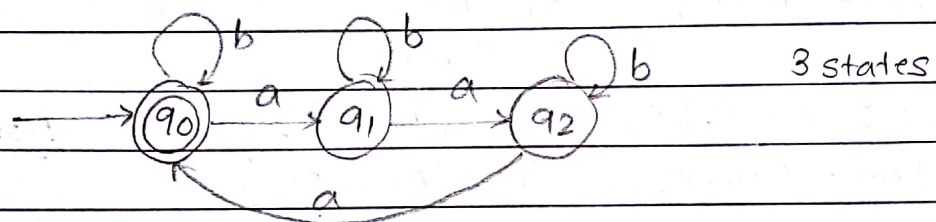
I. If k is in power of 2 $\Rightarrow$ No. of states = $\log_2 k + 1$
 $k = 2, 4, 8, 16, \dots$

II. If k is prime or odd $\Rightarrow$ No. of states = k
 $k = 3, 5, 7, 9, 11, \dots$

III. $k \neq 2^{\text{any}}$, $k \neq \text{prime}$, $k \neq \text{odd}$ then $\Rightarrow$ Apply minimization to answer after constructing with k states.

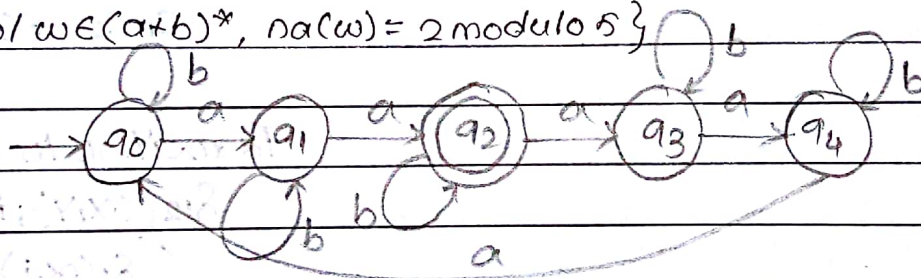
Model 12: Symbol based but divisible

64. $\{ w \mid w \in (a+b)^*, n_a(w) \text{ is divisible by } 3 \}$



3 states

65. $\{ w \mid w \in (a+b)^*, n_a(w) \equiv 2 \pmod{5} \}$



5 states.

Model 13: (Many conditions but no condition)

66. $\{ a^m b^n \mid m=n \text{ or } m < n \text{ or } m > n \} = a^* b^*$

67. $\{w | w \in (a+b)^*, w \text{ starts with 'a' or 'b'}\} = (a+b)^+$

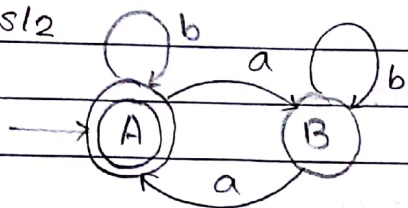
68. $\{w | w \in (a+b)^*, w \text{ ends with 'a' or 'b'}\} = (a+b)^+$

69. $\{w | w \in (a+b)^*, w \text{ contains 'a' or 'b'}\} = (a+b)^+$

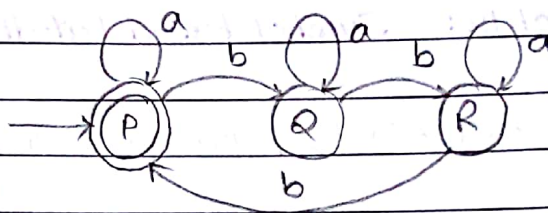
Model 14: Many conditions $\Rightarrow$ Composition (divide and conquer)

70. $\{w | w \in (a+b)^*, n_a(w) = 0 \text{ modulo } 2, n_b(w) = 0 \text{ modulo } 3\}$

FA₁: #a's/2



FA₂: #b's/3



FA₁ = (Q₁, Σ, δ₁, A, {A}) Q₁ = {A, B}

Σ = {a, b}

FA₂ = (Q₂, Σ, δ₂, P, {P}) Q₂ = {P, Q, R}

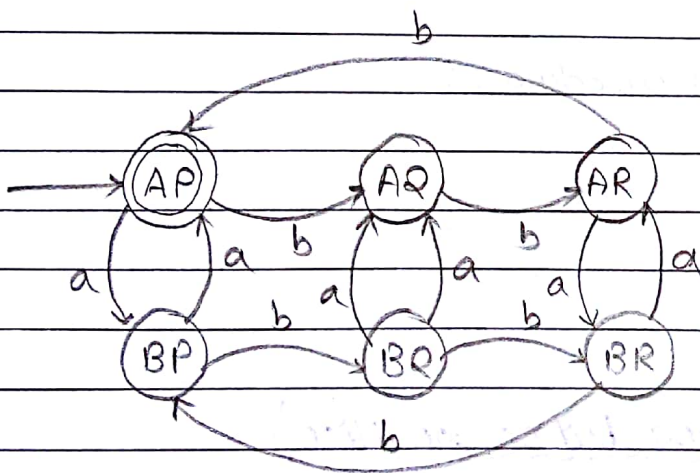
FA₁ × FA₂ = (Q₁ × Q₂, Σ, δ₁₂, (A, P), F₁₂)

Note:

C₁ → m states

C₂ → n states

C₁, C₂ → (mn) states



Transition in FA₁ × FA₂

$$\delta_{12}((x, y), i) = (\delta_1(x, i), \delta_2(y, i))$$

#a's/2 and #b's/3

Final states = {(final, final)} = {(A, P)}

71. #a's/2 or #b's/3

Final states = {(final, non-final), (non-final, final), (final, final)}
= {(AQ), (AR), (BP), (AP)}

72. $\#a's/2$ but not $\#b's/3$

$$\text{Final states} = \{(\text{final}, \text{non-final})\} = \{(AQ), (AR)\}$$

73. $\#b's/3$ but not $\#a's/2$

$$\text{Final states} = \{(\text{non-final}, \text{final})\} = \{(BP)\}$$

$$aobob \Rightarrow n_a(w) = 0 \text{ modulo } 2$$

$$n_b(w) = 0 \text{ modulo } 3$$

$$aibj = n_a(w) = i \text{ modulo } 2$$

$$n_b(w) = j \text{ modulo } 3.$$

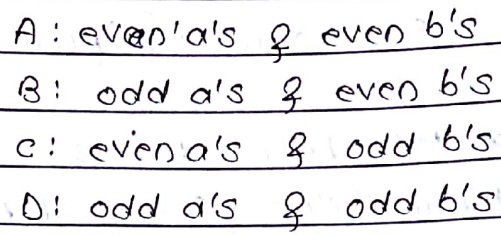
74. $\{w \mid w \in (a+b)^*, n_a(w) = 1 \text{ modulo } 2, n_b(w) \neq 0 \text{ modulo } 3\}$

Note: • Union of two regular languages is always regular language.

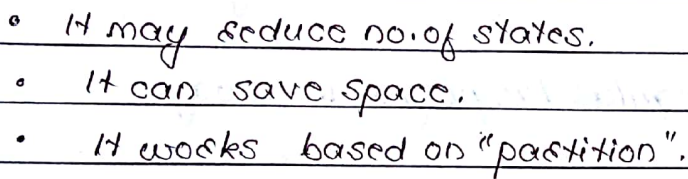
• Intersection of two regular languages is always regular language.

• Difference of two regular language is also regular.

$FA1 \cup FA2$



Requires 'state equivalence' & 'states distinguishable'

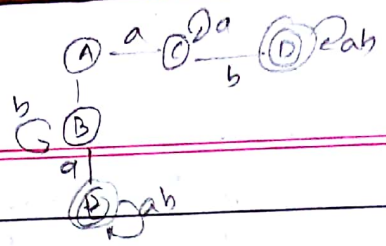


- $A_1 \cup A_2 \cup A_3 \cup \dots \cup A_n = A$
- Every 2 subsets are disjoint (intersection is empty)

Eg: $A = \{1, 2, 3, 4\}$

$\cdot \{1, 2, 3, 4\}$ $\cdot 1, 234$ $\cdot 12, 34$ $\cdot 14, 23$ $\cdot 1, 2, 34$ $\cdot 1, 2, 3, 4$
 $\cdot$ $2, 341$ $13, 24$ $1, 3, 24$
 $3, 241$ $1, 4, 23$
 $4, 231$ $2, 3, 14$
 $2, 4, 13$
 $3, 4, 12$

Scanned by CamScanner



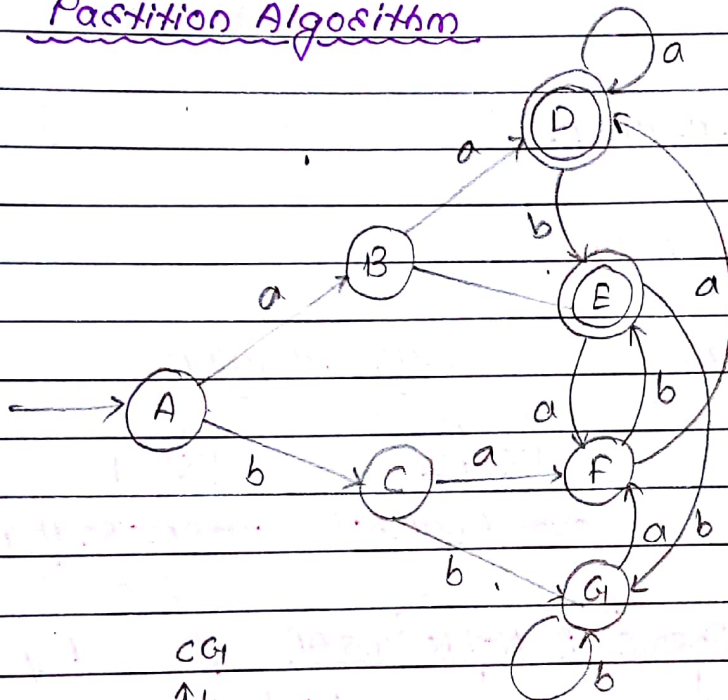
Equivalent States

A and B are equivalent states iff $\forall$ strings $(\forall w \in \Sigma^*)$, both halts at final or both halts at non-final.

Distinguishable States

A and B are distinguishable state iff one halts at final and other halts at non-final.

Partition Algorithm

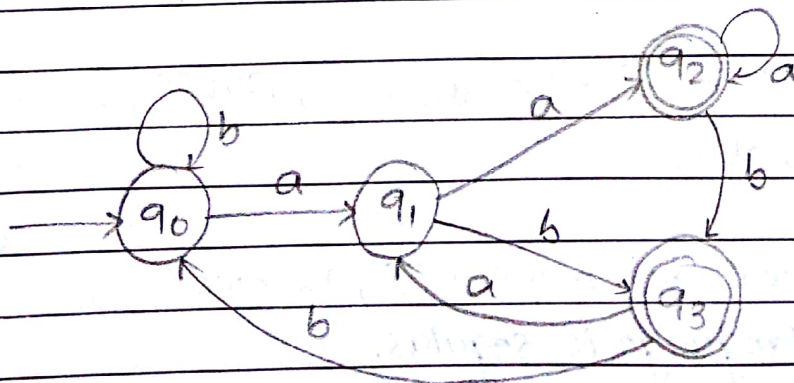
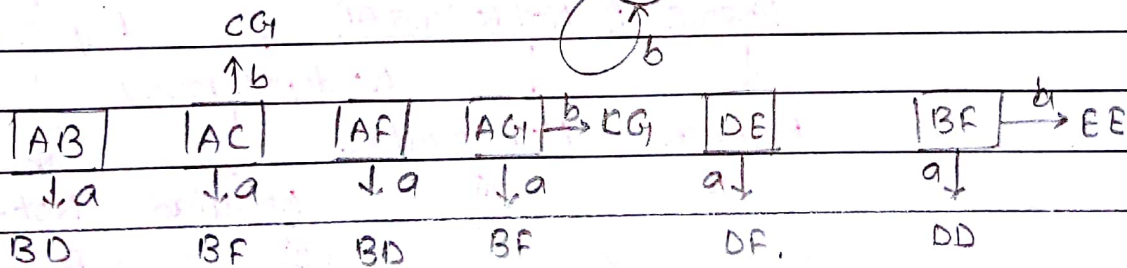


Step 1: $\{A, B, C, D, E, F\}$ $\{D, E\}$
 $|w|=0$ Set of all non-final Set of all final.

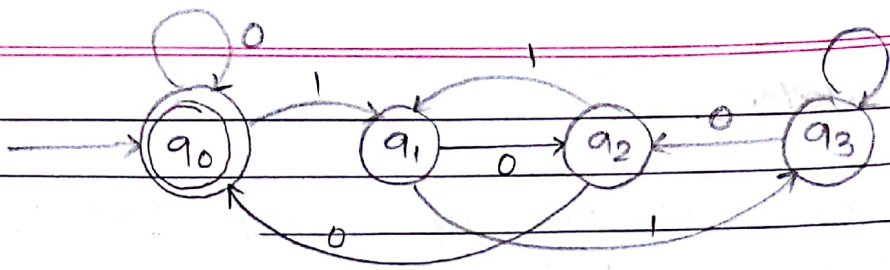
Step 2: $\{A, C, G\}$ $\{B, F\}$ $\{D\}$ $\{E\}$
 $|w|=1$

Step 3: $\{A, C, G\}$ $\{B, F\}$ $\{D\}$ $\{E\}$
 $|w|=2$ q_0 q_1 q_2 q_3

A, C, G are equivalent.
 B, F are equivalent.



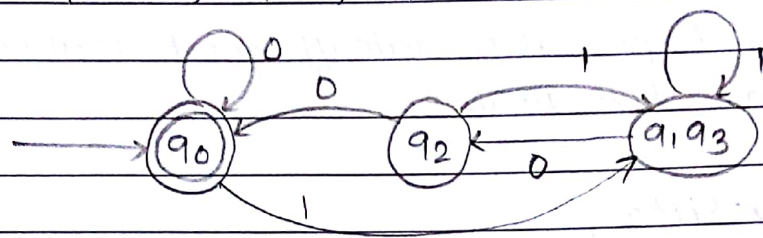
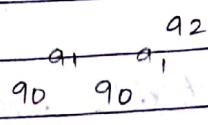
Q.



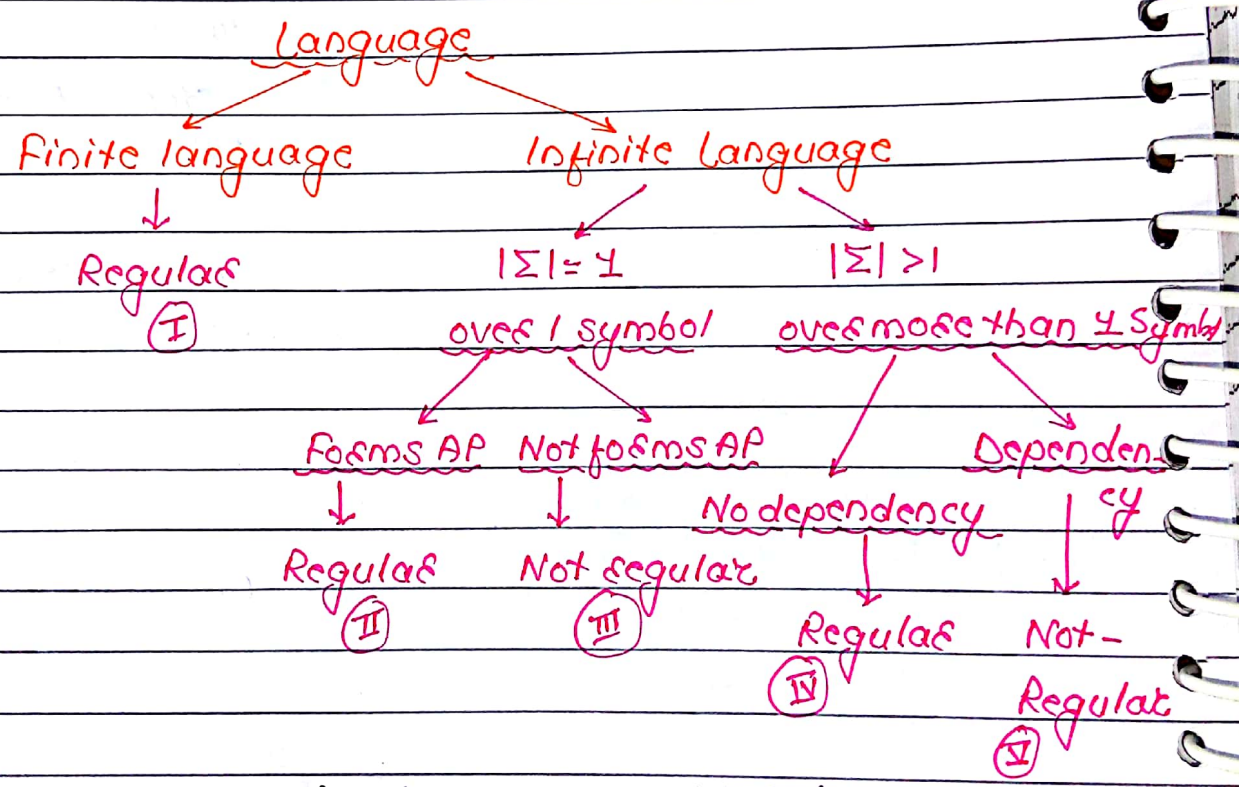
Step 1: $\{q_1, q_2, q_3\}$ $\{q_0\}$

Step 2: $\{q_1, q_3\}$ $\{q_2\}$ $\{q_0\}$

Step 3: $\{q_1, q_3\}$ $\{q_2\}$ $\{q_0\}$



Identification of Regular Language



$\{a^n b^n \mid n < 100\} = \{\epsilon, ab, \dots, a^{99}b^{99}\}$
 $\{a, abb\}$

$\{w \mid n_a(w) = n_b(w), w \in (a+b)^*, |w| < 100\}$

Note: Every finite language is regular.

II. $|Σ|=1$ and forms AP.
 a^{2n}
 a^{100n+3}

III. $|Σ|=1$ and not form AP.
 a^n , $a^{2n}2^n$, a^{prime}

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

V. $a^n b^n$
 $\{w \mid n_0(w) = n_1(w), w \in (0+1)^*\}$

1. $\{a^m b^n\} = a^*b^* \text{ (IV)} \rightarrow \text{Regular}$

2. $\{a^m b^n \mid m=n \text{ or } m \neq n\}$
 $\rightarrow a^*b^* \text{ (Regular)}$

3. $\{a^m b^n \mid m=n\}$
 $\rightarrow a^n b^n \text{ (V)} \rightarrow \text{Not-Regular}$

4. $\{a^m b^n \mid m=2n\}$
 $\rightarrow a^{2n} b^n \text{ (V)} \rightarrow \text{Not-Regular}$

5. $\{a^m a b^n\}$
 $\rightarrow a^* a b^* \rightarrow \text{Regular } (a^+ b^*) \text{ (IV)}$

6. $\{a^m b^n \mid m \geq 2, n \geq 1\}$
 $\rightarrow a a a^* b b^* = a a^+ b^+ \text{ (Regular)}$

7. $\{a^m b^n \mid m < n, n < 100\}$
 $\rightarrow \text{finite. (Regular) (I)}$

8. $\{a^m b^n \mid m > n, m < 100\} \rightarrow \text{finite} \rightarrow \text{Regular (I)}$

9. $\{a^m b^n \mid m=n, m \neq n\} = \emptyset \text{ (I)} \rightarrow \text{Regular (I)}$

10. $\{a^m b^n \mid \text{LCM}(m,n)=1\} \rightarrow \{ab\} \rightarrow \text{Regular (I)}$

11. $\{a^m b^n \mid \text{gcd}(m,n)=1\} = \text{(V)} \rightarrow \text{Not-Regular.}$

12. $\{a^m b^n \mid \text{LCM}(m,n) < 100\} = \text{IV- Regular}$

13. $\{a^m b^n \mid \text{LCM}(m,n)=0\} = \text{IV. Regular}$

14. $\{a^m b^n \mid 0 < \text{LCM}(m,n) < 100\} = \text{I. Regular}$

15. $\{w_1, w_2 \mid w_1, w_2 \in (a+b)^*\} \rightarrow \text{Regular}$

16. $\{w_1 w_2 \mid |w_1| = |w_2|, w_1, w_2 \in (a+b)^*\} = \{w \mid |w| = \text{even}, w \in (a+b)^*\}$

$\therefore$ IV. Regular

17. $\{w_1 w_2 \mid w_1 = w_2, w_1, w_2 \in (a+b)^*\}$ $\nexists$. Not-Regular

18. $\{w_1 w_2 w_3 \mid w_1, w_2, w_3 \in (a+b)^*\}$ IV. Regular

19. $\{w_1 w_2 w_3 \mid |w_1| = |w_2| = |w_3|, w_1, w_2, w_3 \in (a+b)^*\}$ IV. Regular
 $|w| = 0 \text{ modulo } 3$.

20. $\{ww \mid w \in (a+b)^*\}$ $\nexists$. Not-Regular.

21. $\{www \mid w \in (a+b)^*\}$ $\nexists$. Not Regular

22. $\{ww^R \mid w \in (a+b)^*, w^R \text{ is reverse of } w\}$ $\nexists$. Not-Regular.

23. $\{wcw \mid w \in (a+b)^*, c \text{ is a special symbol}\}$ Not-Regular ($\nexists$)

24. $\{wcw^R \mid w \in (a+b)^*\}$ ($\nexists$) Not-Regular
 put min 'w' in given form. If it covers every other form for every 'w' then simply answer

25. $\{ww \mid w \in a^*\}$ $\{ \epsilon, a^2, a^4, a^6, \dots \} = (aa)^*$ depends on min 'w'
 $\therefore$ Regular (II) $\downarrow$ Regular

26. $\{wwx \mid w, x \in (a+b)^*\} = x \text{ (when } w = \epsilon) = (a+b)^*$
 IV. Regular

27. $\{wwx \mid w, x \in (a+b)^+\}$ $\nexists$. Not regular

28. $\{wwx \mid w \in (a+b)^*, x \in (a+b)^+\}$ $x = (a+b)^*$ Regular

29. $\{wwx \mid w \in (a+b)^+, x \in (a+b)^*\}$ $\nexists$. Not-regular

30. $\{wwx \mid w \in (a+b)^*, x \in (a+b)^*\}$ $\nexists$. Not regular

31. $\{wwx \mid w \in (a+b)^*, x \in (a+b)^*\}$ $aa^x + bb^x \rightarrow$ Regular

32. $\{wwx \mid w, x \in (a+b)^*\}$ I. Regular

33. $\{wwx \mid w, x \in a^*\}$ Regular

34. $\{wwx \mid w, x \in a^+\} = aaa^+.$ Regular (II).

35. $\{xww \mid w, x \in (a+b)^*\}$ Regular

36. $\{xww \mid w, x \in (a+b)^+\}$ Not-Regular.

37. $\{xww \mid w \in (a+b)^+, x \in (a+b)^+\}$ Regular

38. $\{xww \mid w \in (a+b)^+, x \in (a+b)^*\}$ Not Regular

39. $\{xww \mid w \in (a+b)^*, x \in (a+b)^*\}$ Not Regular

40. $\{xww \mid w \in (a+b), x \in (a+b)^*\}$ $xaa + xbb \therefore$ Regular

41. $\{xww \mid w, x \in a^*\}$ Regular (a^*)

42. $\{xww \mid w, x \in a^+\}$ Regular. (aaa^+)

43. $\{wxw \mid w, x \in (a+b)^*\}$

44. $\{wxw \mid w, x \in (a+b)^+\}$

45. $\{wxw \mid w \in (a+b)^*, x \in (a+b)^+\}$

46. $\{wxw \mid w \in (a+b)^+, x \in (a+b)^*\}$

47. $\{wxw \mid w \in (a+b)^*, x \in (a+b)^*\}$

$$48. \{ w \alpha w \mid w \in (a+b)^*, \alpha \in (a+b)^* \}^R$$

$$49. \{ w \alpha w \mid w \in (a+b)^*, \alpha \in (a+b)^* \}^R$$

$$50. \{ w \alpha w \mid w, \alpha \in a^* \}^R$$

$$51. \{ w \alpha w \mid w, \alpha \in a^+ \}^R$$

$$52. \{ w w^R \alpha \mid w, \alpha \in (a+b)^* \} \quad \alpha = (a+b)^*. \quad \therefore \text{Regular}$$

$$53. \{ w w^R \alpha \mid w, \alpha \in (a+b)^+ \} \quad \therefore \text{Not regular}$$

$$54. \{ w w^R \alpha \mid w \in (a+b)^*, \alpha \in (a+b)^+ \} \quad \alpha = (a+b)^+. \quad \therefore \text{Regular}$$

$$55. \{ w w^R \alpha \mid w \in (a+b)^+, \alpha \in (a+b)^* \} \quad \text{Not-regular}$$

$$56. \{ w w^R \alpha \mid w \in (a+b)^*, \alpha \in (a+b) \} \quad \text{Not-regular}$$

$$57. \{ w w^R \alpha \mid w \in (a+b), \alpha \in (a+b)^* \} \quad \text{Regular}$$

$$58. \{ w w^R \alpha \mid w, \alpha \in a^* \} \quad \text{Regular}$$

$$59. \{ w w^R \alpha \mid w, \alpha \in a^+ \} \quad \text{Regular}$$

$$60. \{ w \alpha w^R \mid w, \alpha \in (a+b)^* \}^R$$

$$61. \{ w \alpha w^R \mid w, \alpha \in (a+b)^+ \}^R$$

$$62. \{ w \alpha w^R \mid w \in (a+b)^*, \alpha \in (a+b)^+ \} \quad (a+b)^+ \text{ R}$$

$$63. \{ w \alpha w^R \mid w \in (a+b)^+, \alpha \in (a+b)^* \} \quad a \alpha a + b \alpha b \text{ (at least two length)} \\ \therefore \text{Regular}$$

64. $\{wxw^R \mid w \in (a+b)^*, x \in (a+b)^*\}$ Not-regular

65. $\{wxw^R \mid w \in (a+b), x \in (a+b)^*\}$ $axa + bxb \therefore$ Regular

66. $\{wxw^R \mid w, x \in (a+b)^*\}$ Regular

67. $\{wxw^R \mid w, x \in a^*\}$ Regular

68. $\{wxw^R \mid w, x \in a^+\}$ Regular

69. $wxw^R \therefore \min w: w = a, b \Rightarrow axa + bxb$

$\therefore w = aa \therefore aaxaa$

$w = ab \quad abxba$

$w = ba \quad baxab$

$w = bb \quad bbb$

$\therefore axa + bxb = a(a+b)^+a + b(a+b)^+b$

$\therefore$ Regular

70. $\{xww^R \mid w, x \in (a+b)^*\}$ R

71. $\{xww^R \mid w, x \in (a+b)^+\}$ $axa + xbb$

72. $\{xww^R \mid w \in (a+b)^*, x \in (a+b)^+\}$ $(a+b)^+$

73. $\{xww^R \mid w \in (a+b)^+, x \in (a+b)^*\}$ $axa + xbb$ not.

74. $\{xww^R \mid w \in (a+b)^*, x \in (a+b)^*\}$ R

75. $\{xww^R \mid w \in (a+b), x \in (a+b)^*\}$

76. $\{xww^R \mid w, x \in a^*\}$ R

77. $\{xww^R \mid w, x \in a^+\}$ R

78. $\{xww^Ry \mid w, x, y \in (a+b)^+\}$ $axay + xbbx \Rightarrow$ Regular

78. $\{xwzw^Ry \mid w, x, y, z \in (a+b)^+\}$ = $xazay + xbzby$
Regular

79. $\{ww \mid w, x, y \in (a+b)^+\}$
Regular

$wxyw$ $xwyw$
 $axay + bxbx$ $axay + bxbx$

80. $\{wxwy \mid w, x, y \in (a+b)^*\}$ $axay + bxbx \therefore$ Regular

81. $\{xwyw \mid w, x, y \in (a+b)^+\}$ $xaya + xbyb \therefore$ Regular

82. $\{xwywz \mid w, x, y, z \in (a+b)^+\}$ $\therefore xaya + xbyb$
 $\therefore$ Regular

83. $\{wxw^Ry \mid w, x, y \in (a+b)^*\}$ $(a+b)^*$
 $\therefore$ Regular

84. $a^n! \{a^1, a^2, a^6, a^{24}, \dots\}$
 $\therefore$ Not-Regular

95. $(a^{m^n})^* \rightarrow a^* \rightarrow$ Regular

85. $a^{\text{prime}} \{a^2, a^3, a^5, \dots\}$ NR.

96. $(a^{n^n})^* \rightarrow a^* \rightarrow$ Regular

86. $a^{2^n} \{a^0, a^2, a^4, \dots\}$ AP $\rightarrow$ Regular

87. $a^{2^n} \{a^1, a^2, a^4, a^8, a^{16}, \dots\}$ NR

97. $(a^{n^2})^* \rightarrow a^* \rightarrow$ Regular

88. $\{a^{m^n} \mid m, n \geq 1\} \rightarrow \{a^1, a^2, a^3, \dots\}$ Regular

98. $(a^{2^n})^* (a^{\text{prime}})^* \rightarrow a^*$ Reg.

89. $\{a^{n^n} \mid n \geq 1\} \{a^1, a^4, a^{27}, \dots\}$ NR

99. $a^n!, (a^{\text{prime}})^* \rightarrow a^+ \rightarrow$ Reg

90. $\{a^{n^4}\} \{a^0, a^4, a^{16}, a^{81}, \dots\}$ NR

100. $g^m(a^{2^n})^* t^{2^m}(e^{\text{prime}})^* R$

91. $(a^{n!})^* a^* \rightarrow$ Regular

101. $a^{4n+5} \{a^5, a^9, a^{13}, \dots\} R$

92. $(a^{\text{prime}})^* = \Sigma^* - \{a\}$
"Complement of a " $\{a^n \mid n \neq 1\} \rightarrow$ Regular

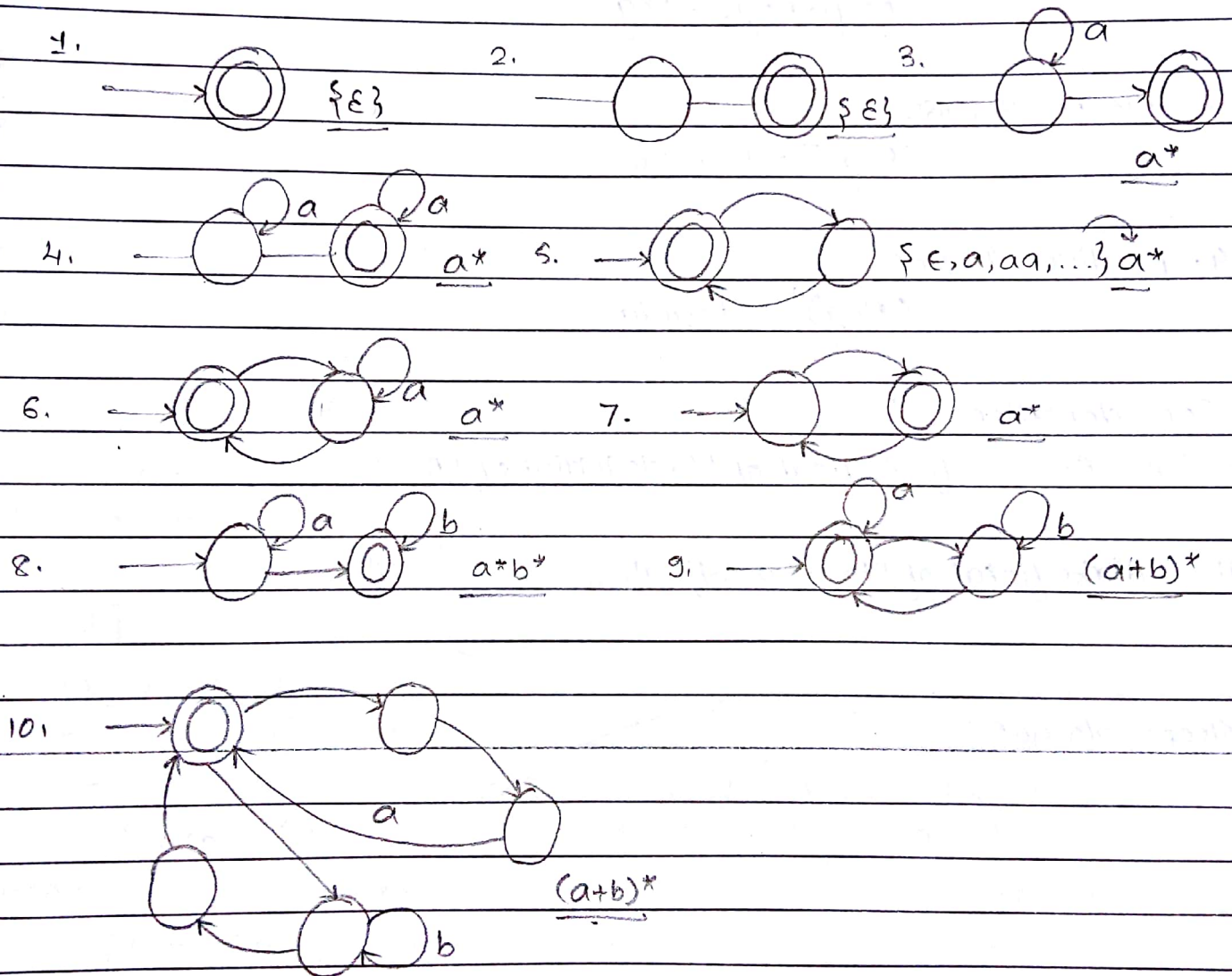
Note: If $L^* =$ Regular then L may be or may not be Regular.

93. $(a^{2^n})^* \rightarrow (aa)^*$ Regular If L is Regular then L^* is also

94. $(a^{2^n})^* \rightarrow a^*$ Regular

$$\begin{aligned}
 (a^p \text{ prime})^* &= (a^2)^* \cup (a^3)^* \cup (a^4)^* \cup (a^5)^* \cup (a^6)^* \cup \dots \\
 &= \{\epsilon, a^2, a^3, a^4, a^5, a^6, \dots\} \\
 &= \Sigma^* - \{a\} \\
 &= \text{complement of } L = \{a\} = \overline{\{a\}} \\
 &= \{a^n \mid n \neq 1\} \\
 &= \{a^n \mid n=0 \text{ or } n \geq 2\} \\
 &= \epsilon + aaaa^* \\
 &= \epsilon + aa^+
 \end{aligned}$$

Understanding -
Non-Deterministic Finite Automata with ϵ -moves:



Application of E-NFA:

01. Union

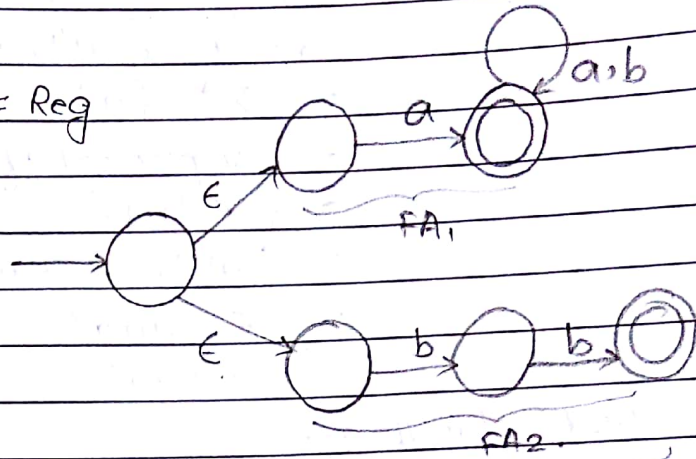
$$Reg_1 \cup Reg_2 = Reg$$

$Reg_1 \rightarrow FA$

$$L_1 = ax$$

$Reg_2 \rightarrow FA$

$$L_2 = xbb$$



02. Concatenation:

$$Reg_1 \cdot Reg_2 = Reg$$

$FA_1 \cup FA_2$

03. Kleene closure

$$(Reg)^* = Regular$$

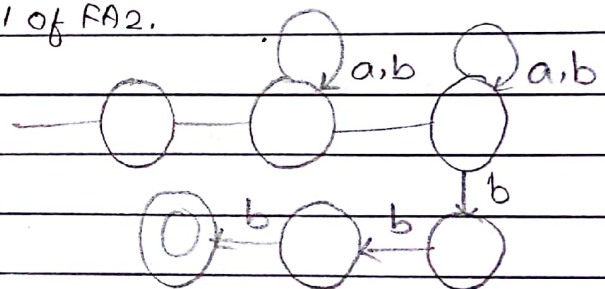
04. positive closure

$$(Reg)^+ = Regular$$

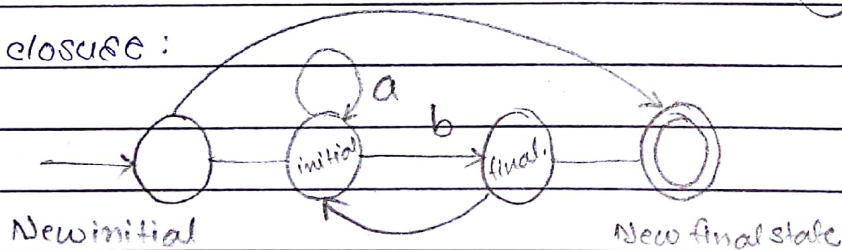
Concatenation:

i. Give ϵ -move from final of FA_1 to initial of FA_2 .

ii. Make final of FA_1 as non-final.



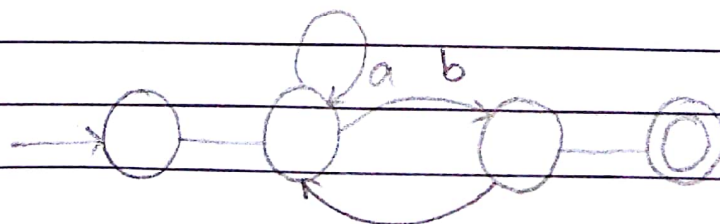
Kleene closure:



$$L = a^*b$$

$$L^* = (a^*b)^*$$

Positive closure:



Conversions

01. $NFA \rightarrow DFA$ (Subset construction)
02. $E-NFA \rightarrow NFA$
03. $E-NFA \rightarrow DFA$ (Subset construction)
04. $DFA \rightarrow NFA$
05. $NFA \rightarrow E-NFA$
06. $E-NFA \xrightarrow{\text{by definition}} DFA$

} "by definition"

$$DFA \cong NFA \cong E-NFA$$

Conversion from NFA to DFA (Subset Construction)

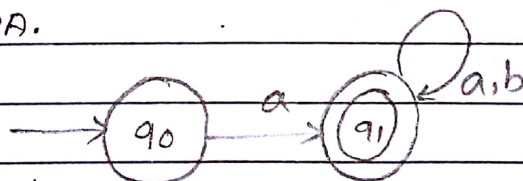
NFA
($Q, \Sigma, \delta_N, q_0, F$)

DFA
($2Q, \Sigma, \delta_D, \{q_0\}, F$)

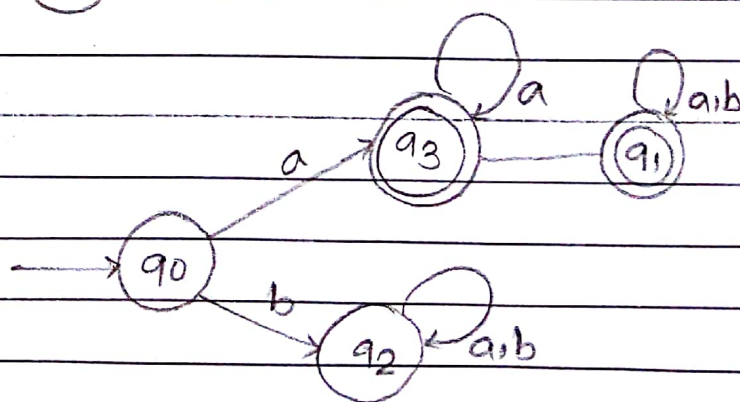
i. Initial (DFA) is Initial (NFA).

ii. $\delta_D(\{P, Q\}, i) = \delta_N(P, i) \cup \delta_N(Q, i)$

iii. If any subset (state) contain final of NFA, make that subset as final in DFA.



δ_N a b
 $\rightarrow A q_0 \{A, B\} \emptyset$
 $* B q_1 \{B\} \{B\}$



DFA: δ_D a b
 $\{A\} q_0 \{A, B\} \{ \}$
 $\{A, B\} q_3 \{A, B\} \{B\}$
 $\{B\} q_1 \{B\} \{B\}$
 $\{ \} q_2 \{ \} \{ \}$

Note:

- If NFA has n states, No. of states in min. DFA = atmost 2^n
[atleast 1]

- If DFA has n states, No. of states in min. NFA =
 $\log_2 n \leq \text{No. of states in min NFA} \leq n$

- If min NFA has n states, No. of states in min. DFA =
 $1 \leq \text{No. of states in minimum DFA} \leq 2^n$
(minimum (atleast, lower bound $\rightarrow n$)
maximum (atmost, upper bound $\rightarrow 2^n$)

- If NFA has n states, No. of states in min. DFA:

$$1 \leq \text{No. of states in minimum DFA} \leq 2^n$$

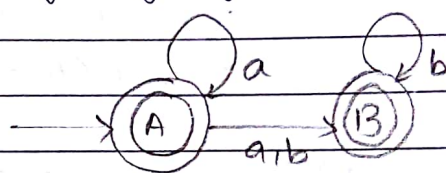
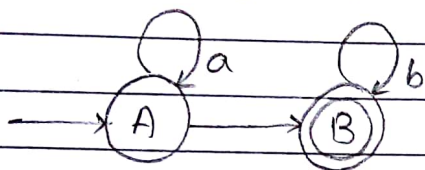
- If DFA has n states, No. of states in min. NFA:

$$\log_2 n \leq \text{No. of states in minimum NFA} \leq n$$

Conversion from E-NFA to NFA

- i. No. of states in NFA = No. of states in E-NFA.
- ii. Initial state (NFA) is same as initial (E-NFA).
- iii. $\delta_N(P, i) = \epsilon\text{-closure}(\delta_\epsilon(\epsilon\text{-closure}(P), i))$
- iv. If any $\epsilon\text{-clo}(q)$ contain final of E-NFA then make 'q' as final in NFA.

$\epsilon\text{-closure}(q)$: It is set of states where each state is reachable from q without reading any symbol.



$$\delta_\epsilon(A, a) = A$$

$$\delta_\epsilon(A, b) = \emptyset$$

$$\delta_\epsilon(B, a) = \emptyset$$

$$\delta_\epsilon(B, b) = B$$

$$\epsilon\text{-closure}(A) = \{A, B\}$$

$$\epsilon\text{-closure}(B) = \{B\}$$

$$\begin{aligned}\delta_N(A, a) &= \epsilon\text{-clo}(\delta_q(\epsilon\text{-clo}(A, a))) \\ &= \epsilon\text{-clo}(\delta_\epsilon\{A, B\}, a) \\ &= \epsilon\text{-clo}(\delta_\epsilon(A, a) \cup \delta_\epsilon(B, a)) \\ &= \epsilon\text{-clo}(\{A\} \cup \emptyset) \\ &= \epsilon\text{-clo}(A) \\ &= \{A, B\}\end{aligned}$$

$$\delta_N(A, a) = \{A, B\}$$

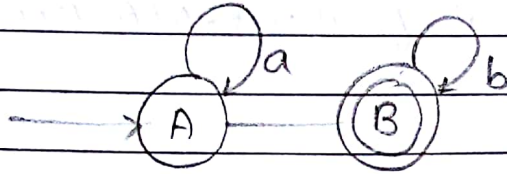
$$\delta_N(A, b) = \{B\}$$

$$\delta_N(B, a) = \emptyset$$

$$\delta_N(B, b) = \{B\}$$

Conversion from E-NFA to DFA

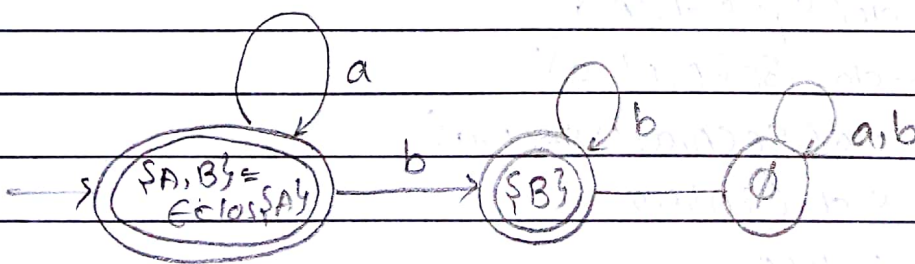
- Initial state (DFA) = ϵ -closure (Initial of E-NFA)
- $\delta_D(P, i) = \epsilon\text{-clo}(\delta_E(P, i))$
- If any state contain final of E-NFA, then make it as final in DFA.



$$\begin{aligned}\delta_D(\{A, B\}, a) &= \epsilon\text{-clo}(\delta_E(\{A, B\}, a)) \\ &= \epsilon\text{-clo}(\{A\} \cup \emptyset) \\ &= \epsilon\text{-clo}(A) = \{A, B\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{A, B\}, b) &= \epsilon\text{-clo}(\delta_E(\{A, B\}, b)) \\ &= \epsilon\text{-clo}(\emptyset \cup \{B\}) \\ &= \epsilon\text{-clo}(B) = \{B\}\end{aligned}$$

$$\begin{aligned}\delta_D(\{B\}, a) &= \epsilon\text{-clo}(\delta_E(\{B\}, a)) & \delta_D(\{B\}, b) &= \{B\} \\ &= \epsilon\text{-clo}(\emptyset) = \emptyset\end{aligned}$$



Regular Grammars

$$G_1 = (V, T, P, S)$$

$V \rightarrow$ set of variables (non-terminals)

$T \rightarrow$ set of terminals

$P \rightarrow$ set of rules / productions

$S \rightarrow$ start symbol. $S \in V$

Grammar: It is set of rules that generate a language (set of strings)

Eg. $S \rightarrow ABb$

$$A \rightarrow aA/b$$

$$B \rightarrow bBa/ab$$

$$V = \{S, A, B\}, T = \{a, b\}$$

$$P = \{S \rightarrow ABb, A \rightarrow aA, A \rightarrow b, B \rightarrow bBa, B \rightarrow ab\}$$

$$S = S$$

- Regular Grammar $\rightarrow$ It is LLG or RLG.

Left Linear Grammar

$$V \rightarrow VT^* / T^* \rightarrow \text{at most one non-terminal (at left most)}$$

$\hookrightarrow$ Exactly one non-terminal.

Eg. $S \rightarrow Aa/bcd$

$$A \rightarrow c$$

Right Linear Grammar

$$V \rightarrow T^*V / T^*$$

$\hookrightarrow$ at most one non-terminal but at right most position.

Eg. $S \rightarrow abS/efg.$

1. $S \rightarrow abc$ LLG,
RLG

Regular Grammar.

2. $S \rightarrow A$ LLG,
 $A \rightarrow ab$ RLG

$\therefore$ Regular Grammar

3. $S \rightarrow Sa | Sbc | \epsilon$
only LLG.

4. $S \rightarrow as | bs | ab$
only RLG.

5. $S \rightarrow as | sb | a$
Not RLG, not LLG.

Identify language generated by the following Regular Grammars:

1. $S \rightarrow a$ $L = \{a\}$

2. $S \rightarrow a | b$
 $L = a^+b = \{a, b\}$
 $= \{w | w \in (a+b)^+\}$

3. $S \rightarrow Aa | b$
 $A \rightarrow a$
 $L = \{aa, b\}$

4. $S \rightarrow AB$ (Not-Reg. Gr)
 $A \rightarrow a, B \rightarrow b$
 $L = \{ab\}$

5. $S \rightarrow a | \epsilon$
 $L = \{\epsilon, a\}$

6. $S \rightarrow Sa | b$
Left recursion

- Regular language always contain regular grammar but non-regular grammar can also generate regular language.

- Every regular-grammar generates regular language.

Not Grammar.

$$6. S \rightarrow sab$$

$$L = \{b, ba, baa, \dots\}$$

left recursion = ba^*

$$7. S \rightarrow abs/c$$

$$L = \{ab\}^*c$$

(Right recursion)

$$8. S \rightarrow asb/c$$

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

$$9. S \rightarrow as/\epsilon$$

$$a^*$$

$$10. S \rightarrow sa/\epsilon$$

$$a^*$$

$$11. S \rightarrow as/a$$

$$a^+$$

$$12. S \rightarrow as/b$$

$$a^*b$$

$$13. S \rightarrow sala$$

$$a^+$$

$$14. S \rightarrow salb$$

$$ba^*$$

$$15. S \rightarrow aas/\epsilon$$

$$(aa)^*$$

$$16. S \rightarrow saa/\epsilon$$

$$(aa)^*$$

Equivalence

- Regular Expression $\cong$ Finite Automata
- Finite Automata $\cong$ Regular grammar.

} R.E $\cong$ Reg Grammar

Conversion:

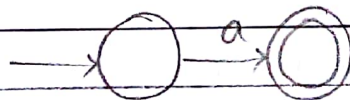
$$1. RE \rightarrow FA$$

$$2. FA \rightarrow RE$$

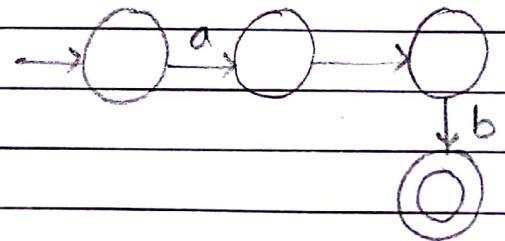
Conversion from Regular Expression to finite Automata.

(Induction Method)

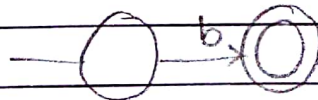
$$1. a$$



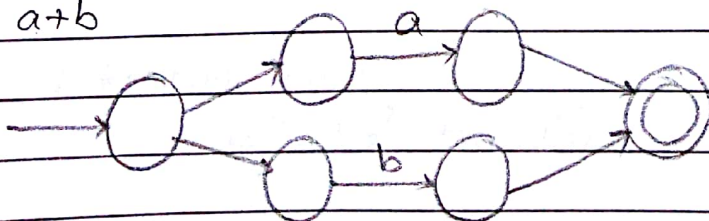
$$4. a.b.$$



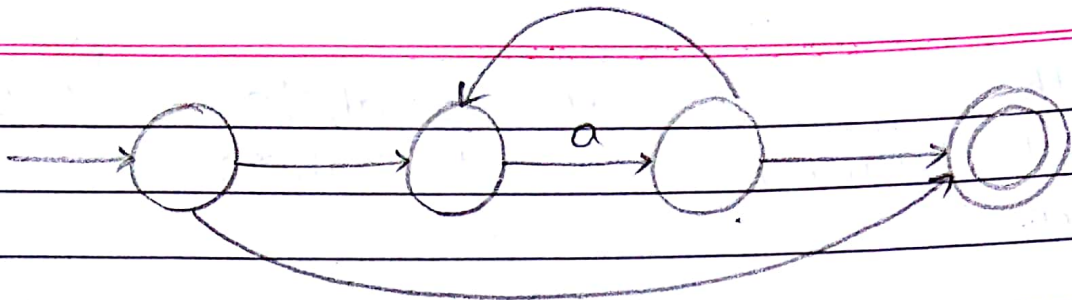
$$2. b$$



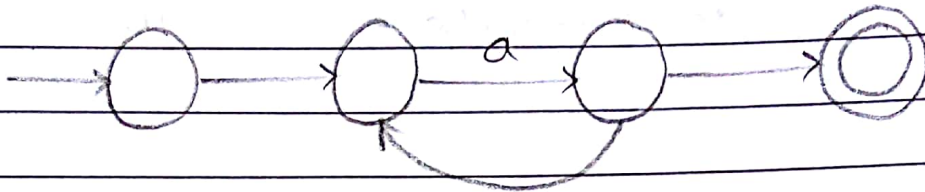
$$3. a+b$$



5. a^*

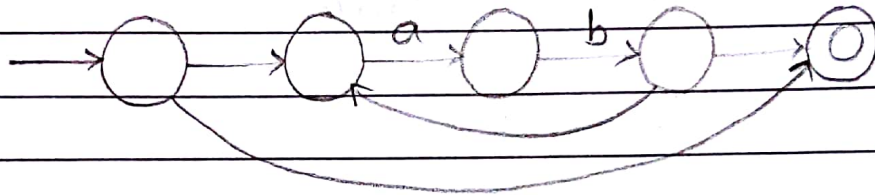
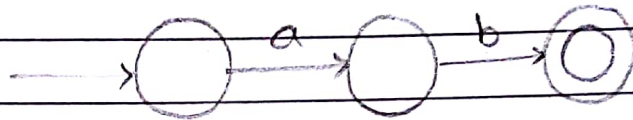


6. a^+



Q.1. $(ab)^*$

ab:



2. $((a^*b)^+ \cdot ab + (ab)^*a)^*b$

Conversion of Finite Automata to R.E.

1. Kleene method (It can be applied to any FA)
2. Arden's method (only applied to DFA/NFA but not ϵ -NFA)

Kleene Method

- Dynamic programming
- similar to all pairs shortest paths.
- $O(n^3)$

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$$

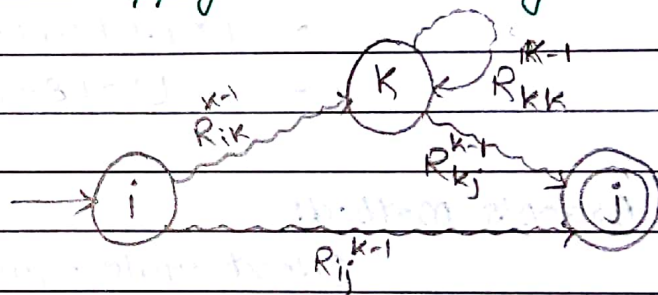
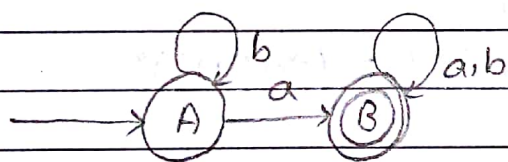
Arden's Method:

- state equation based
- Every state, we compute equation.
- Final state will decide Regular Expression.

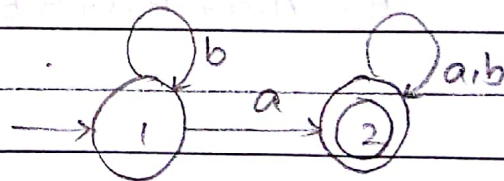
If $R = Q + RP$ then $R = QP^*$

Note: If P contain ϵ then it may have infinite solutions

(Apply to DFA/NFA only)



1. Label the states with numbering.



2. Find equation R_{12}^2 (where 1 is initial and 2 is final)

3. Take base cases:

$$R_{11}^0, R_{12}^0, R_{21}^0, R_{22}^0$$

$$R_{11}^0 = b + \epsilon$$

$$R_{21}^0 = \phi$$

$$R_{12}^0 = a$$

$$R_{22}^0 = a + b + \epsilon$$

$$R_{ij}^k = R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$$

$$i=1, j=2, k=2$$

$$= R_{12}^1 + R_{12}^1 (R_{22}^1) R_{22}^1 \quad \text{--- (1)}$$

$$R_{12}^1 = ? \quad R_{22}^1 = ?$$

$$\begin{aligned} R_{12}^1 &= R_{12}^0 + R_{11}^0 (R_{11}^0)^* R_{12}^0 \\ &= a + (b + \epsilon) (b + \epsilon)^* a \\ &= a + b^* a \\ &= b^* a. \quad \text{--- (2)} \end{aligned}$$

$$\begin{aligned} R_{22}^1 &= R_{22}^0 + R_{21}^0 (R_{11}^0)^* R_{12}^0 \\ &= (a + b + \epsilon) + \phi \\ &= a + b + \epsilon \quad \text{--- (3)} \end{aligned}$$

Substitute (2) & (3) in 1

$$\begin{aligned} R_{12}^2 &= R_{12}^1 + R_{12}^1 (R_{22}^1)^* R_{22}^1 \\ &= b^* a + b^* a (a + b + \epsilon)^* (a + b + \epsilon) \\ &= b^* a + b^* a (a + b)^* \\ &= b^* a [\epsilon + (a + b)^*] = b^* a (a + b)^* \end{aligned}$$

Arden's Method:

(Next state equation)

$$A = \epsilon + A \cdot b \quad \longrightarrow \quad A = \epsilon \cdot b^* = b^*$$

$$B = A \cdot a + B \cdot a + B \cdot b \quad \longrightarrow \quad B = b^* a + B(a + b)$$

$$\begin{aligned} R &= \emptyset + R P_{p.s.} \\ B &= (b^* a) (a + b)^* \end{aligned}$$

Equivalence between Regular Grammars & Finite Automata.

$$1. \quad FA \Rightarrow RLG$$

$$2. \quad RLG \Rightarrow FA$$

$$3. \quad FA \Rightarrow LLG$$

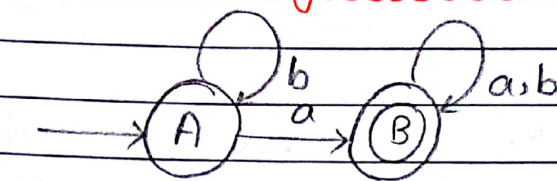
$$4. \quad LLG \Rightarrow FA$$

$$5. \quad RLG \Rightarrow LLG$$

$$6. \quad LLG \Rightarrow RLG$$

$$FA \cong LLG \cong RLG$$

Finite Automata to Right Linear Grammars



P.S I/P NS
 $A \rightarrow bA$ $A \rightarrow aB$

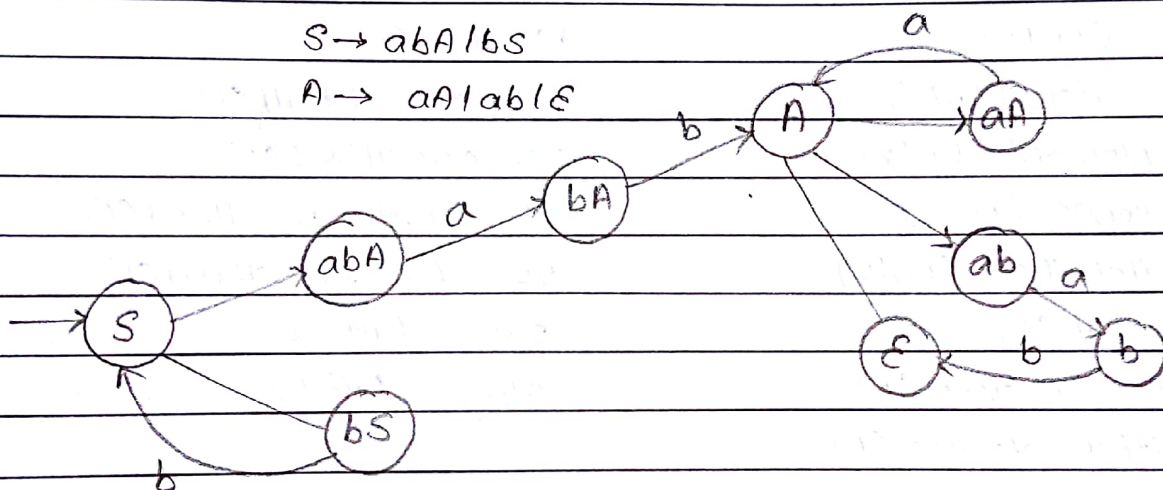
$B \rightarrow bB$ $B \rightarrow aB$ $B \rightarrow \epsilon$ (final state)

($\because$ B is final state add null product)

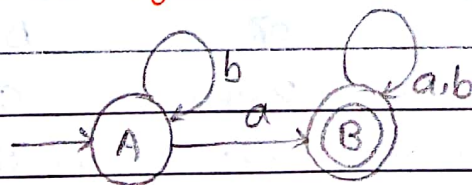
$\therefore \left. \begin{array}{l} A = bA / aB \\ B = aB / bB / \epsilon \end{array} \right\} \begin{array}{l} \text{with null} \\ \text{production} \end{array} \quad \left. \begin{array}{l} A = bA / aB / a \\ B = aB / bB / a / b \end{array} \right\} \begin{array}{l} \text{without null} \\ \text{production} \end{array}$

Right Linear Grammars to Finite Automata

$S \rightarrow abA / bS$
 $A \rightarrow aA / ab / \epsilon$



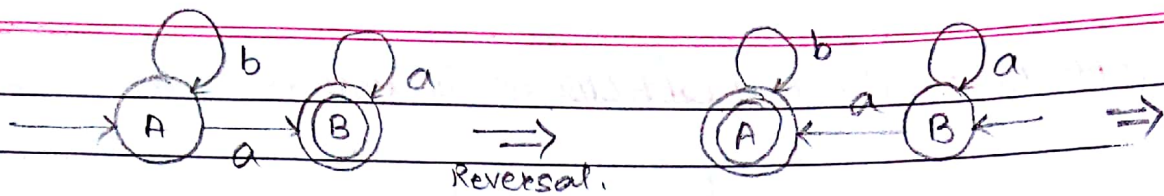
Finite Automata to Left Linear Grammars



$FA \xrightarrow{\text{Reversal}} (FA)^{\text{Rev}} \xrightarrow{\text{RLG}} (RLG)^{\text{Rev}} = LLG$

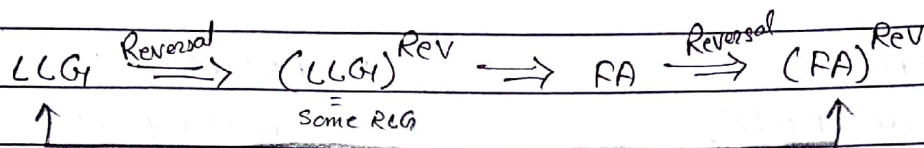
Reversal :- Final = Initial

Initial = Final. Reverse every edge.



$$\begin{aligned}
 B &\rightarrow aB \mid aA & \Rightarrow & B \rightarrow Ba \mid Aa \\
 A &\rightarrow bA \mid \epsilon & \text{reversal} & A \rightarrow Ab \mid \epsilon. \\
 & & \text{(Reverse RHS)} &
 \end{aligned}$$

Left Linear Grammars to Finite Automata:



Close Properties of Regular languages:

- | | |
|---|----------------------------|
| 1. Union (L_1, L_2) | 18. Half (L) |
| 2. Intersection (L_1, L_2) | 19. second Half (L) |
| 3. Complement (L_1, L_2) | 20. one-third (L) |
| 4. Difference (L_1, L_2) | 21. middle one-third (L) |
| 5. Concatenation (L_1, L_2) | 22. last one-third (L) |
| 6. Reversal (L) | 23. $L_1 \oplus L_2$ |
| 7. Kleene Closure (L) | 24. $L_1 \odot L_2$ |
| 8. positive closure (L) | |
| 9. prefix (L) | 25. finite Union |
| 10. suffix (L) | 26. finite Intersection |
| 11. substring | 27. Difference |
| 12. subsequence (L) | 28. Concatenation |
| 13. Quotient (L_1, L_2) | 29. Subset |
| 14. <u>subset (L)</u> not closed | 30. Substitution |
| 15. Substitution (L) | |
| 16. Homomorphism (L) | |
| 17. Inverse Homomorphism (L) | |

31. Infinite Union
32. Infinite Intersection
33. Infinite Difference
34. Infinite Concatenation
35. Infinite Subset
36. Infinite Substitution.

} not closed.

Union

Regular $\cup$ Regular = Regular

1. Use ϵ -NFA
2. Use Regular Expression
3. Use compound finite automata.

} Proofs.

Eg. $L_1 = a^*b^*$, $L_2 = (a+b)^*$
 $\therefore L_1 \cup L_2 = L_2$

$L_1 = a^*b^*$, $L_2 = a^*$
 $L_1 \cup L_2 = L_1$

$L_1 = a^*$, $L_2 = b^*$
 $L_1 \cup L_2 = a^* + b^*$

$$\left. \begin{array}{l} \text{Reg. 1} \Rightarrow \text{FA}_1 \\ \text{Reg. 2} \Rightarrow \text{FA}_2 \end{array} \right\} \text{FA}_1 \times \text{FA}_2 \Rightarrow \text{Reg. 1} \cup \text{Reg. 2}$$

$L_1 = \emptyset$, $L_2 = \text{any}$
 $L_1 \cup L_2 = L_2$

$L_1 = \Sigma^*$, $L_2 = \text{Any}$
 $L_1 \cup L_2 = \Sigma^*$, L_1

$L_1 = (aa)^*$
 $L_2 = a(aa)^*$
 $L_1 \cup L_2 = a^*$

$L_1 = \text{Regular}$, $L_2 = \text{Non-Regular}$
 $L_1 \cup L_2 = \text{may be or may not be regular}$

Reg: $\emptyset, \Sigma^*$
 Non-reg: $a^n b^n$

$L_1 = \text{Non-Regular}$, $L_2 = \text{Non-Regular}$
 $L_1 \cup L_2 = \text{may be or may not be regular}$

Intersection

Regular $\cap$ Regular = Regular

- Use compound FA

- Regular languages closed under Intersection.
- Intersection is closed for regular languages.
- (Set of all regulares, $\cap$) is closed.
- ($\{L_1 \mid L_1 \text{ is regular}\}, \cap$) is closed.

Eg. $L_1 = a^*, L_2 = b^* \quad L_1 \cap L_2 = \epsilon$

$L_1 = a^+, L_2 = b^+ \quad L_1 \cap L_2 = \emptyset$

$L_1 = a^*b^*, L_2 = a^+b^+ \quad L_1 \cap L_2 = L_2$

$L_1 = \emptyset, L_2 = \text{any} \quad L_1 \cap L_2 = L_1$

$L_1 = \Sigma^*, L_2 = \text{Any} \quad L_1 \cap L_2 = L_2$

$L_1 = (ab)^*, L_2 = a^*b^* \quad L_1 \cap L_2 = \{\epsilon, ab\}$

$L_1 = \text{Regular}, L_2 = \text{Not-Regular}$

$L_1 \cap L_2 = \text{may or may not be regular}$

Complement

Regular = Regular

Proof: Regular $\Rightarrow$ DFA $\xrightarrow{*}$ modified DFA $\Rightarrow$ Regular
(Interchange finals & non-final)

Eg.

$L =$ Set of all strings where each string not containing 'aaa' as substring.

$\bar{L} =$ Set of all strings containing 'aaa'.

Note:

" L & $\bar{L}$ has same no. of states in DFA."

$L = \emptyset \Rightarrow \bar{L} = \Sigma^*$

$L = \Sigma^* \Rightarrow \bar{L} = \emptyset$

$L = a(a+b)^* \Rightarrow$

$\bar{L} = b(a+b)^* + \epsilon$

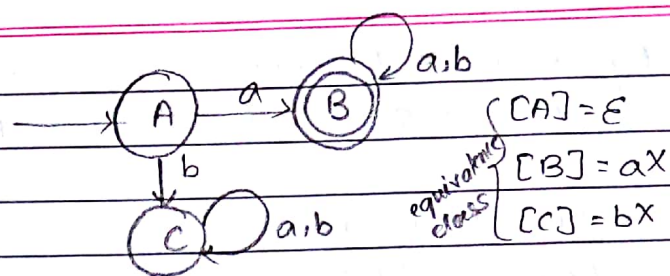
$L \cup \bar{L} = \Sigma^*$

$\bar{L} = \Sigma^* - L$

	ba
aa	ϵ

$$L = (a+b)^*b \Rightarrow$$

$$\bar{L} = \epsilon + (a+b)^*a$$



Note:

No. of equivalence class is equal to the no. of states in minimum DFA.

Difference

$$Reg_1 - Reg_2 = Regular$$

proof:

1. use compound finite automata (FA).

$$2. L_1 - L_2 = L_1 \cap \bar{L}_2$$

$$3. L_1 - L_2 = \overline{(L_1 \cap \bar{L}_2)} = \bar{L}_1 \cup L_2$$

$$Reg_1 - Reg_2 = Reg_1 \cap \bar{Reg}_2$$

$$= Reg_1 \cap Reg = Regular$$

Eg.

$$\bullet L_1 = a^*, L_2 = b^*$$

$$L_1 - L_2 = a^+$$

$$L_2 - L_1 = b^+$$

$$\bullet L_1 = \phi, L_2 = \Sigma^*$$

$$L_1 - L_2 = \phi$$

$$L_2 - L_1 = \Sigma^*$$

$$\bullet L_1 = \phi, L_2 = any$$

$$L_1 - L_2 = \phi$$

$$L_2 - L_1 = any$$

$$\bullet L_1 = any, L_2 = \Sigma^*$$

$$L_1 - L_2 = \phi$$

$$L_2 - L_1 = \bar{L}_1 (\Sigma^* - L_1 = \bar{L}_1)$$

$$\bullet L_1 \rightarrow Regular, L_2 \rightarrow Not-regular$$

$$L_1 - L_2 = may or may not be regular$$

$$L_2 - L_1 = \text{---} \text{---} \text{---}$$

$$\bullet \overline{Non-regular} = Non-regular$$

$$\bullet \overline{Regular} = Regular$$

Concatenation

$$Reg_1 \cdot Reg_2 = Regular$$

Proof: 1. Use ε-NFA

2. Use RE.

Eg. • $L_1 = \emptyset, L_2 = \text{any}$
 $L_1 \cdot L_2 = \emptyset$ $L_2 \cdot L_1 = \emptyset$

• $L_1 = a^*, L_2 = b^*$

$L_1 \cdot L_2 = a^*b^*$

• $L_2 \cdot L_1 = b^*a^*$

• $L_1 = (a+b)^*, L_2 = a^*$

$L_1 \cdot L_2 = (a+b)^* \cdot a^* = (a+b)^* L_1$

$L_2 \cdot L_1 = L_1$

• $L_1 = a^*b^*, L_2 = a^*$

$L_1 \cdot L_2 = a^*b^*a^* \quad L_2 \cdot L_1 = a^*a^*b^* = a^*b^*$

Reversal

$(\text{Regular})^{\text{Rev}} = \text{Regular}$

Proof:

• $\text{Reg} \Rightarrow \text{finite automata} \Rightarrow (\text{finite automata})^{\text{Rev}} \Rightarrow (\text{Reg})^{\text{Rev}}$

• Reverse each RHS, in regular grammar

$S \rightarrow Sa|b \Rightarrow ba^*$

Reverse: $S \rightarrow as|b \Rightarrow a^*b$

Eg. • $L = a^*b^* \quad L^R = b^*a^*$

• $L = (a+b)^* \Rightarrow L^R = (a+b)^* \Rightarrow L$

• $L = a^*, \Rightarrow L^R = L = a^*$

• $L = \emptyset, L^R = L$

• $L = a(a+b)^* \Rightarrow L^R = (a+b)^*a$

Kleene Closure

$(\text{Regular})^* = \text{Regular}$

Proof:

• use ϵ -NFA

• use Regular expression

Eg. • $L = (a+b)^*$

$L^* = L$

• $L = a^*$

$L^* = L$

• $L = \emptyset$

$L^* = \{\epsilon\}$

• $L = a^*b^*$

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

• $L = a(a+b)^*$

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

• $L = a^* + b^*$

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

$= (a+b)^*$

• $L = a + b$

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

• $L = \{\epsilon\}$

$L^* = L$

• $L^* = (a(a+b)^*)^*$

$= \epsilon + a(a+b)^*$

Positive Closure

$$(Regular)^+ = Regular$$

Eg. • $L = (a+b)^*$ • $L = a^*$ • $L = \emptyset, L^+ = \emptyset$
 $L^+ = [(a+b)^*]^+ = L$ • $L^+ = L$

• $L = a^*b^*$ • $L = a(a+b)^*$ • $L = a^*+b^*$
 $L^+ = (a+b)^*$ • $L^+ = (a(a+b)^*)^+ = L$ • $L^+ = (a^*+b^*)^+$
 $L^+ = (a+b)^*$

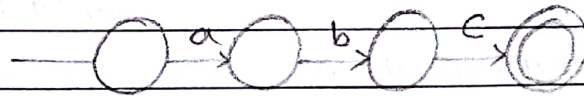
• $L = a+b$ • $L = \{\epsilon\}$
 $L^+ = (a+b)^{++}$ • $L^+ = L$

Prefix:

$$w = abc$$

$$prefix(w) = \epsilon, a, ab, abc \text{ (sequence of beginning symbols)}$$

$$L = \{abc\}$$



from initial to final path make all states as final.



$$w = abc = uv$$

$$prefix(L) = \{u \mid uv \in L, \text{ any } v\}$$

$$|w| = n \Rightarrow \text{no. of prefixes} = n+1$$

$$\text{no. of proper prefixes} = n$$

$$\begin{aligned} \text{prefix} &= \epsilon \mid abc \\ &= a \mid bc \\ &= ab \mid c \\ &= abc \mid \epsilon \end{aligned}$$

Suffix

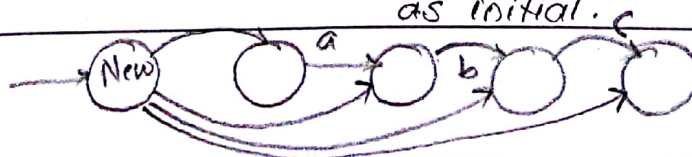
$$w = abc$$

$$suffix(w) = \epsilon, c, bc, abc \text{ (sequence of ending symbols)}$$

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



from initial to final path, make all states as initial.



$w = abc$

$= uv$

$= \epsilon abc$

$= a bc$

$= ab c$

$= abc \epsilon$

Suffix

$$\text{suffix}(L) = \{v \mid uv \in L, \text{ any } u\}$$

$$|w| = \text{no. of suffixes} = n+1$$

$$\text{no. of proper suffixes} = n$$

• Language over one symbol have exactly same prefixes and suffixes. (Eg. a^*)

Substring

$$\text{string}(w) = aaa$$

$$\text{substring} \Rightarrow \epsilon, a, aa, aaa$$

$$\text{string}(w) = abc$$

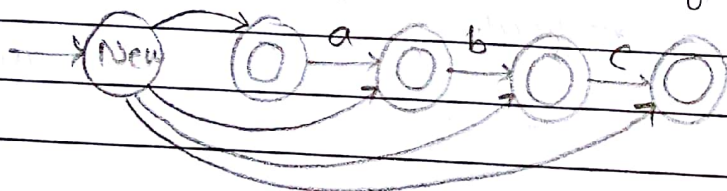
$$\text{substring} \Rightarrow \epsilon, a, b, c, ab, bc, abc$$

$$L = \{abc\}$$



from initial to final path

every state is initial & final.



$$\text{string}(w) = abc = xyz$$

substrings of 'abc' ϵ - a & bc

a ϵabc

b abc

c $abc \epsilon$

ab ϵabc

bc $abc \epsilon$

abc $\epsilon abc \epsilon$

$$\text{Substring}(L) : \{y \mid xyz \in L, \text{ any } x \text{ and } z\}$$

$$\text{If } |w| = n$$

$$n+1 \leq \text{no. of substrings} \leq \Sigma n+1$$

$$\Sigma n = \frac{n(n+1)}{2}$$

$$2$$

Q. No. of substrings of different lengths for given string of length 'n'. $= n+1$

$= n$ (exclude zero length) or

non-empty substring of different length.

$w = abc$

$\text{substring}(w) = \epsilon, (a, b, c), (ab, bc), abc$
4 different lengths

Subsequence

$w = aaa$

$\text{subsequence}(w) = \epsilon, a, aa, aaa$

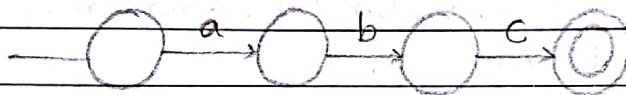
$w = abc$

$\text{subsequence}(w) = \epsilon, a, b, c, ab, bc, ac, abc$

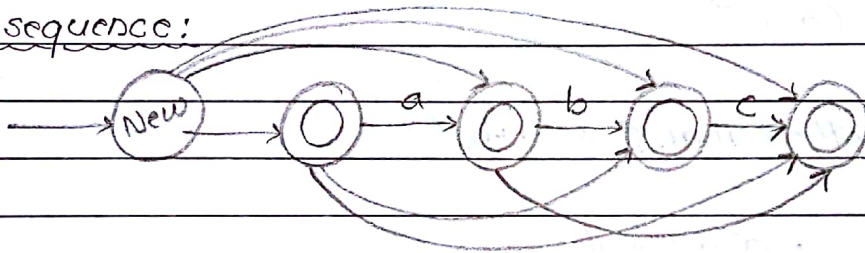
If $|w| = n$,

$$n+1 \leq \text{no. of subsequences of } w \leq 2^n$$

$L = \{abc\}$



Subsequence:



Quotient

$$L_1/L_2 = \{u \mid uv \in L, v \in L_2\}$$

$L_1 = \{a, ab, aac\}$

$L_2 = \{\epsilon, a, b, c\}$

$L_1/L_2 = \{a/a, ab/\epsilon, ab/b, aac/\epsilon, aac/c\}$ possible

$= \{a/b, a/c, ab/a, ab/c, aac/a, aac/b\} \rightarrow \text{Not possible}$

$= \{a, \epsilon, ab, aac, aa\}$

$Any/\epsilon = Any$

$ab/\epsilon = ab$

$a/\epsilon = a$

$aa/\epsilon = aa$

$$\begin{aligned} \bullet \quad a^*/a &= \{\epsilon/a, a/a, aa/a, aaa/a, \dots\} \\ &= \{\epsilon, a, a^2, a^3, \dots\} \\ &= a^* \end{aligned}$$

$$\bullet \quad a^*a/a = \frac{a^*a}{a} = a^*$$

$$\bullet \quad a^+/a = a^+$$

$$\bullet \quad a^*b/ab^* \text{ (HW)}$$

$$\begin{aligned} \bullet \quad (a+b)^*/a^+ &= (a+b)^*/\epsilon \\ &= (a+b)^* \end{aligned}$$

$$\begin{aligned} \bullet \quad (a+b)^*/ab &= \{\epsilon \cdot ab/ab, a \cdot ab/ab, bab/ab, aa \cdot ab/ab, \dots\} \\ &= \{\epsilon, a, b, aa, \dots\} \\ &= (a+b)^* \end{aligned}$$

Note:

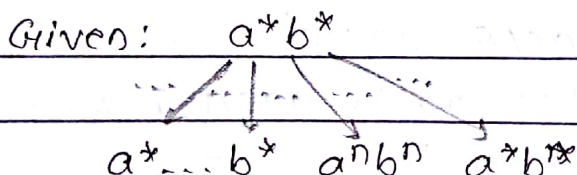
$$(a+b)^*/\text{any other than } \emptyset = (a+b)^*$$

$$\bullet \quad L/\emptyset = \emptyset \quad \bullet \quad a^+/b^+ = \emptyset$$

Subset

Subset (Regular) = need not be Regular

$\therefore$ subset property is not closed.



Note:

• Subset operation is not closed for all types of formal languages.

• Subset of finite language is finite. (So regular)

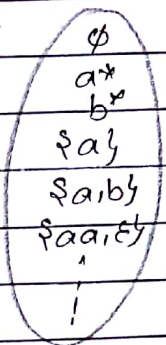
Substitution for Regular languages (Regular Substitution)

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

$$\Sigma^* = \{\epsilon, a, b, ab, ba, \dots\}$$

$$2^{\Sigma^*} = P(\Sigma^*)$$

$$= \{ \cdot$$



set of symbols

set of strings

set of languages

Language:
 • set of strings over Σ .
 • L is a subset of Σ^*
 • $L \in P(\Sigma^*)$

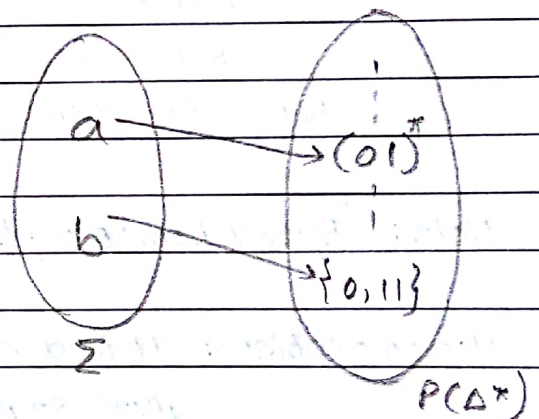
$$f: \Sigma \rightarrow P(\Delta^*)$$

Given: $L = a^*b^*b$ (Regular)

$$f(a) = (01)^* \text{ Regular } \Sigma = \{a, b\}$$

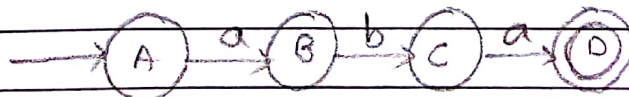
$$f(b) = \{0, 11\} \quad \Delta = \{0, 1\}$$

$$f(L) = (f(a))^* (f(b))^* f(b) \\ = (01)^* (0+11)^* (0+11)$$

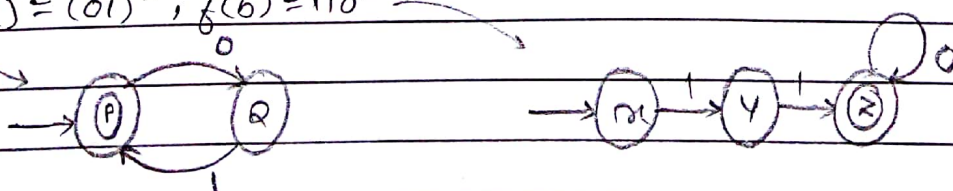


Substitution: It is a function where each input symbol of given regular language over Σ is substituted with some regular over Δ .

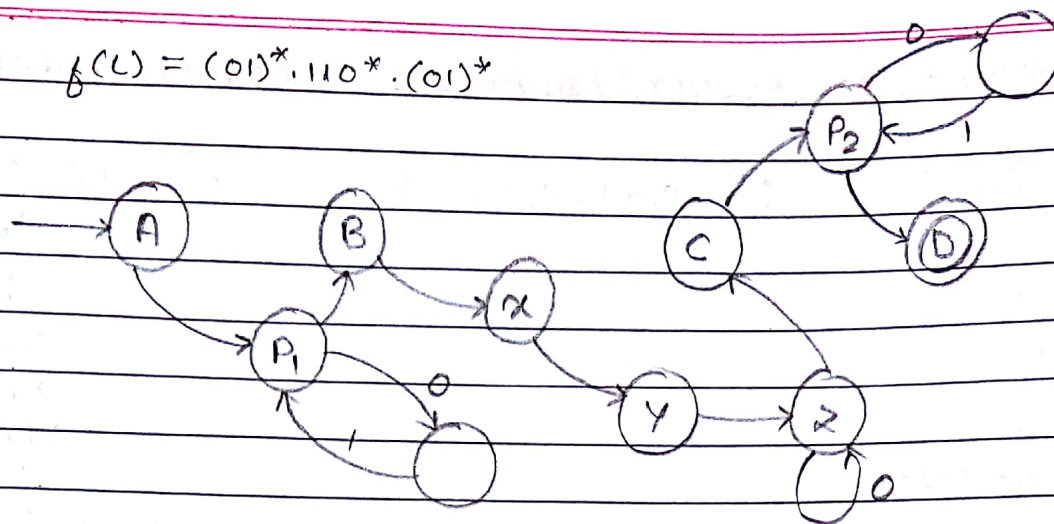
$$L = \{aba\}$$



$$f(a) = (01)^*, f(b) = 110^*$$



$$f(L) = (01)^* \cdot 110^* \cdot (01)^*$$



Homomorphism for Regular languages (String substitution)

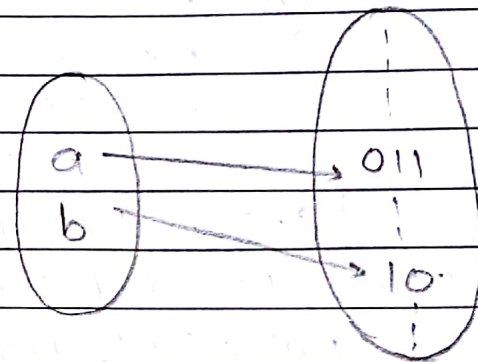
$$h: \Sigma \rightarrow \Delta^*$$

Given: $L = a^*b^*b$ (Regular)

Substitutions $\begin{cases} h(a) = 011 \\ h(b) = 10 \end{cases}$

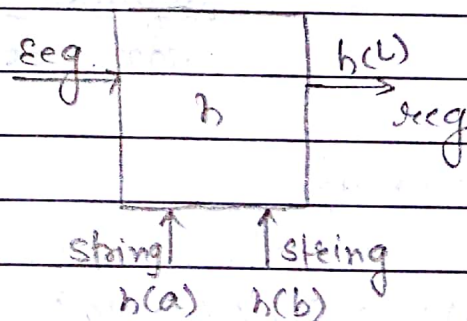
with $\Delta = \{0, 1\}$

strings $h(L) = (011)^*(10)^*10$



Note: "Every homomorphism is substitution."

Homomorphism: It is a substitution where each input symbol of given regular language over Σ is substituted with some string over Δ .



Inverse Homomorphism

$$L = \{011, 11\}$$

Given: $h(a) = 0, h(b) = 1, h(c) = 11$

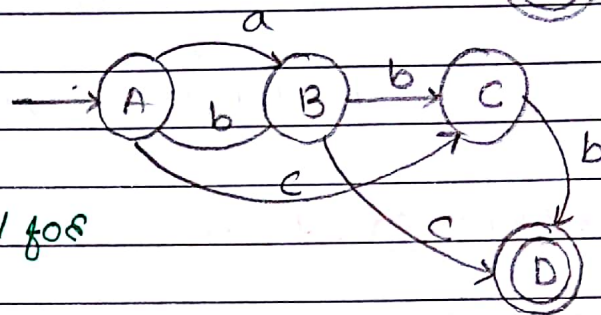
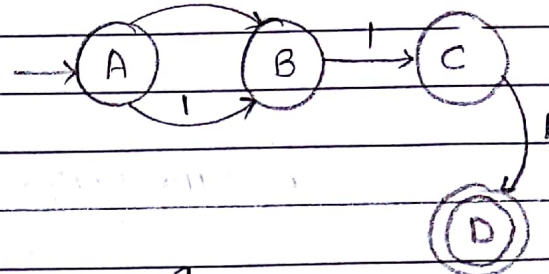
$$h^{-1}(0) = a, h^{-1}(1) = b, h^{-1}(11) = c$$

Each state $\rightarrow 0, 1, 11$.

$$h^{-1}(L) = \{abb, ac, bbb, bc, cb\}$$

- "Inverse Homomorphism is closed for all types of languages."

$$L = \{011, 11\}$$



Half(L)

$$\text{Half}(L) = \frac{1}{2}(L) = \{u \mid uv \in L, |u| = |v|\}$$

$$L = \{ \begin{array}{c} \varepsilon \\ \textcircled{\varepsilon} \end{array}, a, ab, \begin{array}{c} bb \\ \textcircled{b} \end{array}, aba, abaa \}$$

$\begin{array}{c} \varepsilon \\ \textcircled{\varepsilon} \end{array} \begin{array}{c} \varepsilon \\ \textcircled{\varepsilon} \end{array} \quad \begin{array}{c} a \\ x \end{array} \begin{array}{c} a \\ |a|=|b| \end{array} \quad \begin{array}{c} b \\ \textcircled{b} \end{array} \begin{array}{c} b \\ b \end{array} \quad \begin{array}{c} aba \\ x \end{array} \quad \begin{array}{c} abaa \\ ab \quad aa \end{array}$

$$\text{Half}(L) = \{\varepsilon, a, b, ab\} = \text{first half}(L)$$

$$\text{second Half} = \{\varepsilon, b, aa\} \quad \frac{1}{3}(L) = \{\varepsilon, a\} \quad \text{middle } \frac{1}{3}L = \{\varepsilon, b\}$$

$$\text{last } \frac{1}{3}L = \{\varepsilon, a\}$$

$$1. L = (a+b)^*$$

$$\text{Half}(L) = (a+b)^*$$

$$\text{second half}(L) = (a+b)^*$$

$$2. L = a(a+b)^*$$

$$\frac{1}{2}(L) = a(a+b)^*$$

$$\text{second half}(L) = (a+b)^+$$

• Second Half(L) : $\{v \mid uv \in L, |u| = |v|\}$

• one-third(L):

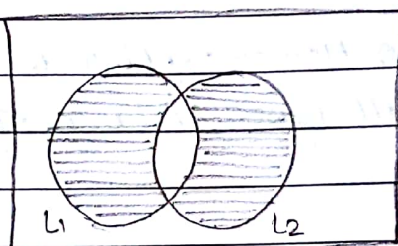
$$\frac{1}{3}(L) = \{x \mid xyx \in L, |x|=|y|=|z|\}$$

$$\text{middle } \frac{1}{3}(L) = \{y \mid xyx \in L, |x|=|y|=|z|\}$$

$$\text{last } \frac{1}{3}(L) = \{z \mid xyx \in L, |x|=|y|=|z|\}$$

Exclusive OR ($L_1 \oplus L_2$)

$$\begin{aligned} L_1 \Delta L_2 &= (L_1 \cup L_2) - (L_1 \cap L_2) \\ &= (L_1 - L_2) \cup (L_2 - L_1) \end{aligned}$$



Symmetric Difference.

Exclusive NOR ($L_1 \odot L_2$)

$$\begin{aligned} \overline{L_1} \cdot \overline{L_2} \cup L_1 \cdot L_2 \\ \text{Reg}_1 \odot \text{Reg}_2 &= \overline{\text{Reg}_1} \cdot \overline{\text{Reg}_2} \cup \text{Reg}_1 \cdot \text{Reg}_2 \\ &= \downarrow \text{Reg} \cdot \downarrow \text{Reg} \cup \text{Reg} \cdot \text{Reg} \\ &= \text{Regular} \end{aligned}$$

Finite Union

$$\begin{aligned} R_1 \cup R_2 \cup R_3 \cup \dots \cup R_k \text{ finite} \\ \therefore \text{Regular} \end{aligned}$$

Finite Intersection

$$\begin{aligned} R_1 \cap R_2 \cap R_3 \cap \dots \cap R_k \\ \therefore \text{Regular} \end{aligned}$$

Finite Difference

$$\begin{aligned} R_1 - R_2 - R_3 - \dots - R_k \\ \therefore \text{Regular} \end{aligned}$$

Finite Concatenation

$R_1, R_2, R_3, \dots, R_k = \text{Regular}$

Finite Subset

finite subset of regular is finite regular.
subset should be finite

Note:

- Finite subset of any language is finite.
- Finite subset operation is closed for all types of languages except infinite languages.

Finite Substitution

Finite substitution over regular is regular.

string $\rightarrow$ finite language $\rightarrow$ Regular.

Infinite Union

$R_1 \cup R_2 \cup R_3 \dots R_k$: Need not be regular.

Infinite Intersection

$R_1 \cap R_2 \cap R_3 \dots R_k$: Need not be regular

Infinite followed by (union, intersection, concatenation,) is not closed for all languages.

i. $\Sigma^* \cup L_1 \cup L_2 \dots = \Sigma^*$

ii. $\{\epsilon\} \cup \{ab\} \cup \{a^2b^2\} \cup \dots = a^n b^n$

Infinite Difference

$R_1 - R_2 - R_3 \dots = \text{Need not be regular.}$

Infinite Concatenation

Need not be regular.

i. $\phi, L_2, L_3 \dots = \phi$

ii. $\{a\}, \{a\}, \{a\} = \{a^\infty\}$

$\hookrightarrow$ one string infinite

Infinite subset, Infinite Substitution over regular is
Need not be regular.

i. $\phi - R_1 - R_2 - R_3 - \dots = \phi$

ii. $\Sigma^* - \{ \epsilon \} - \{ ab \} - \{ a^2 b^2 \} - \dots = \Sigma^* - a^n b^n$

Note:

$F_i \rightarrow$ Finite language

$I_i \rightarrow$ Infinite language

$R_i \rightarrow$ Regular

1. $F_1 \cup F_2 = F$

2. $F_1 \cap F_2 = F$

3. $F_1 \cap F_2 = F$

4. $R_1 \cup R_2 = R$

5. $R_1 \cap R_2 = R$

6. $R_1 - R_2 = R$

7. $I_1 \cup I_2 = I$

8. $I_1 \cap I_2 =$ maybe or may not finite.

9. $I_1 - I_2 =$ ——— " ———

10. $F \cap \text{Any} = F$

11. $F - \text{any} = F$

12. $I \cup \text{any} = I$

13. $F - R =$

14. $F - I =$

15. $R - F = R$

16. $R - I =$

17. $I - R = ?$

18. $I - F =$

19. $F \cup R =$

20. $F \cup I =$

21. $R \cup I =$

22. $R \cap R =$

23. $F \cap I =$

24. $R \cap I =$

25. $\phi - \text{any} = \phi$

26. $\text{Any} - \Sigma^* =$

27. $\phi \cdot \text{Any} = \text{any.}$

28. $\phi \cap \text{Any} = \phi$

Context Free Grammars (CFG)

Type 0

REG (unrestricted grammars)

$$\alpha \rightarrow \gamma$$

(phrase structured grammars)

def. 01: $(V+T)^* \rightarrow (V+T)^*$

def. 02: $(V+T)^+ \rightarrow (V+T)^*$

def. 03: $(V+T)^* V (V+T)^* \rightarrow (V+T)^*$

Type 1

(CSG) Bounded restricted grammars

$$\alpha \rightarrow \gamma$$

$|\alpha| \leq |\gamma|$ and avoid using null productions

$$(V+T)^* V (V+T)^* \rightarrow (V+T)^*$$

Type 2

(CFG)

$$V \rightarrow \text{any}$$

$$V \rightarrow (V+T)^*$$

Eg. $S \rightarrow AaBb \mid ab$

$$A \rightarrow a$$

$$B \rightarrow bA$$

Type 3

(Regular Grammars)

RLG and LRG.

$$V \rightarrow T^* V \mid T^*$$

$$V \rightarrow VT^* \mid T^*$$

1. derivation of string — LMD, RMD, Parse Tree
2. CFGs & CFLs
3. Types of CFG — Ambiguous, Unambiguous.

Derivation of String

Left most derivation (LMD)

Given:

$$S \rightarrow AB$$

$$A \rightarrow ac|ef$$

$$B \rightarrow b|dB$$

$$C \rightarrow d$$

adb = string

left most non-terminal
 (a) b (A) b (B) a
 left most symbol right most non-terminal

$\xrightarrow{SF} S$
 $\Downarrow$ $S \rightarrow AB$
 S derivative $\xrightarrow{SF} AB$
 steps $\Downarrow$ $A \rightarrow ac$
 (S sentential $\xrightarrow{SF} acB$
 forms) $\Downarrow$ $C \rightarrow d$
 $\xrightarrow{SF} adb$
 $\Downarrow$ $B \rightarrow b$
 (string) $\xrightarrow{SF} adb$
 only terminals

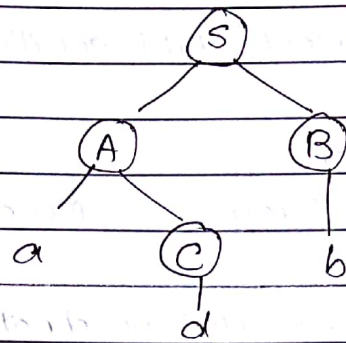
derivation
 length of derivation = 4
 (No. of substitutions)

Right Most Derivation

S
 $\Downarrow$ $S \rightarrow AB$
 AB
 $\Downarrow$ $B \rightarrow b$
 Ab
 $\Downarrow$ $A \rightarrow ac$
 acb
 $\Downarrow$ $C \rightarrow d$
 adb

length of derivation = 4
 5 derivative steps

Parse Tree:



string: adb.

Derivation:

Derive string using start symbol of grammar.

LMD

In each sentential form, left most non-terminal is substituted to derive the string.

RMD

In each sentential form, right most non-terminal is substituted to derive string.

Parse Tree

Each leaf node is terminal and non-leaf node is non-terminal.

Derivation length: No. of substitutions.

SF

It is a sequence of symbols derive using start symbol.

Each sentential form also called as derivative step.

String

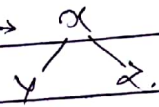
In LMD, RMD, sentential form with only terminal.

Substitution.

Non-terminal is replaced with RHS of its production.

Production Tree

Every production creates parent/child relation. $x \rightarrow yz \rightarrow$



For given CFG and given string,

Q.1. Is LMD and RMD string $\rightarrow$ Need not be same.

Q.2. Will LMD unique $\rightarrow$ LMD need not be unique.

Q.3. Relation of LMD, RMD and parse tree.

No. of parse tree = No. of LMD's = No. of RMD's, ANY string

Q.4. Is every string has one parse tree?

For string, parse tree may be more.

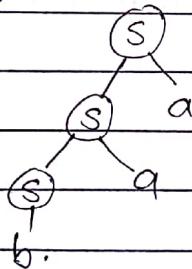
Unambiguous Context free grammar

$\forall$ string,

No. of parse tree = 1.

$S \rightarrow Sa|b$

baa



Note:

- Is given CFG ambiguous?
- Is given CFG unambiguous?

} undecidable

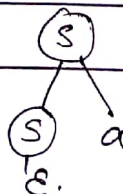
- Is given amb. CFG has any logic?
- decidable.

$\exists$ string,

No. of parse trees > 1 .

$S \rightarrow Sa|a|e$

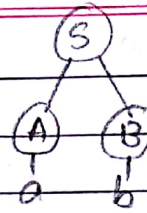
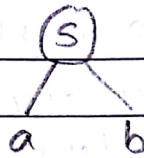
a:



1. $S \rightarrow AB/ab$

$A \rightarrow aA/a$

$B \rightarrow bB/b$



$\therefore$ Ambiguous

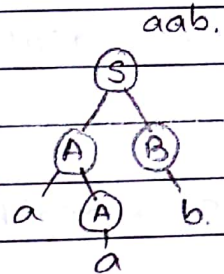
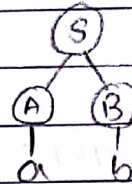
2. $S \rightarrow AB$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

ab

unambiguous



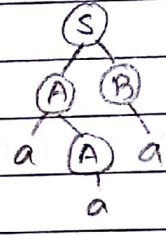
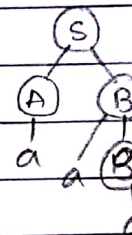
3. $S \rightarrow AB$

$A \rightarrow aA/a$

$B \rightarrow aB/a$

aaa.

$\therefore$ Ambiguous.



CFGs and CFLs

1. $S \rightarrow a/b$ $L = \{a, b\} = a+b$

2. $S \rightarrow AB/a$ $L = \{a\}$

$A \rightarrow a, B \rightarrow \epsilon$

3. $S \rightarrow asb/\epsilon$ $L = a^n \epsilon b^n = a^n b^n$

4. $S \rightarrow aaS/\epsilon$ $L = \{\epsilon, aa, aaaa, \dots\} = (aa)^*$

5. $S \rightarrow Sa/\epsilon$

6. $S \rightarrow as/a/\epsilon$

7. $S \rightarrow asa/\epsilon$ $L = a^*$

8. $S \rightarrow sa/sb/\epsilon$ $(a^* + b^*) \Rightarrow \epsilon.(a+b)^* \Rightarrow (a+b)^*$

9. $S \rightarrow sala/b$ $L = (a+b)a^*$

10. $S \rightarrow salsb/a/b$ $L = (a+b)^+$

11. $S \rightarrow as/bS/\epsilon$ $L = (a+b)^*$

12. $S \rightarrow as/a/b$ $L = a^*(a+b)$

13. $S \rightarrow as/bS/a/b$ $L = (a+b)^* \cup (a+b)^*(a+b)$

14. $S \rightarrow as/bS$ $L = \emptyset$ 15. $S \rightarrow as$ $L = \emptyset$

$$14. \quad S \rightarrow aSa | bSb | \epsilon \quad L = \{ \epsilon, aa, bb, abba, \dots \}$$

(Even length palindrome.)

$$L = \{ ww^R \mid w \in (a+b)^* \}$$

$$15. \quad S \rightarrow aSa | bSb | a | b \quad L = \text{set of odd length strings}$$

$$= \{ w \alpha w^R \mid w \in (a+b)^*, \alpha \in (a+b) \}$$

$$16. \quad S \rightarrow aSa | bSb | a | b | \epsilon \quad L = \text{set of palindrome (odd \& even)}$$

$$= \{ w(a+b)\epsilon w^R \mid w \in (a+b)^* \}$$

$$= \{ w \alpha w^R \mid w \in (a+b)^*, \alpha \in (a+b)\epsilon \}$$

$$17. \quad S \rightarrow aSbb | \epsilon \quad L = \{ \epsilon, abb, aabbbb, \dots \}$$

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

$$18. \quad S \rightarrow aSb | \epsilon \quad L = \{ \epsilon, a^2b, a^4b^2, \dots \}$$

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

$$19. \quad S \rightarrow aSbb | a \quad L = \{ a, aabb, \dots \}$$

$$= a^n a b^{2n} = a^{n+1} b^{2n} \quad / n \geq 0$$

$$20. \quad S \rightarrow aSbb | b \quad L = \{ b, ab^3, \dots \}$$

$$= a^n b b^{2n} = a^n b^{2n+1} \quad / n \geq 0$$

$$21. \quad S \rightarrow aaSbb | \epsilon \quad L = \{ \epsilon, aabb, \dots \}$$

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

$$22. \quad S \rightarrow AB$$

$$A \rightarrow aA | \epsilon \rightarrow a^*$$

$$B \rightarrow bB | \epsilon \rightarrow b^*$$

$$L = \{ \epsilon, ab, \dots \}$$

$$= a^* \cdot b^*$$

$$23. \quad S \rightarrow AB$$

$$A \rightarrow aAb | \epsilon \rightarrow a^n b^n$$

$$B \rightarrow cB | \epsilon \rightarrow c^*$$

$$L(S) = L = L(A) \cdot L(B)$$

$$= a^n b^n \cdot c^*$$

$$= \{ a^m b^n c^k \mid m=n, m, k, c \geq 0 \}$$

$$\begin{aligned}
 24. \quad S &\rightarrow AB & L(S) &= L(A) \cdot L(B) \\
 A &\rightarrow aA/\epsilon & &= a^*b^nc^n \\
 B &\rightarrow bBc/\epsilon \rightarrow b^nc^n & &= \{a^mb^nc^k / n=k, m,n,k \geq 0\}
 \end{aligned}$$

$$\begin{aligned}
 25. \quad S &\rightarrow aSb/A/\epsilon & L(S) &= a^nb^*c^n \\
 A &\rightarrow bA/\epsilon \rightarrow b^* & &= \{a^mb^nc^k / m=k, m,n,k \geq 0\}
 \end{aligned}$$

$$\begin{aligned}
 26. \quad S &\rightarrow AB & a^*b^n L(S) &= a^* \cdot a^nb^n \\
 A &\rightarrow aA/\epsilon & &= a^m \cdot b^n / m \geq n \\
 B &\rightarrow aBb/\epsilon \rightarrow a^nb^n
 \end{aligned}$$

$$\begin{aligned}
 27. \quad S &\rightarrow AB & a^n b^* L(S) &= a^n b^n \cdot b^* \\
 A &\rightarrow aAb/\epsilon & &= a^n b^m / n \leq m \\
 B &\rightarrow bB/\epsilon \rightarrow b^*
 \end{aligned}$$

$$\begin{aligned}
 28. \quad S &\rightarrow SA/\epsilon & (ab)^* & L(S) = (ab)^* \\
 A &\rightarrow ab
 \end{aligned}$$

$$\begin{aligned}
 29. \quad S &\xrightarrow{Q} AS/\epsilon & (ab)^* & L(S) = (ab)^* \\
 A &\rightarrow ab
 \end{aligned}$$

$$\begin{aligned}
 30. \quad S &\rightarrow A/B & a^nb^{2n} + a^{2m}b^m \{a^nb^{2n}\} \cup \{a^{2m}b^m\} \\
 A &\rightarrow aAbb/\epsilon & & \\
 B &\rightarrow aaBb/\epsilon & a^{2m}b^m
 \end{aligned}$$

$$\begin{aligned}
 31. \quad S &\rightarrow AB & L(S) &= \{a^nb^{2n} \cdot a^{2m}b^m\} \\
 A &\rightarrow aAbb/\epsilon & &= \{a^ib^ja^kb^l / j=2i, k=2l\} \\
 B &\rightarrow aaBb/\epsilon
 \end{aligned}$$

$L_1 =$ Regular Language (language of strings)
 $L_2 =$ Set of all regular language
 (Language of Languages)

$\bar{L}_1 =$ Regular language

$\bar{L}_2 =$ Set of all non-regular language



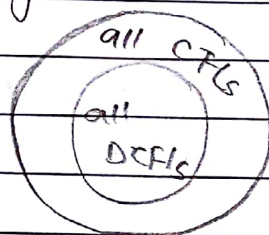
Push Down Automata (Non-deterministic PDA)

- It accepts Context free language (CFL).
- It can be used to verify the syntax of programs.
 (in syntax ~~Any~~ Analysis) (structure)

• $CFG \cong PDA \cong CFL$

• $DPDA < PDA$

- Languages accepted by all DPDAs $\subset$ languages accepted by all PDAs.



($id_1.type == id_2.type$)
 context sensitive

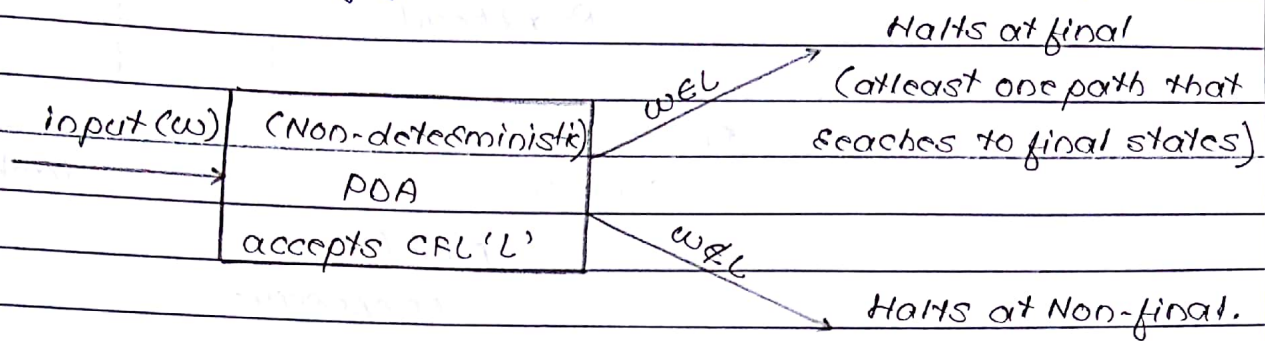
- CFL is also called as **Non-deterministic CFL**.
- Every DCFL is CFL.
- CFL need not be DCFL.
- These are CFLs but not DCFLs.

Applications of PDA

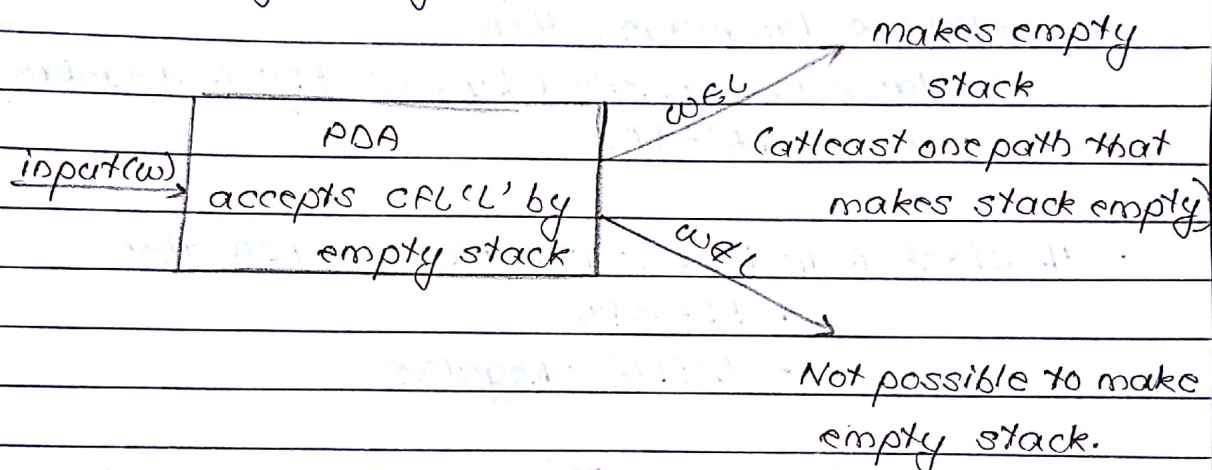
- Used in syntax ~~any~~ analysis of compilers.
- Any application uses single stack. (DFS, Recursion, Expressions)

Definations

- PDA acceptance by final state (universal mechanism)



- PDA acceptance by Empty stack. (Local mechanism)



- PDA acceptance by both final state and empty stack.

Let $L_1 =$ set of all languages accepted by PDA with final state

$L_2 =$ set of all languages accepted by PDA with Empty stack.

$L_3 =$ set of all languages accepted by PDA with both final state & ES.

$L_4 =$ set of all CFLs.

$L_5 =$ set of all languages generated by CFGs.

$$L_1 = L_2 = L_3 = L_4 = L_5 = L_6.$$

$L_6 =$ set of all languages accepted by PDA.

Configuration of PDA:

Tape

aabb

Read head

unbounded memory

Finite

control

stack

push/pop

no operation

Note:

- If PDA never uses a stack (performs no operation) for a particular language, then
 - language accepted by such PDA is Regular language.
 - then $PDA \cong FA$.
- If stack is limited to 10 symbols in PDA, then
 - $PDA \geq FA$
 - $L(PDA) = \text{Regular}$

$PDA(Q, \Sigma, \delta, q_0, F, z_0, \Gamma)$

where

$z_0 \rightarrow$ initial top of stack

$\Gamma \rightarrow$ stack alphabet (set of stack symbols)

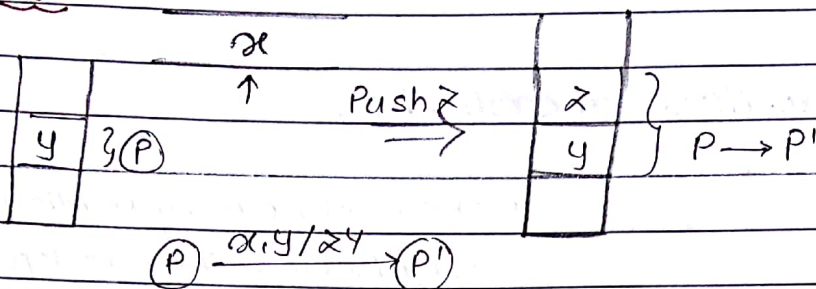
(PDA) $\delta : Q \times \Sigma \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$

(DPDA) $\delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

PS if top of stack no operation

Operations on the stack

Push:



Push 'x' (Read x)

$$\delta(P, x, y) = (P', xy)$$

I: when top of ~~stack~~ stack is y

$$\delta(P, x, \epsilon) = (P', x)$$

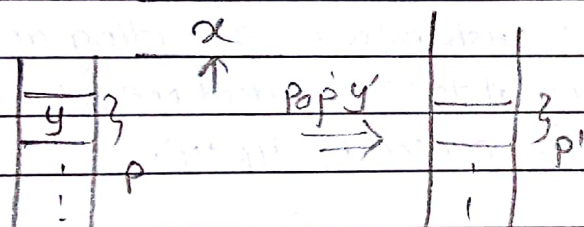
II: without top of stack

$$\delta(P, x, s) = (P', xs)$$

III: with entire stack

POP

pop 'y'



$$(P) \xrightarrow{x, y/\epsilon} (P')$$

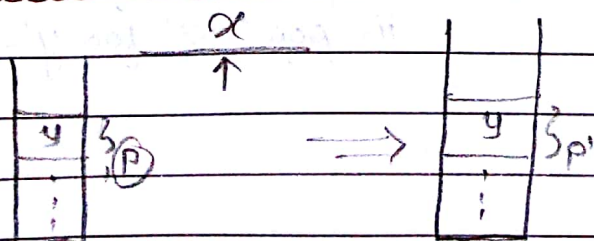
$$\delta(P, x, y) = (P', \epsilon)$$

pop "y"

$$\delta(P, x, y) = (P', \epsilon)$$

$$\delta(P, x, ys) = (P', s)$$

No operation



$$(P) \xrightarrow{x, y/y} (P')$$

$$\delta(P, x, y) = (P', y)$$

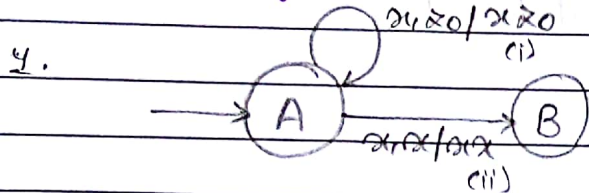
$$\delta(P, x, y) = (P', y)$$

$$\delta(P, \alpha, y) = (P', y)$$

$$\delta(P, \alpha, \epsilon) = (P', \epsilon)$$

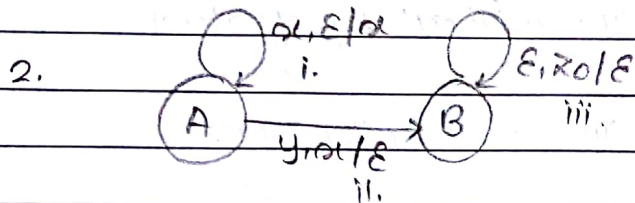
$$\delta(P, \alpha, s) = (P', s)$$

Understanding transitions at each state



i. From state A, by reading input α , by looking at z_0 as top of stack, it pushes α' onto stack and goes to state A.

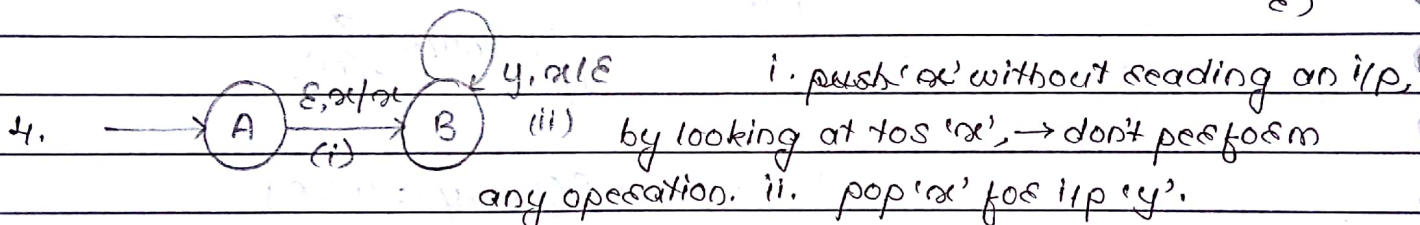
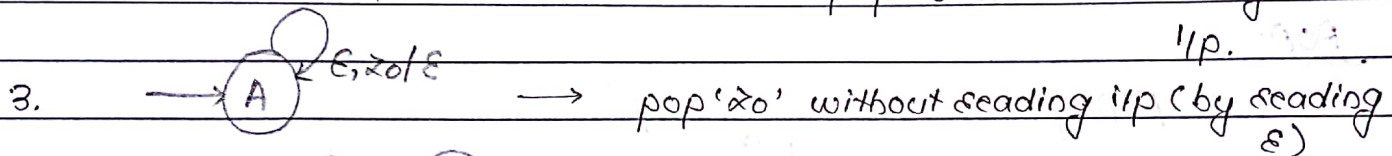
ii. From state A, by reading i/p α , by looking at α as tos it pushes α' onto stack and goes to state B.



i. push ' α ' for every i/p ' α '.

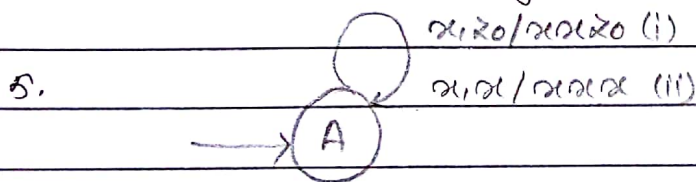
ii. pop ' α ' for i/p ' y '.

iii. pop z_0 without seeing an i/p.



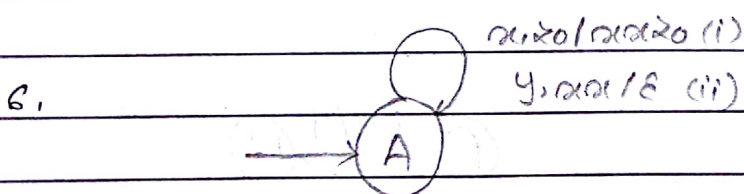
i. push ' α ' without reading an i/p,

by looking at tos ' α ', $\rightarrow$ don't perform any operation. ii. pop ' α ' for i/p ' y '.



i. push ' $\alpha\alpha$ ' for ' α ' (first)

ii. push ' $\alpha\alpha$ ' for ' α ' (every)



i. push ' $\alpha\alpha$ ' for ' α ' (first)

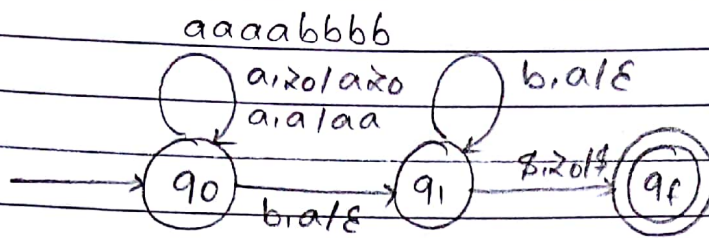
ii. pop ' $\alpha\alpha$ ' for ' y '.

PDA acceptance
by FS by ES Both.

- using $\$$ & z_0
- using $\$$ but no z_0
- using z_0 but no $\$$
- without using $\$$, z_0

Designing PDA for the following CFLs.

$$1. \{a^n b^n \mid n \geq 1\}$$



q_0 : push a's for i/p a's

q_1 : pop a's for i/p b's

$$\delta(q_0, a, z_0) = (q_0, a z_0)$$

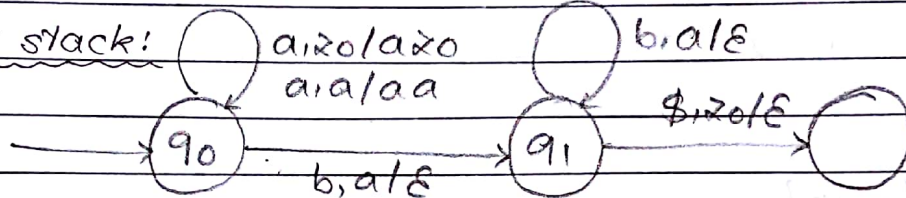
$$\delta(q_0, a, a) = (q_0, a a)$$

$$\delta(q_0, b, a) = (q_1, \epsilon)$$

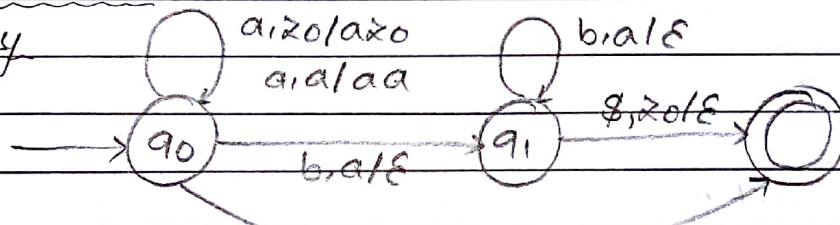
$$\delta(q_1, b, a) = \delta(q_1, \epsilon)$$

$$\delta(q_1, \$, z_0) = \delta(q_f, \$)$$

By empty stack:



By both (final state and empty stack)



$\$, z_0/z_0 \rightarrow$ accepts empty string

$a^m b^n \rightarrow$
 $m = n \rightarrow$ acceptance: $\$, z_0$
 $m > n \rightarrow$ " " ; $\$ a$
 $m < n \rightarrow$ b's are more

$$m < n \Rightarrow m < n \text{ or } m = n$$

$$m \geq n \Rightarrow m > n \text{ or } m = n$$

$$m \neq n \Rightarrow m < n \text{ or } m > n.$$

$$(3) \{a^m b^n \mid m > n, m, n \geq 1\}$$

$$(4) \{a^m b^n \mid m > n\}$$

$$(5) \{a^m b^n \mid m < n, m, n \geq 1\}$$

$$(6) \{a^m b^n \mid m < n\}$$

$$(7) \{a^m b^n \mid m \neq n, m, n \geq 1\}$$

$$(8) \{a^m b^n \mid m \neq n\}$$

$$(9) \{a^m b^n \mid m < n \text{ or } m > n\}$$

$$(10) \{a^m b^n \mid m \leq n\}$$

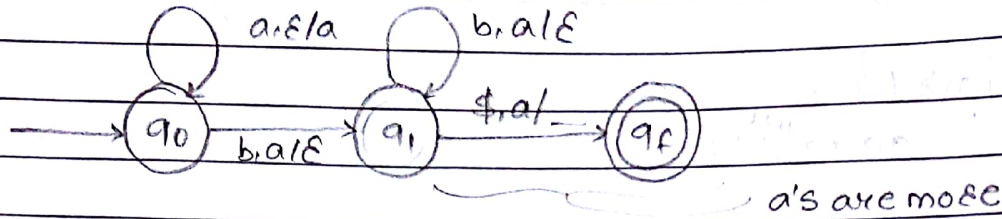
$$(11) \{a^m b^n \mid m \leq n\}$$

$$(12) \{a^m b^n \mid m \leq n\}$$

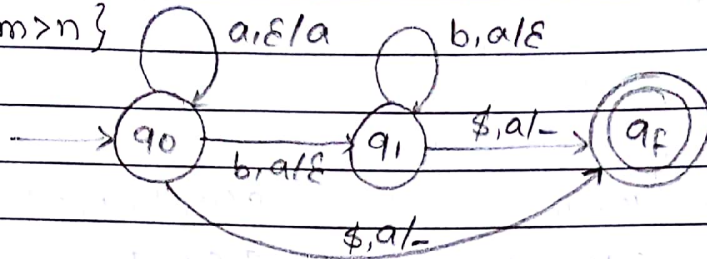
$$(13) \{a^m b^n \mid m \leq n\}$$

$$(14) \{a^m b^n \mid m \leq n\}$$

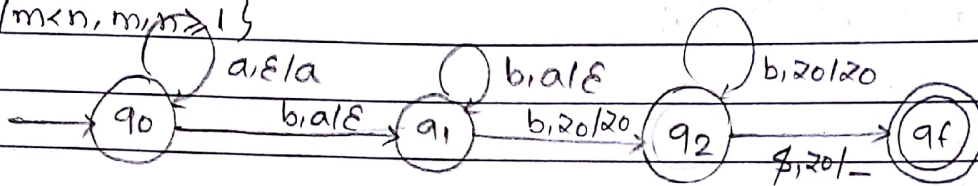
3. $\{a^m b^n \mid m > n, m, n \geq 1\}$



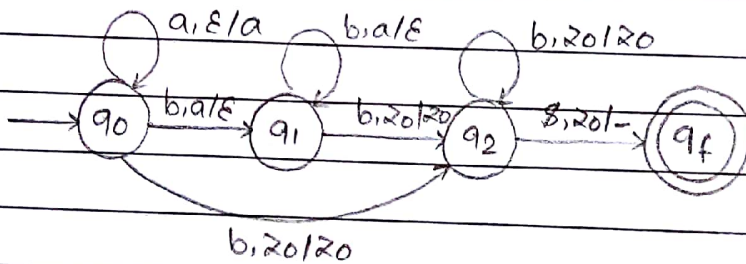
4. $\{a^m b^n \mid m > n\}$



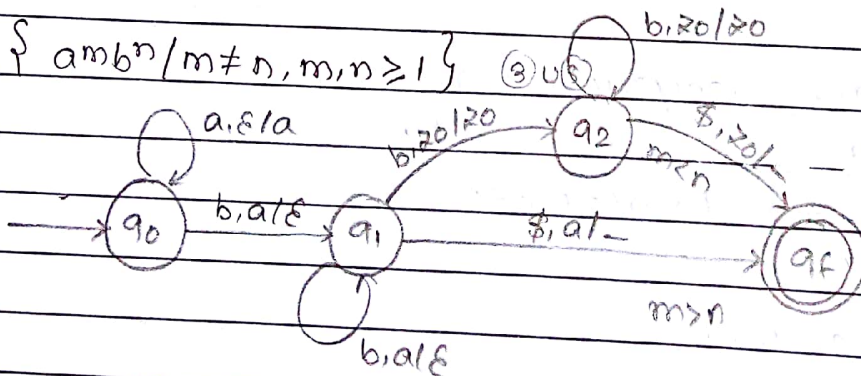
5. $\{a^m b^n \mid m < n, m, n \geq 1\}$



6. $\{a^m b^n \mid m < n\}$ $m=0 \rightarrow b^+$



7. $\{a^m b^n \mid m \neq n, m, n \geq 1\}$



15. $\{w \mid w \in (a+b)^+, na(w) = nb(w)\}$

16. $\{w \mid w \in (a+b)^+, na(w) < nb(w)\}$

17. $\{w \mid w \in (a+b)^+, na(w) < nb(w)\}$

18. $\{w \mid w \in (a+b)^+, na(w) > nb(w)\}$

19. $\{w \mid w \in (a+b)^+, na(w) > nb(w)\}$

20. $\{w \mid w \in (a+b)^+, na(w) \geq nb(w)\}$

21. $\{w \mid w \in (a+b)^+, na(w) \geq nb(w)\}$

22. $\{w \mid w \in (a+b)^+, na(w) \leq nb(w)\}$

$\{w \mid w \in (a+b)^+, na(w) < nb(w)\}$

$\{w \mid w \in (a+b)^+, na(w) \neq nb(w)\}$

$\{w \mid w \in (a+b)^+, na(w) \neq nb(w)\}$

$a^n b^n$

$\{w/w \in a^* b^*, n_a(w) = n_b(w)\}$

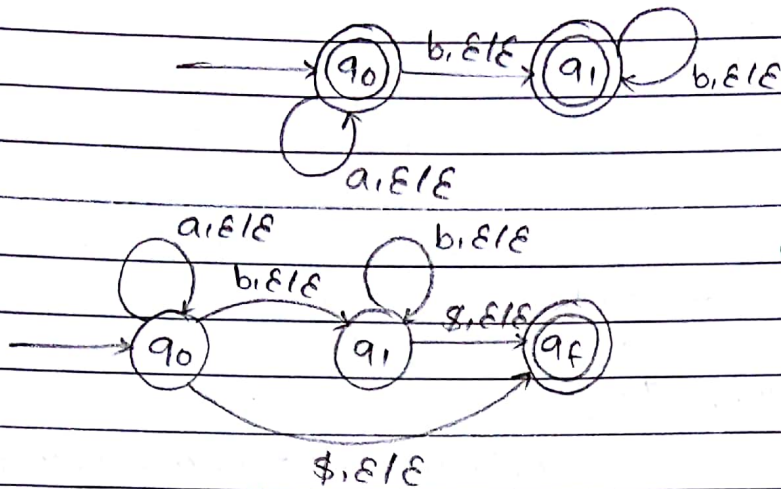
13.

without \$

$a^m b^n / m = n$

$a^m b^n / \text{no. of } a\text{'s} = \text{no. of } b\text{'s}$

$a^n b^n / n \geq 0$



Note:

Any finite automata can be converted to PDA.

14. $\{w/w \in (a+b)^*, n_a(w) = n_b(w)\}$

case 01: $a, \epsilon / a\epsilon$ (push a)

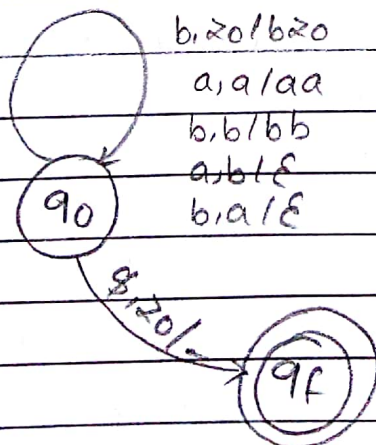
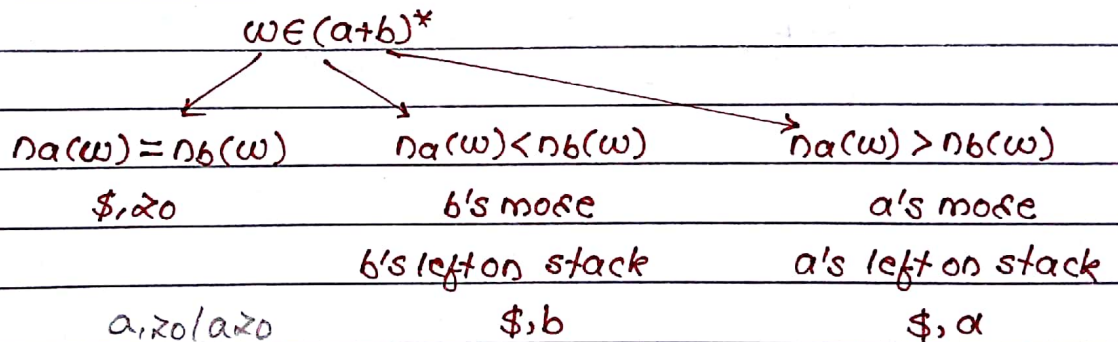
case 04: $b, b / bb$ (push b)

case 02: $b, \epsilon / b\epsilon$ (push b)

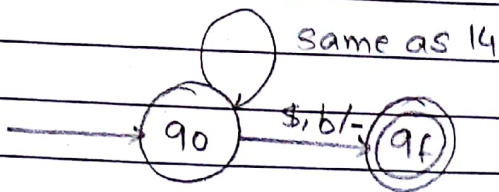
case 05: $a, b / \epsilon$ (pop b)

case 03: $a, a / aa$ (push a)

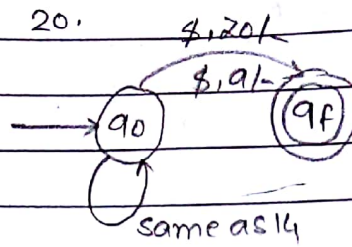
case 06: $b, a / \epsilon$ (pop a)



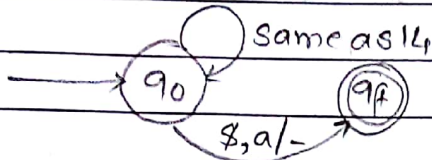
16.



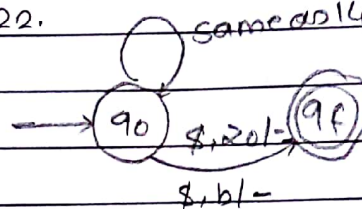
20.



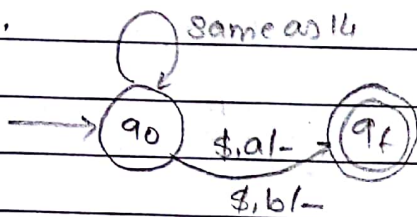
18.



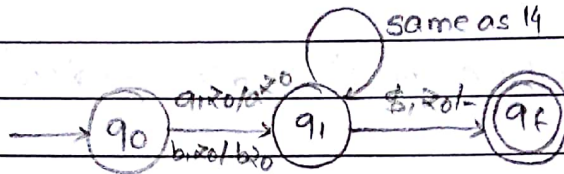
22.



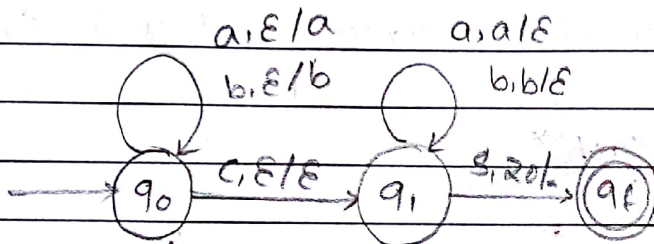
24.



15. $\{w \mid w \in (a+b)^+, n_a(w) = n_b(w)\}$



26. $\{w \mid w \in (a+b)^*\}$



27. $\{w \mid w \in (a+b)^+\}$

28. $\{a^n b^{2n} \mid n \geq 1\}$

29. $\{a^n b^{2n}\}$

30. $\{a^{2n} b^n \mid n \geq 1\}$

31. $\{a^{2n} b^n\}$

32. $\{w \mid w \in (a+b)^*\}$

$a, \epsilon / a$
 $b, \epsilon / b$
 $a, a / aa$
 $b, b / bb$
 $a, b / ab$
 $b, a / ba$

$a, a / \epsilon$
 $b, b / \epsilon$
 $c, \epsilon / \epsilon$
 $c, a / a$
 $c, b / b$

Every student is clever
 $\forall x (S(x) \rightarrow C(x))$

Some student is present

$\exists x (S(x) \wedge P(x))$

$S \rightarrow C \quad \neg S \vee C$

Student is clever.

every student is clever.

If student exist, must be clever.
 no student exist or student clever.

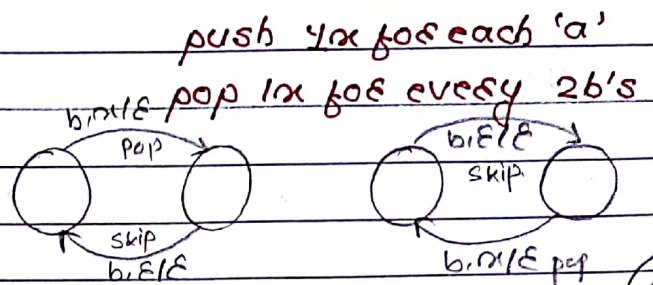
logically same

$a^n b^{2n}$ (#b's = 2.#a's)

I. $a^n b^{2n}$

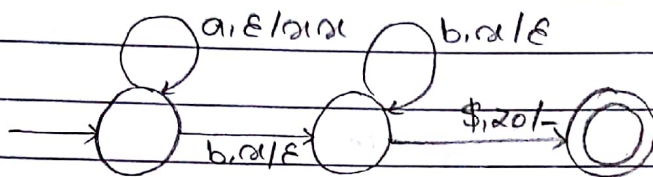
Each 'a' push 2x's
Each 'b' pop 1x.
 $a, \epsilon / xx$
 $a, \epsilon / aa$

II. $a^n b^{2n}$

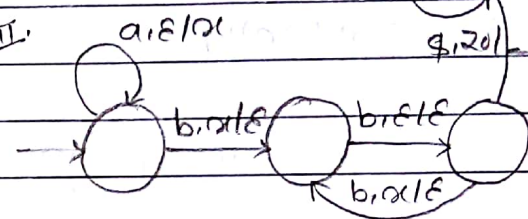


28. $a^n b^{2n} / n \geq 1$

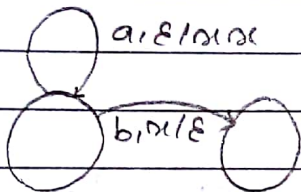
I.



II.



29.



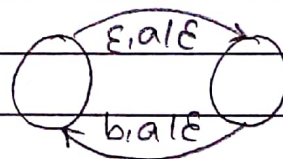
$a^{2n} b^n$

I. $a^{2n} b^n$

each 2a's push 1x
each 'b' pop '1x'

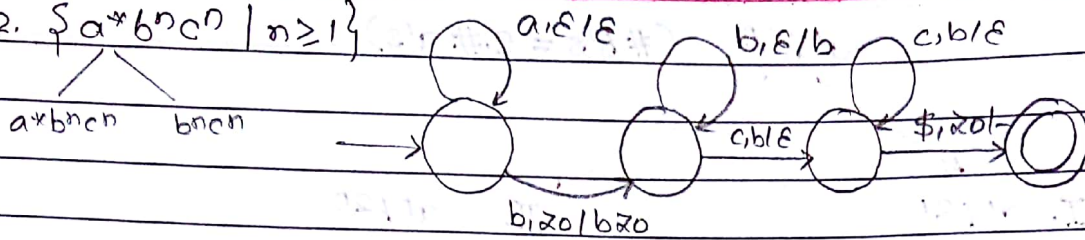
II. $a^{2n} b^n$

push 'a' for each 'a'
pop 2a's for each 'b'
 $b, a / a \epsilon$

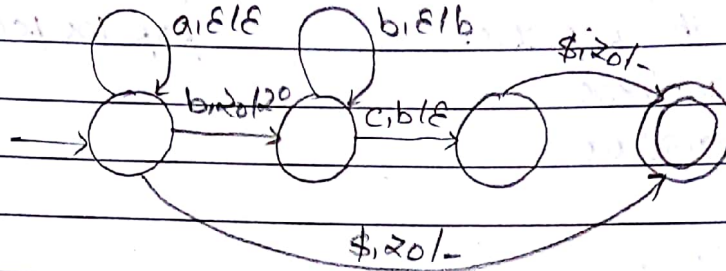


33. $\{a^*b^n c^n\}$ 34. $\{a^n b^* c^n\}$ 35. $\{a^n b^n c^*\}$

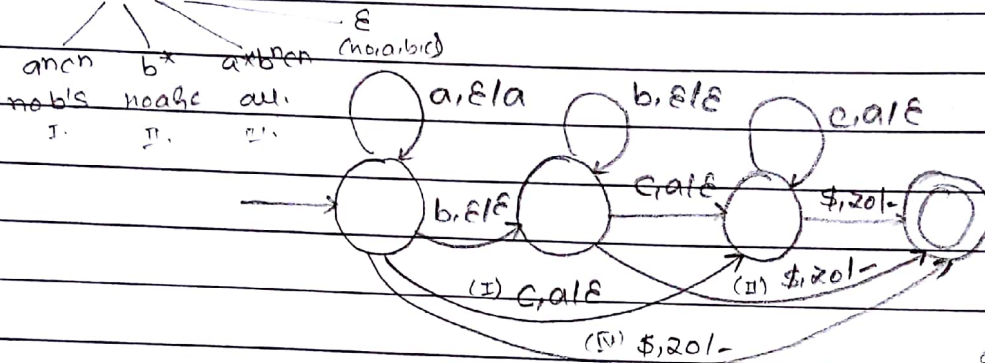
32. $\{a^*b^n c^n \mid n \geq 1\}$



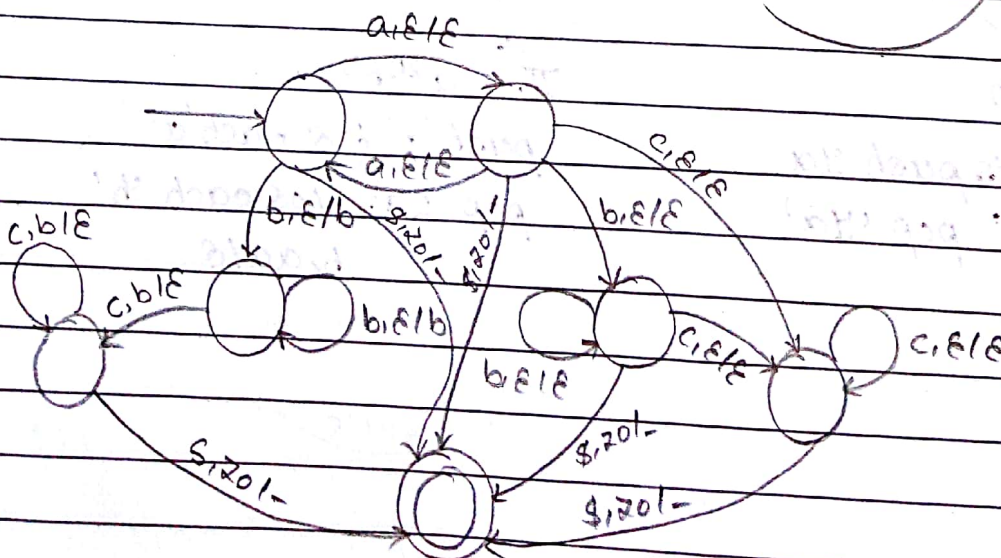
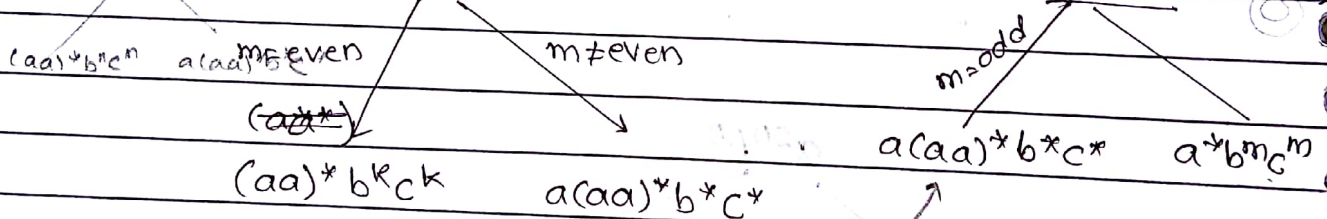
33. $\{a^*b^n c^n\}$



34. $\{a^n b^* c^n\}$



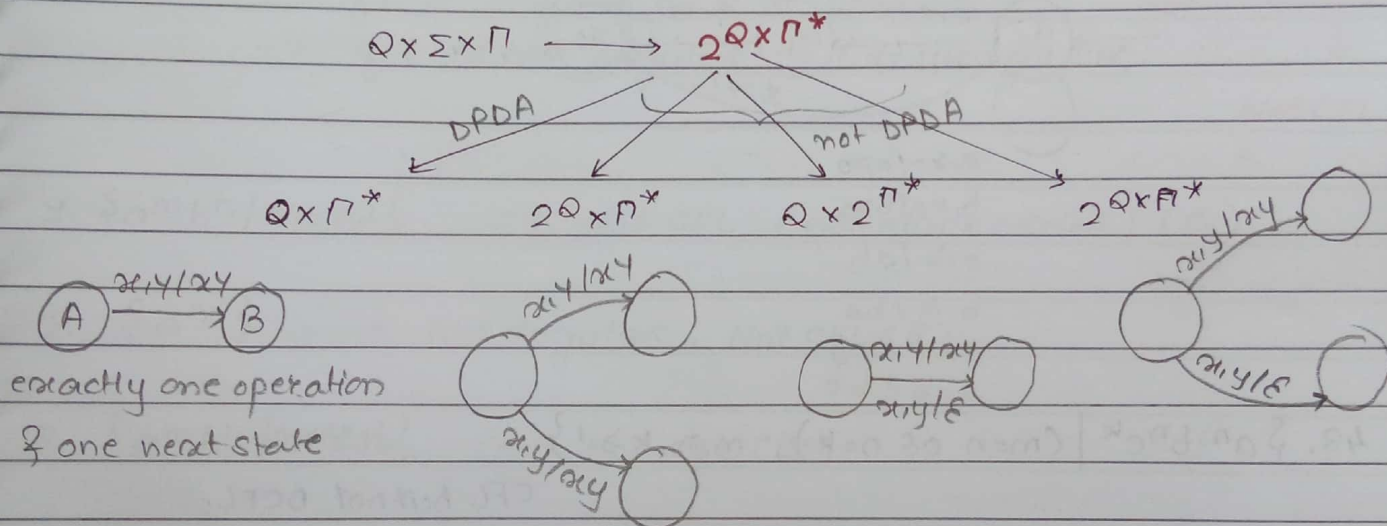
36. $\{a^m b^n c^k \mid \text{if } (m=\text{even}) \text{ then } (n=k)\} \Rightarrow \{a^m b^n c^k \mid (m=\text{odd}) \text{ or } (n=k)\}$



37. $\{ amb^nc^k \mid \text{if } (n=\text{even}) \text{ then } (m=k) \}$
 38. $\{ amb^nc^k \mid \text{if } (k=\text{even}) \text{ then } (m=n) \}$
 39. $\{ amb^nc^k \mid \text{if } (m=n) \text{ then } (k=\text{even}) \}$
 40. $\{ amb^nc^k \mid \text{if } (n=k) \text{ then } (m=\text{even}) \}$
 41. $\{ amb^nc^k \mid \text{if } (m=k) \text{ then } (n=\text{even}) \}$

Note:

Above all problems (41 problems) can be constructed by DPDA. So, above languages are DCFL's.
 Assume, each state for every input symbol if any transition (combination) is missing then that goes to dead state.



42. $\{ ww^R \mid w \in (a+b)^* \}$ CFL but not DCFL.

$w = aa$

$w = ab$

$w = abba$

Q.1. when w^R begins?

present symbol same as previous i/p.

if present i/p same as previous i/p

if

w^R may begin or w may continue.

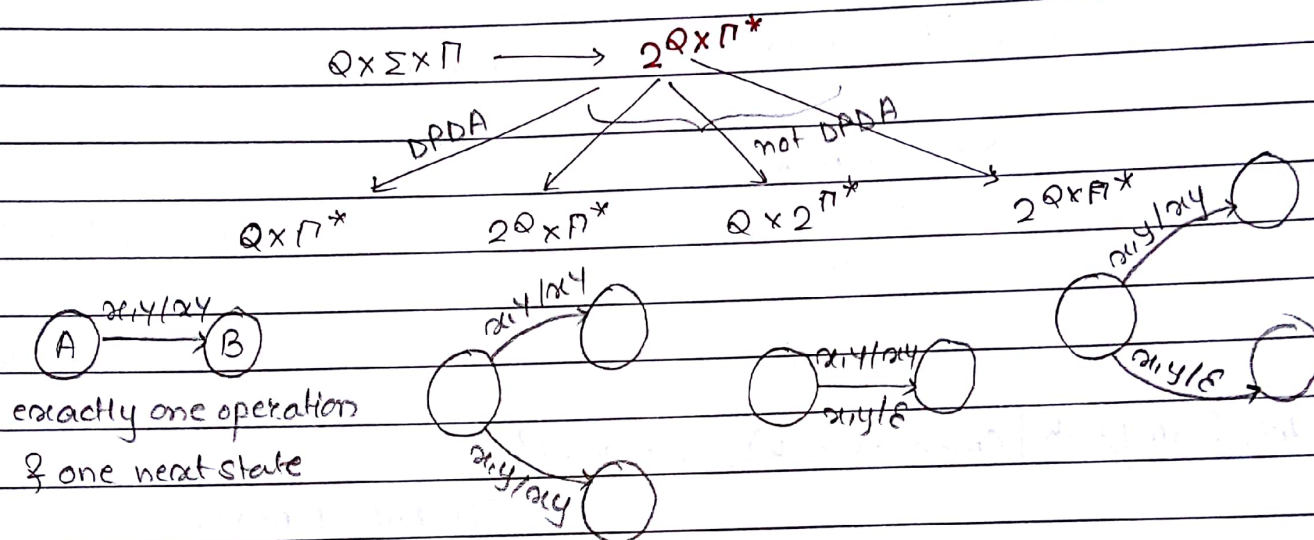
Q.2. if begins, how do you understand?

if (present i/p = top) $\Rightarrow$ we may reverse w^R .

37. $\{a^m b^n c^k \mid \text{if } (n=\text{even}) \text{ then } (m=k)\}$
 38. $\{a^m b^n c^k \mid \text{if } (k=\text{even}) \text{ then } (m=n)\}$
 39. $\{a^m b^n c^k \mid \text{if } (m=n) \text{ then } (k=\text{even})\}$
 40. $\{a^m b^n c^k \mid \text{if } (n=k) \text{ then } (m=\text{even})\}$
 41. $\{a^m b^n c^k \mid \text{if } (m=k) \text{ then } (n=\text{even})\}$

Note:

Above all problems (41 problems) can be constructed by DPDA. So, above languages are DCFL's.
 Assume, each state for every input symbol if any transition (combination) is missing then that goes to dead state.



42. $\{w w^R \mid w \in (a+b)^*\}$ CFL but not DCFL.

$w = aa$

$w = ab$

$w = abba$

Q.1. when w^R begins?

present symbol same as previous i/p.

if present i/p same as previous i/p

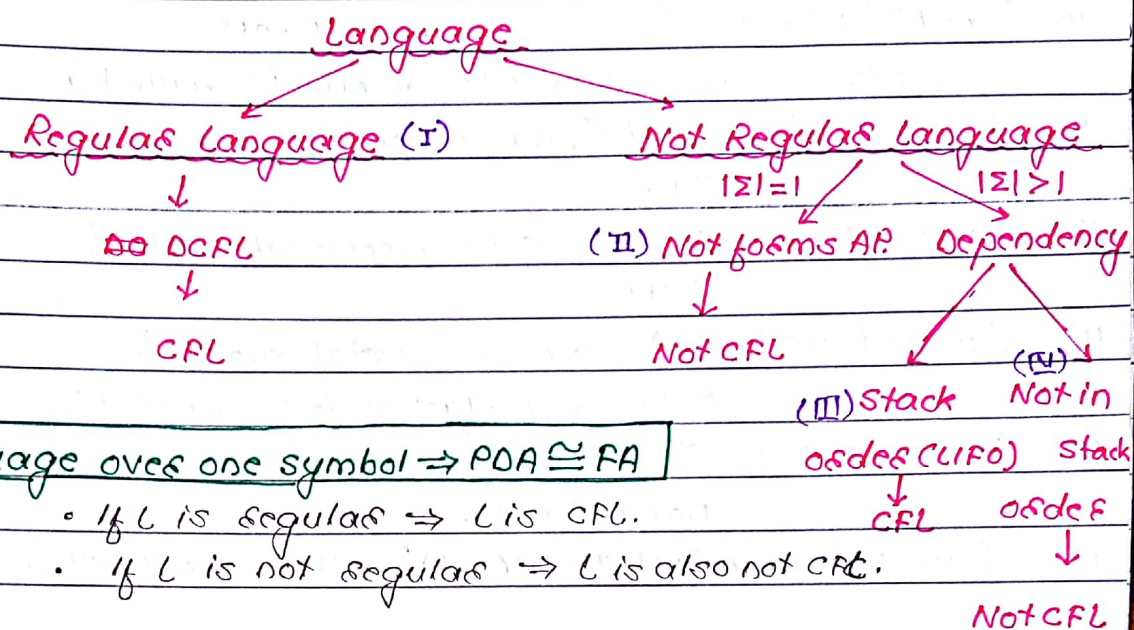
$\Downarrow$

w^R may begin or w may continue.

Q.2. if begins, how do you understand?

if (present i/p = top) $\Rightarrow$ we may reverse w^R .

Identifying CFLs and DCFLs.



Language over one symbol $\Rightarrow PDA \cong FA$

- If L is regular $\Rightarrow L$ is CFL.
- If L is not regular $\Rightarrow L$ is also not CFL.

1. $\{a^m b^n / m = n^2\} \rightarrow$ Not CFL (not stack order) (IV)

2. $\{a^{n^2}\} \rightarrow$ Not regular, Not CFL (II)

3. $\{a^m b^n / (m!) \leq n\} \rightarrow$ Not CFL (IV)

4. $\{a^m b^n / \gcd(m, n) = 1\} \rightarrow$ Not CFL.

5. $\{a^m b^n c^k / m = n = k\} \rightarrow a^n b^n c^n \therefore$ Not CFL

6. $\{a^m b^{2m} c^{3m}\} \rightarrow$ Not CFL. (IV)

7. $\{aw / na(w) = nb(w), w \in (a+b)^*\} \rightarrow$ DCFL

8. $\{wb / na(w) = nb(w), w \in (a+b)^*\} \rightarrow$ DCFL

9. $\{a^m b^n / m = n \text{ or } m < n\} \rightarrow$ DCFL

10. $\{a^m b^n \mid m=n \text{ or } m=2n\} \rightarrow \text{CFL but not DCFL.}$

min: No. of a's = no. of b's max: no. of a's = 2 x no. of b's.

11. $\{a^m b^n \mid n \leq m \leq 2n\} \text{ CFL but not DCFL.}$

$\{w \mid n_a(w) = n_b(w) + 1, w \text{ ends with } b\}$

12. $\{a^m c w c^R \mid w \in (a+b)^*, c \text{ is a special}\} \text{ DCFL.}$

13. $\{w c w^R b \mid w \in (a+b)^*, c \text{ is a special symbol}\} \text{ DCFL}$

14. $\{w c w \mid w \in (a+b)^*, c \text{ is a special symbol}\} \text{ not CFL. not stack order. i.e. not CFL.}$

15. $\{w c w \mid w \in (a+b)^*, c \in (a+b)\} \text{ not CFL.}$

16. $\{w c w^R \mid w \in (a+b)^*, c \in (a+b)\} \text{ CFL but not DCFL.}$

17. $\{a^m b^n \mid n = 100m\} = a^m b^{100m} \rightarrow \text{DCFL}$

18. $\{a^m b^n \mid m = n^n\} \Rightarrow \text{not CFL}$

19. $\{a^m b^n \mid m = 2^n\} \Rightarrow \text{not CFL.}$

20. $\{a^{m+n} b^n c^m\} \text{ DCFL.}$

21. $\{a^m b^{m+n} c^n\} \text{ DCFL}$

22. $\{a^m b^n c^{m+n}\} \text{ not CFL. DCFL.}$

23. $\{a^{m+n} b^{m+n} c^{m+n}\} \text{ not CFL}$

24. $\{a^{m+n} b^m c^{m+n}\} c^m \text{ not CFL.}$

25. $\{a^{m+n} b^{n+k}\} a^* b^* \text{ (Regular)}$

26. $\{a^m b^n c^k \mid \text{if } (m=n) \text{ then } k=\text{even}\} \rightarrow \text{DCFL}$

27. $\{a^m b^n c^k \mid \text{if } (m=n) \text{ then } (n=k)\} = \{a^m b^n c^k \mid (m \neq n) \text{ or } (n=k)\}$

$m=n=k$

$m \neq k$

$m \neq n$

$a^p b^p c^p$

$a^q b^r c^s \mid q \neq r$

$a^q b^r c^s \mid q \neq r$

$a^* b^t c^t$

$a^q b^p c^p \mid q \neq p$

$\therefore \text{CFL but not DCFL.}$

$\{a^i b^j c^i \mid i=j \text{ or } i \neq j\}$

28. $\{ xww^R / w, x \in (a+b)^+ \}$ CFL but not DCFL
29. $\{ ww^R x / w, x \in (a+b)^+ \}$ — " —
30. $\{ w x w^R / w, x \in (a+b)^+ \}$ Regular
31. $\{ ww / w \in (a+b)^+ \}$ Not CFL
32. $\{ \bar{w}w / w \in (a+b)^+ \}$ CFL but not DCFL.
33. $\{ w x w / w, x \in (a+b)^+ \}$ Not regular, not CFL.
34. $\{ ww x / w, x \in (a+b)^+ \}$ Not CFL
35. $\{ x w x / w, x \in (a+b)^+ \}$ Not CFL.
36. $\{ a^n b^{2n} \} \cup \{ d a^{2n} b^n \}$ DCFL
37. $\{ a^n b^{2n} c \} \cup \{ a^{2n} b^n d \}$ CFL but not DCFL.
38. $\{ a^m b^n c^k d^l / \text{if } (m+n=k) \text{ then } (d=\text{even}) \}$ DCFL
39. $\{ a^{2^{n^2}} \}$ not regular $\rightarrow$ not CFL.
40. $\{ a^m b^n c^k d^l / m+n=k+l \}$ DCFL
- 41.

Note: Any CFL can be constructed with only one state
(CFG $\Rightarrow$ PDA algorithm)

$L = \{ ww / w \in (a+b)^+ \}$ No odd length string

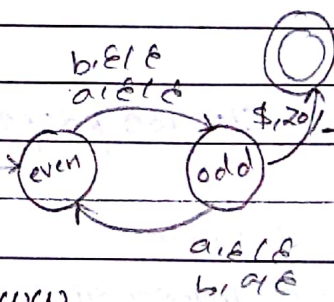
$\bar{L} = \{ \bar{w}w / w \in (a+b)^+ \}$ Even length in form of ww

$= \Sigma^* - L$ even length not in ww

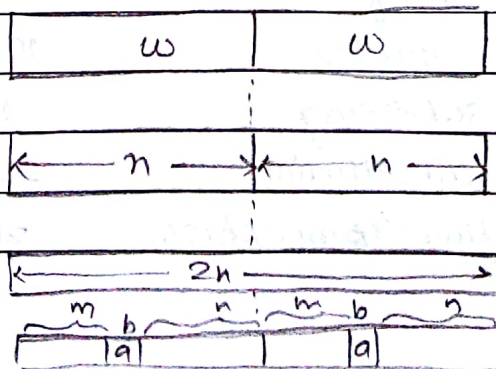
$= \Sigma^* - \{ ww / w \in (a+b)^+ \}$ Every odd length

$\bar{w}w =$ All odd length strings.

Even length not in form of ww



ww logic



$\{n+m\}$ Between same positions of $2w$'s length $= m+n = n+m$

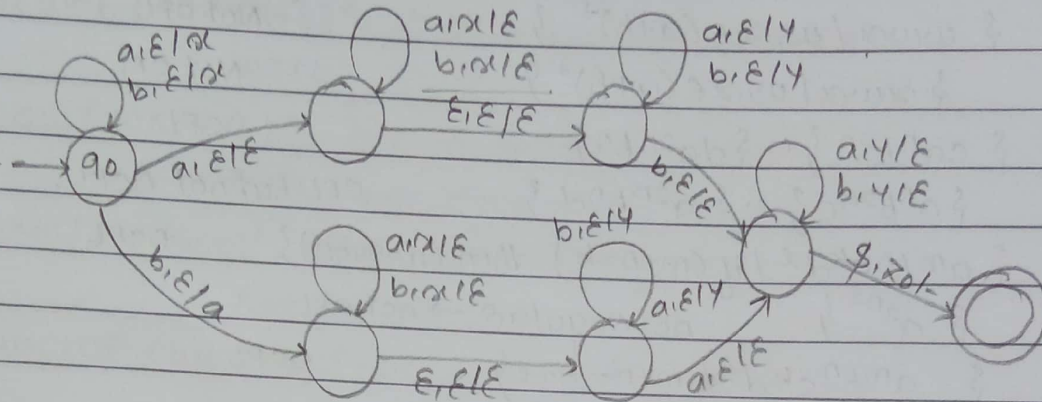
To show not in ww , we need different symbol somewhere in the even length string.

$\exists \alpha (uv = \alpha, |u| = |v|, \text{ same position})$

Diagram illustrating the decomposition of a string into three parts: m symbols, n symbols, and $m+n$ symbols. The string is divided into segments of length a and b .

Diagram illustrating the sequence of operations for the postfix expression $a * b + 1$ using a stack:

- Initial state: Stack is empty, $ip \rightarrow 0$, $final$.
- Operation 1: $push\ m$ (push a onto the stack). Stack: a , $ip \rightarrow 1$.
- Operation 2: $pop\ m$ (pop a from the stack). Stack: (empty), $ip \rightarrow 2$.
- Operation 3: $push\ m$ (push b onto the stack). Stack: b , $ip \rightarrow 3$.
- Operation 4: pop (pop b from the stack). Stack: (empty), $ip \rightarrow 4$.
- Final state: $ip \rightarrow 4$, $final$.



Complement of CFL $\rightarrow$ need not be CFL.
complement of DCFL $\rightarrow$ DCFL.

If not in w/w form and even length
 6 symbols
 b b @ a a b a b b @ a a b a
 push pop's push pop's
 0's (2) (2)

Closure properties of CFLs and DCFLs

- | | | |
|-------------------|---------------------|--------------------------|
| 1. Union | 8. positive closure | 15. $b^*(L)$ |
| 2. Intersection | 9. subset | 16. Finite union |
| 3. Complement | 10. prefix | 17. Finite Intersection |
| 4. Difference | 11. suffix | 18. Finite difference |
| 5. concatenation | 12. substring | 19. finite concatenation |
| 6. Reversal | 13. substitution | 20. finite subset |
| 7. Kleene closure | 14. Homomorphism | 21. finite substitution |

22. Infinite union

25. Infinite concatenation

23. Infinite Intersection

26. Infinite subset

24. Infinite Difference

27. Infinite substitution

28. Regular closure

DCFLs (closed) $\rightarrow$ Complement, prefix

CFLs (not closed) $\rightarrow$ Intersection, complement, difference
(finite $\cap$) (finite diff.)

Note:

1. Subset is not closed for all languages (Regular, DCFLs, CFLs, ...)
2. Infinite (union, Intersection, difference, ...) are not closed for all languages.
3. Inverse homomorphism is closed for all languages.
4. Finite subset is closed for all languages.

Above four points is not applicable for : domain of finite language and domain of infinite language.

Union

DCFLs (not closed)

$DCFL_1 \cup DCFL_2 =$ Need not be DCFL.
(always CFL)

CFLs (closed)

$CFL_1 \cup CFL_2 = CFL$

Eg.

$CFL_1 = a^n b^n \rightarrow S_1 \rightarrow a S_1 b \mid \epsilon$

$CFL_1 \cup CFL_2 : S \rightarrow S_1 \mid S_2$

$CFL_2 = a^{2n} b^n \rightarrow S_2 \rightarrow a a S_2 b \mid \epsilon$

$S_1 \rightarrow a S_1 b \mid \epsilon$

$S_2 \rightarrow a a S_2 b \mid \epsilon$

Eg. (for not DCFL)

$DCFL_1 = a^n b^n$

$DCFL_1 \cup DCFL_2 = \{ a^m b^n \mid m=n \text{ or } m=2n \}$

$DCFL_2 = a^{2n} b^n$

$= \text{not DCFL}$

$= CFL$

CFL but not DCFL
 $ww^R, \overline{ww}$
 not CFL $\rightarrow ww, a^n b^n c^n$

$\emptyset; \Sigma^*$ Eg
 $a^n b^n \rightarrow$ DCFL (non-reg)

Intersection

DCFLs (not closed)

$DCFL_1 \cap DCFL_2 =$ need not be DCFL.

CFLs (not closed)

$CFL_1 \cap CFL_2 =$ need not be CFL.

Eg. (not CFL)

$CFL_1 \rightarrow a^n b^n c^*$

$CFL_2 \rightarrow a^* b^n c^n$

$\therefore CFL_1 \cap CFL_2 = a^n b^n c^n$

DCFL₁

DCFL₂

not DCFL, CFL.

$DCFL_1 \cap DCFL_2 \rightarrow$ always CSL, need not be CFL, DCFL.

$CFL_1 \cap CFL_2 \rightarrow$ always CSL.

Complement

$\overline{DCFL} = DCFL$

$\overline{CFL} =$ need not be CFL.

Algo: Interchange finals and non-finals.

Eg. $L = \{ \overline{ww} \mid w \in (a+b)^* \} \rightarrow$ CFL

$\overline{L} = \{ ww \mid w \in (a+b)^* \} \rightarrow$ not CFL

Eg. $a^n b^n \rightarrow$ DCFL

$\overline{a^n b^n} \rightarrow$ DCFL ($\Sigma^* - a^n b^n$)

$L = \overline{a^n b^n c^n} \rightarrow$ not CFL

$\{ a^m b^n c^k \mid m \neq n \text{ or } n \neq k \text{ or } m \neq k \}$

$\cup (a+b)^* (ba+cb+ca+ac) (a+b)^* \cup a^+ + b^+ + c^+$

Difference

Not closed for both DCFLs and CFLs.

$DCFL_1 - DCFL_2 =$ need not be DCFL.

$DCFL_1 - DCFL_2 = DCFL_1 \cap \overline{DCFL_2}$

$= DCFL_1 \cap DCFL$

$=$ need not be DCFL.

$CFL_1 - CFL_2 =$ need not be CFL.

$CFL_1 - CFL_2 = CFL_1 \cap \overline{CFL_2}$

not closed

$=$ need not be CFL.

Concatenation

CFLs (closed)

$$L_1 = a^n b^n \rightarrow S_1 \rightarrow a S_1 b / \epsilon \quad L_2 = a^{2n} b^n \rightarrow S_2 \rightarrow a a S_2 b / \epsilon$$

$$S \Rightarrow S_1 S_2$$

$$S_1 \rightarrow a S_1 b / \epsilon, \quad S_2 \rightarrow a a S_2 b / \epsilon$$

DCFLs (not closed)

$$L_1 = a^n b^n \text{ (DCFL)} \quad L_2 = a^{2n} b^n \text{ (DCFL)}$$

$$L_1 \cdot L_2 = a^n b^m, a^{2m} b^m \rightarrow \text{CFL but not DCFL.}$$

$$\text{DCFL}_1 \cdot \text{DCFL}_2 = \text{need not be DCFL} \\ = \text{always CFL.}$$

Reversal

$$(\text{DCFL})^{\text{Rev}} = \text{Need not be DCFL.}$$

$$(\text{CFL})^{\text{Rev}} = \text{CFL.}$$

$$L = (\{a^n b^n\} \cup \{a^{2n} b^n\}) = \text{DCFL.}$$

$$\text{Eg. } L = a^n b^n \text{ (CFL)}$$

$$L^{\text{Rev}} = (\{b^n a^n\} \cup \{b^n a^{2n}\})$$

$$S \rightarrow a S b / \epsilon$$

not DCFL but CFL.

$\Downarrow$ reverse every R_k

$$S \rightarrow b S a / \epsilon$$

$$L^{\text{Rev}} = b^n a^n$$

$$(\text{DCFL})^{\text{Rev}} = \text{always CFL.}$$

Kleene closure

$$(\text{DCFL})^* = \text{Need not be DCFL.}$$

$$(\text{CFL})^* = \text{CFL}$$

$$\text{DCFL} = \{a^n b^n\} \cup \{a^{2n} b^n\} = L$$

$$L = a^n b^n \quad S \rightarrow a S b / \epsilon$$

$\Downarrow$

$$L^* = (\{a^n b^n\} \cup \{a^{2n} b^n\})^* \\ \text{not DCFL (always CFL)}$$

$$L^*: \alpha \rightarrow S \alpha / \epsilon$$

$$S \rightarrow a S b / \epsilon$$

also $L^+ = \dots$

Positive closure

$$(\text{DCFL})^+ = \text{Need not be DCFL} \\ \text{always CFL.}$$

$$(\text{CFL})^+ = \text{CFL}$$

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

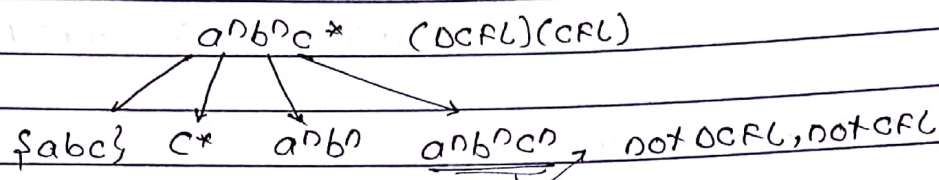
$$S \rightarrow a S b / \epsilon$$

$$L^+: \alpha \rightarrow S \alpha / S \\ S \rightarrow a S b / \epsilon$$

Subsets

Subset(DCFL) = Need not be DCFL.

subset(CFL) = Need not be CFL.



Substitution

DCFL substitution
(not closed)

$$L = \{c a^n b^n\} \cup \{d a^{2n} b^n\}$$

$$\left. \begin{array}{ll} f(c) = \{\epsilon\} & f(d) = \{\epsilon\} \\ f(a) = \{a\} & f(b) = \{b\} \end{array} \right\} \text{DCFL}$$

$$f(L) = \{a^n b^n\} \cup \{a^{2n} b^n\}$$

not DCFL

CFL substitution

(closed)

$L = a^n b^n$ Given CFL

$$\left. \begin{array}{l} f(a) = 0^m 1 2^m \\ f(b) = 0^{2k} 1^k \end{array} \right\} \text{CFLs}$$

$$f(L) = (f(a))^n (f(b))^n$$

$$= (0^m 1 2^m)^n (0^{2k} 1^k)^n$$

$$S \rightarrow x 5 y 1 \epsilon$$

$$x \rightarrow 0 x 1 1 / \epsilon, y \rightarrow 0 0 y 1 / \epsilon$$

Homomorphism

closed for CFLs, not closed for DCFLs.

Inverse Homomorphism

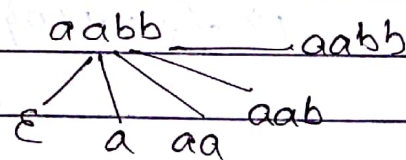
closed for CFLs and DCFLs.

Prefix : closed for both CFLs & DCFLs

$$L = a^n b^n \Rightarrow S \rightarrow a s b 1 \epsilon$$

$$\text{prefix}(L) = \{a^n b^m \mid m \geq n\}$$

$$S \rightarrow \epsilon / a s a / a s b$$



Prefix property is different from prefix operation.

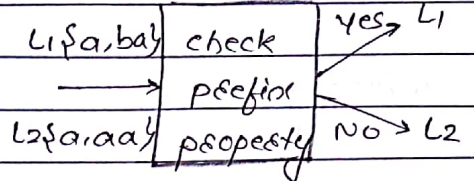
prefix operation

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



$$\text{prefix}(L) = \{\epsilon, a, ab\}$$

prefix property



Suffix

DCFLs $\Rightarrow$ not closed

CFLs $\Rightarrow$ closed

$$\text{DCFL} = \{ca^n b^n\} \cup \{da^{2n} b^n\}$$

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

$$\text{suffix}(L) = \text{not DCFL}$$

$$S \rightarrow asb \mid \epsilon$$

$$= \text{CFL}$$

$$\text{suffix}(L) \quad S \rightarrow \epsilon \mid b \mid sb \mid asb \mid \{a^n b^n \mid n \geq m\}$$

Substring

CFLs $\Rightarrow$ closed

DCFLs $\Rightarrow$ not closed

DCFLs

Finite:

X

X

X

X

✓

X

Infinite:

X

X

X

X

X

X

CFLs

Finite:

✓

X

X

✓

✓

✓

Infinite:

X

X

X

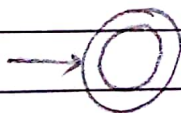
X

X

X

Understanding PDA:

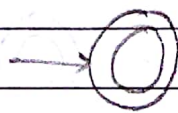
1.



with z_0 & $\$$

$$L = \emptyset$$

2.

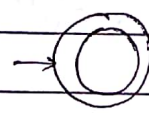


z_0 is present

and no $\$$

$$L = \{\epsilon\}$$

3.

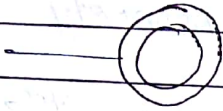


z_0 is not present

but $\$$ exist.

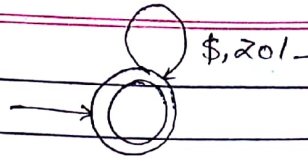
$$L = \emptyset$$

4.



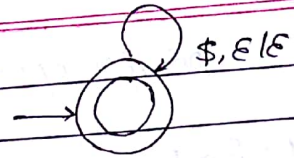
without z_0 & $\$$
 $L = \{\epsilon\}$

5.



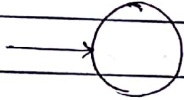
with $\$$ & z_0
 $L = \{\epsilon\}$

6.



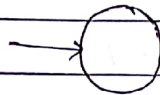
with $\$$ but no z_0
 $L = \{\epsilon\}$

7.



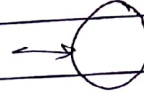
with z_0 & $\$$
 PDA by empty stack
 $L = \emptyset$

8.



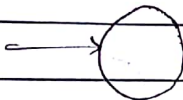
z_0 is present
 & no $\$$.
 $\therefore$ PDA by Empty stack.
 $L = \emptyset$

9.



z_0 is not present
 but $\$$ exist
 PDA by ES.
 $L = \emptyset$

10.



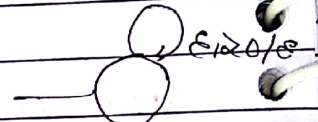
without z_0 & $\$$
 PDA by ES.
 $L = \{\epsilon\}$

11.



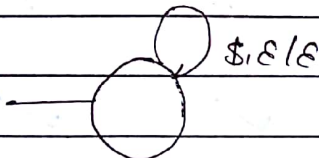
with $\$$ & z_0
 PDA by ES.
 $L = \{\epsilon\}$

12.



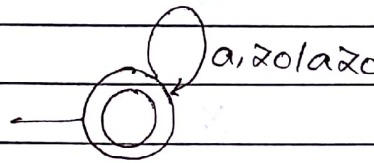
with z_0 & no $\$$
 PDA by ES.
 $L = \{\epsilon\}$

13.



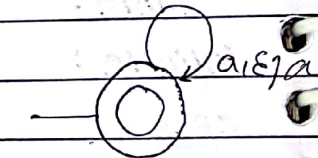
$z_0^x, \$ \checkmark$ E.S.V
 $L = \{\epsilon\}$

14.



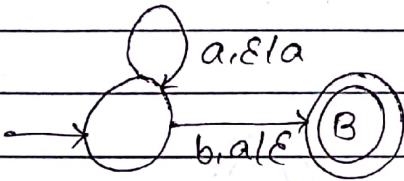
final state mechanism
 no $\$$ but z_0 .
 $\{\epsilon, a\}$

15.



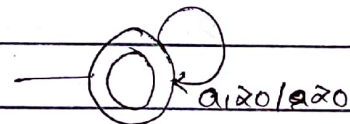
$L = a^*$

16.



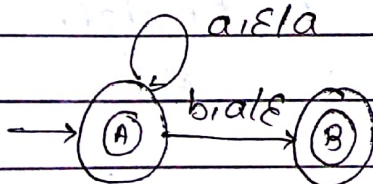
$L = a^+b$

17.



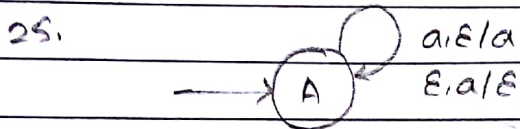
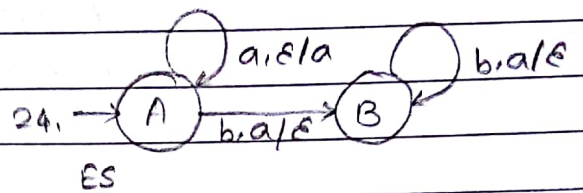
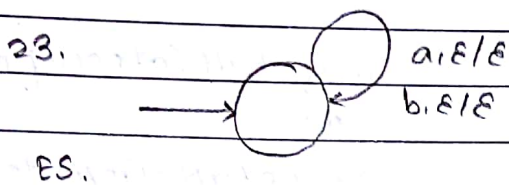
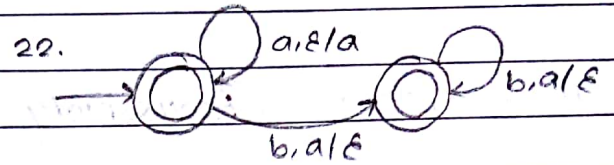
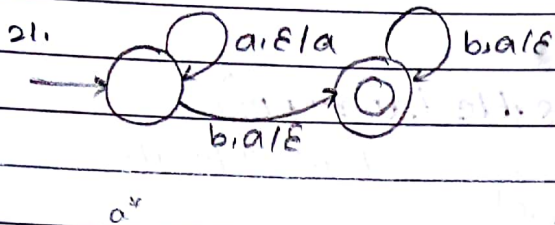
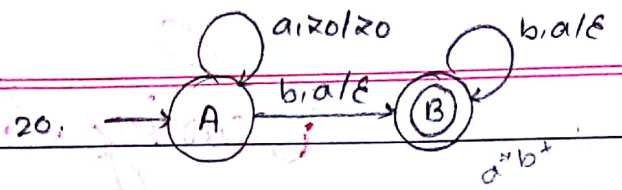
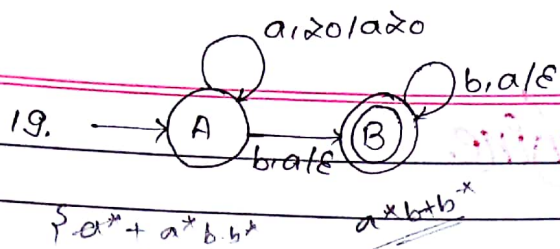
$L = \{\epsilon, a, aa, \dots\}$
 $= a^*$

18.



$A \rightarrow a^*, B \rightarrow a^+b$

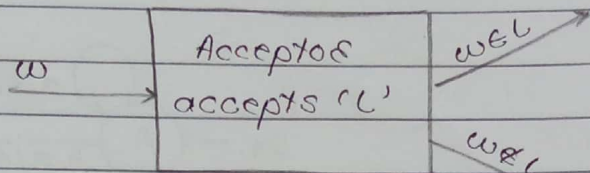
$\therefore a^* + a^+b$



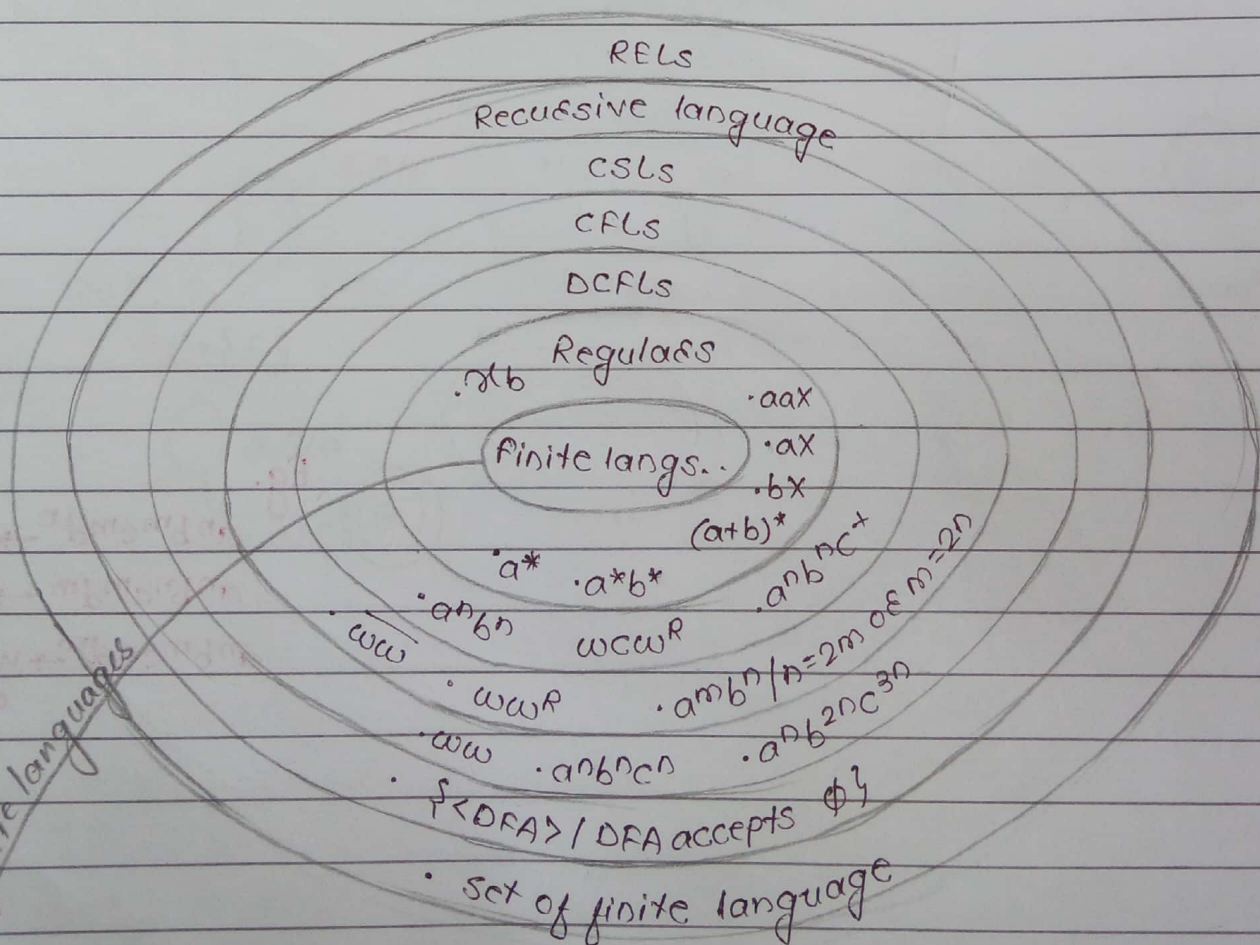
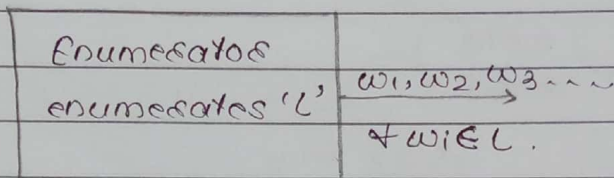
Eg.
 $a^n b^m c^m d^n \rightarrow CFL$
 $a^n b^n c^m d^m \rightarrow CFL$
 $a^n b^m c^n d^m \rightarrow \text{not } CFL$

Turing Machine

- It accepts REL. (Recursively Enumerable Language)



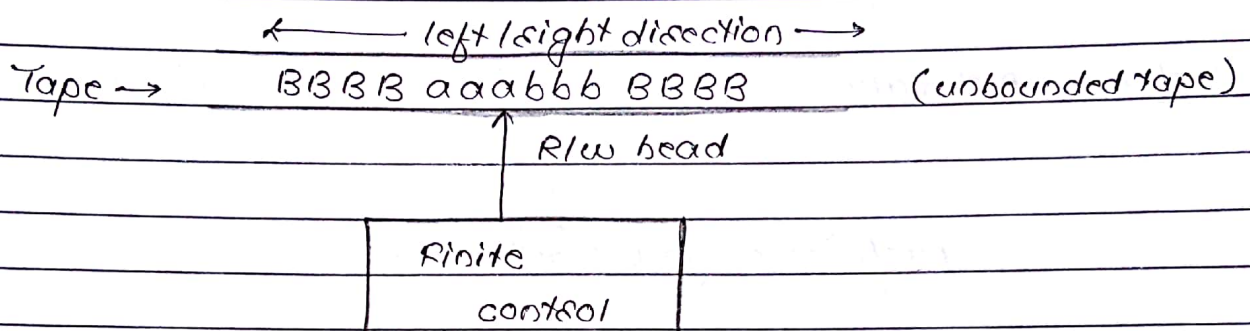
- It enumerates REL.



Configuration :

$$\text{PDA} + n\text{-stack} \underset{(\geq 1)}{\cong} \text{PDA} + 1\text{stack} \underset{(2\text{stack PDA})}{\cong} \text{TM} \underset{\text{stacks}}{\cong} \text{FA} + 2\text{stacks} \underset{\text{stacks}}{\cong} \text{FA} + n (\geq 2)$$

$\text{FA} + \text{R/W head} + \text{L/R direction} \cong \text{Turing machine}$



Linear Bounded Automata

It is an Turing machine but tape is restricted (bounded) to length of input.

$\rightarrow$ string
If $|w| = n$, then size of tape $= O(n)$.

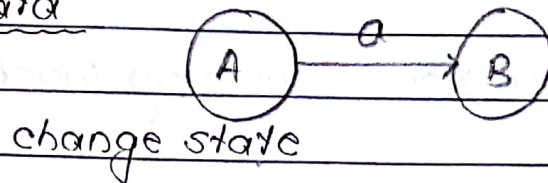
A Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ where Q, Σ, Γ are all finite sets and

1. Q is the set of states.
2. Σ is the input alphabet not containing the blank symbol (B),
3. Γ is the tape alphabet, where $B \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.
5. $q_0 \in Q$ is the start state.
6. $q_A \in Q$ is the accept state, and
7. $q_R \in Q$ is the reject state, where $q_A \neq q_R$.

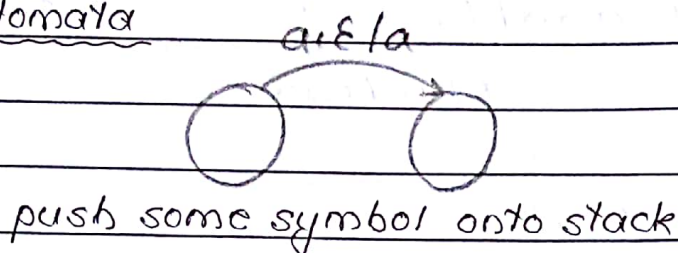
Non-deterministic Turing Machine

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$$

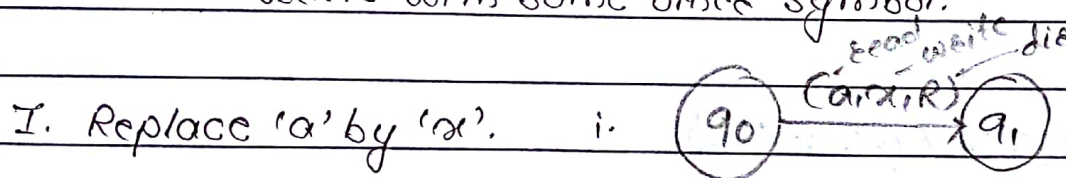
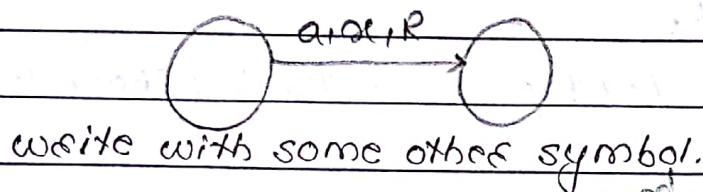
Finite Automata



Push Down Automata



Turing machine



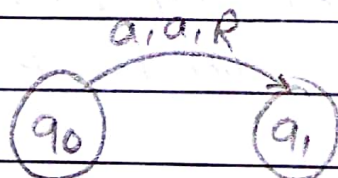
ii. $\delta(q_0, a) = (q_1, x, R)$

$$Q \times \Gamma = Q \times \Gamma \times \{L, R\}$$

iii.

δ	a
q_0	(q_1, x, R)

II. Skip 'a':



1. $a^n b^n$

I. Equal no. of a's & b's (a's followed by b's)

II. $a \rightarrow$ replaced by x $x: x$
 $b \rightarrow$ replaced by y $y: y$

III. $q_0: x x$ $q_3: \text{skip 'y' to reach B.}$
 $q_1: y y$
 $q_2: \text{reverse scan}$

IV. $\dots B a a a a b b b b B \dots$
 $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $x a a a y b b b$
 $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $x x a a y y b b$
 $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $x x x a y y y b$
 $\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow$
 $x x x x y y y y$

V. $\dots B \overset{x}{a} a a a \overset{y}{b} b b b B \dots$ 1st scan
 $\uparrow \quad \uparrow \quad \text{skip } a \quad \uparrow$
 $q_0 \rightarrow q_1 \rightarrow q_2$
 $\downarrow \quad \text{skip } a \quad \downarrow$
 $q_2 \rightarrow q_0$

$\dots B \overset{x}{x} \overset{y}{a} a a y \overset{y}{b} b b B$
 $\uparrow \quad \uparrow \quad \text{skip } a, y \quad \uparrow$
 $q_0 \rightarrow q_1 \rightarrow q_2$
 $\downarrow \quad \text{skip } y, a \quad \downarrow$
 $q_2 \rightarrow q_0$

$B \overset{y}{x} \overset{y}{x} \overset{y}{x} \overset{y}{x} \overset{y}{y} \overset{y}{y} \overset{y}{y} \overset{y}{y} B$ Last seen
 $\uparrow \quad \uparrow \quad \uparrow$
 $q_2 \leftarrow q_2 \leftarrow q_1$
 $\downarrow$
 q_0

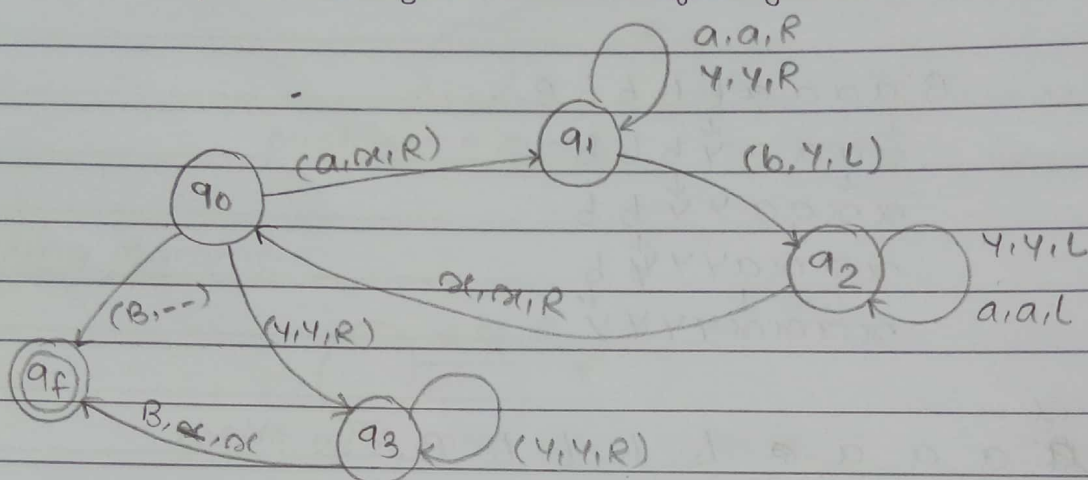
q₀:

- i. $\cancel{a}a$ - and move to q₁
- ii. when it sees 'y' move to q₃.

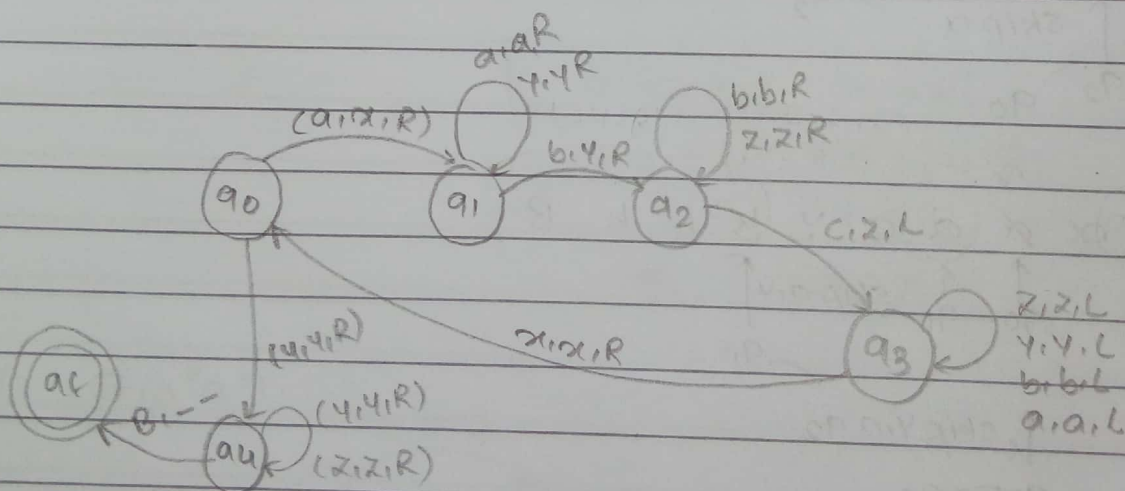
q₁:

- i. skip a, y
- ii. Replace 'b' with 'y' and move to q₂ in left direction.

q₂: skip y, a in left direction to find first 'a'.



2. $\{a^n b^n c^n \mid n \geq 1\}$



q₀: $\cancel{a}a$

q₁: $\cancel{b}y$

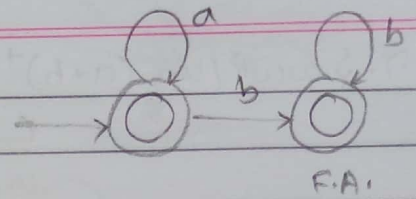
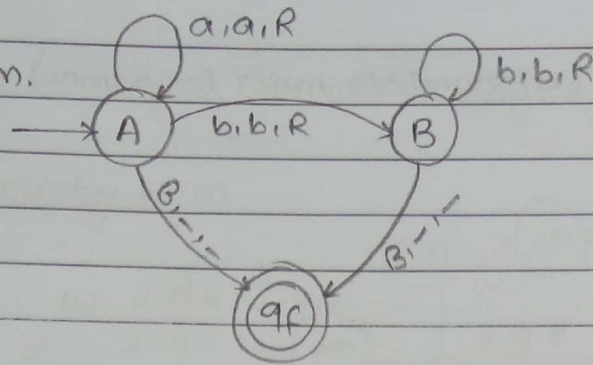
q₂: $\cancel{z}y$

q₃: reverse direction

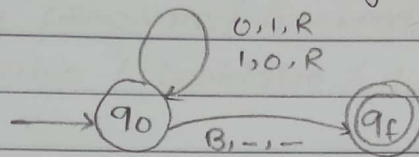
q₄: skip y & z to reach B.

3. a^*b^*

T.M.



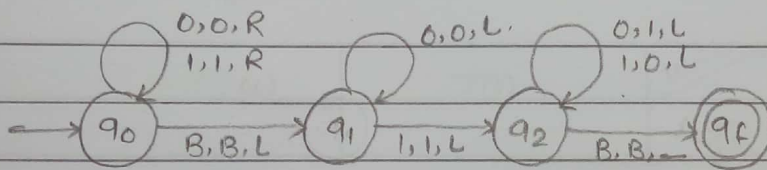
4. 1's complement of a binary number.



... B 1 0 1 0 0 B ...
0 1 0 1 1

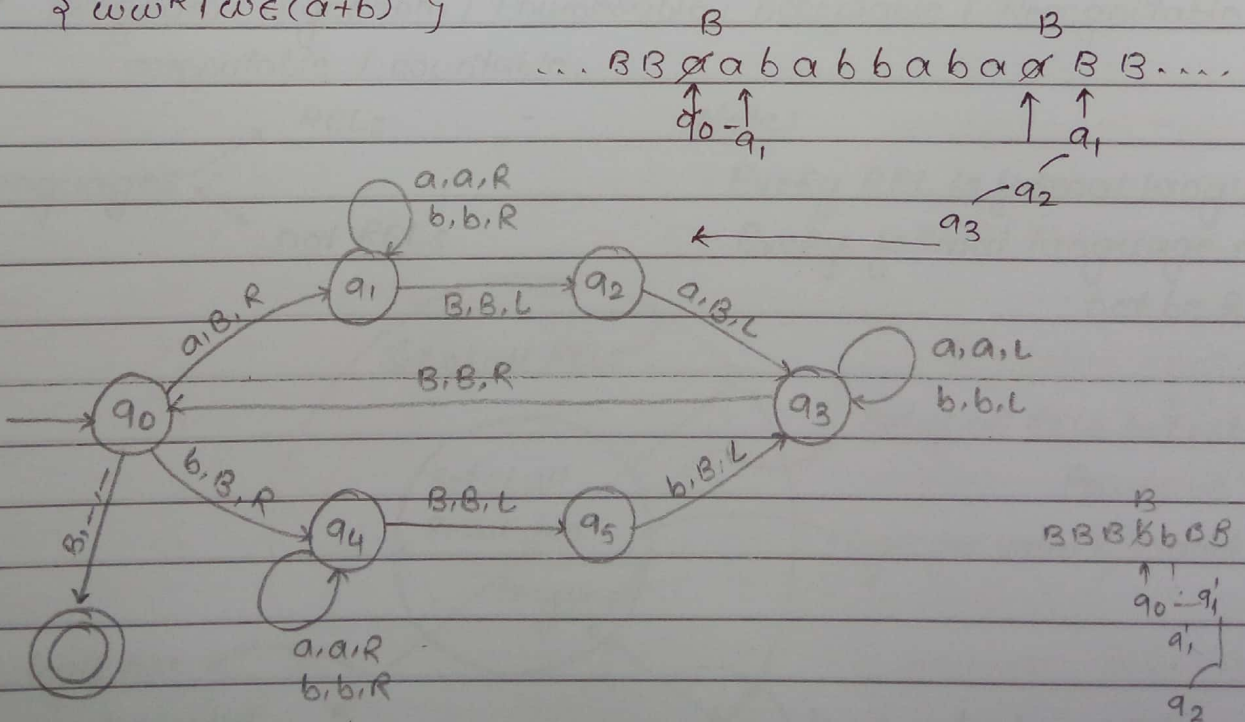
5. 2's complement of a binary number.

Given: 10100



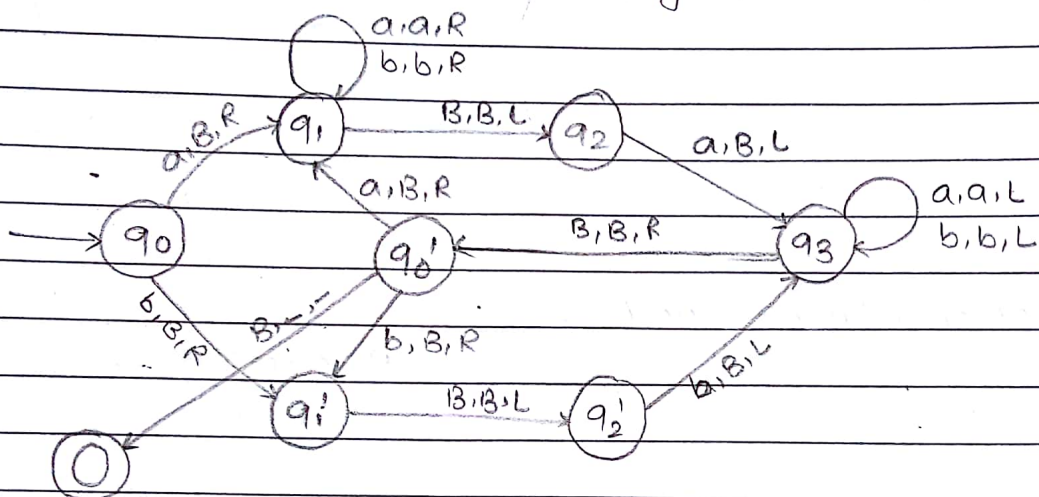
0 1 0 1 1 $\rightarrow$ 1's C.
0 1 1 0 0 $\rightarrow$ 2's C.
1's C Same

6. $\{ww^R | w \in (a+b)^*\}$

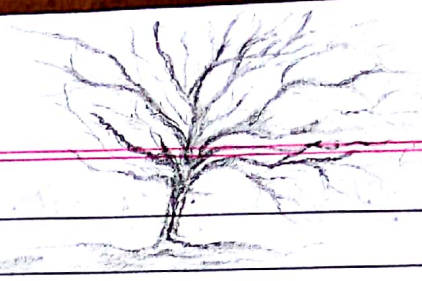


7. $\{w w^R \mid w \in (a+b)^+\}$

(1st and last symbols must be same)

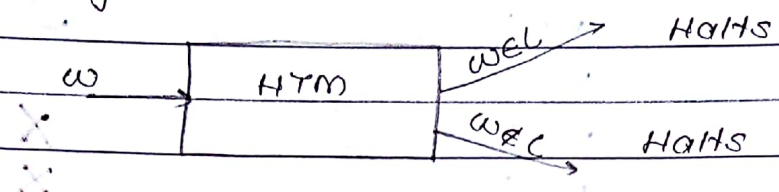


8. $\{w c w^R \mid w \in (a+b)^*\}$ / $w \in (a+b)^+$



Recursive language : (Decidable language)

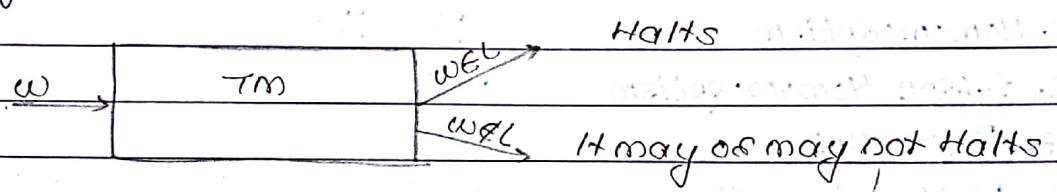
→ Accepted by HTM.



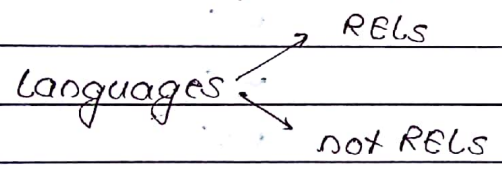
→ Halting program / Generated by algorithm / lexicographically enumerable / Decidable / Turing decidable / TM decidable.

Recursively Enumerable Language (REL)

→ Accepted by TM

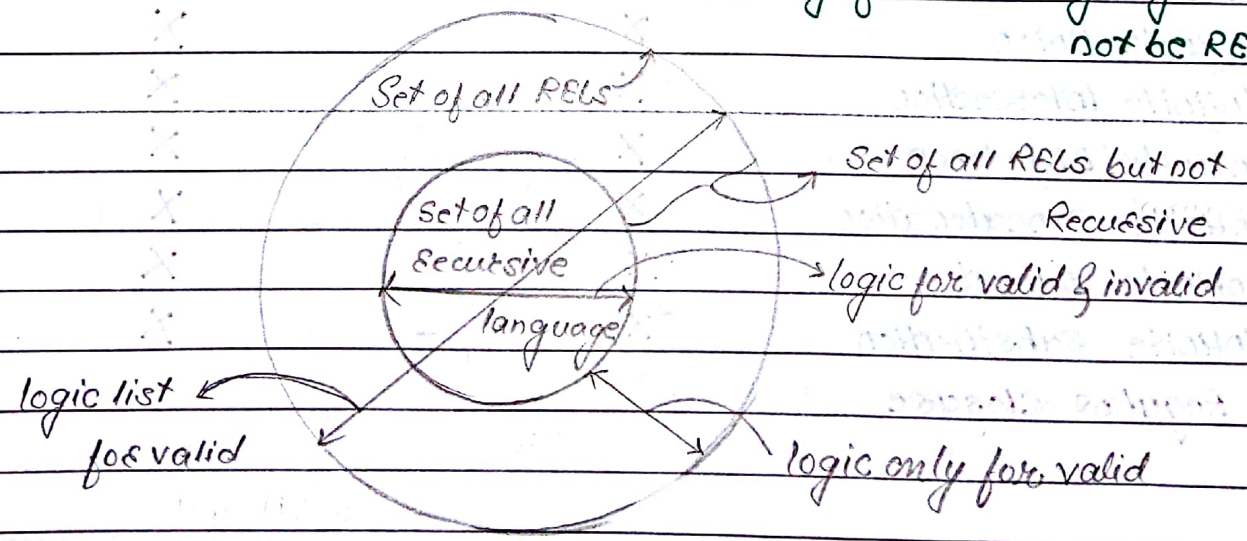


→ Program / Algorithm / Enumerable / Acceptable / Recognizable / computable / countable.



Note:

- Every REL is formal language.
- Every formal language need not be REL.



Closure properties of Recursive language and REL.

	<u>Recursive language</u>	<u>Recursively Enumerable language.</u>
1. Union	✓	✓
2. Intersection	✓	✓
3. Complement	✓	X
4. Difference	✓	X
5. Kleene closure	✓	✓
6. positive closure	✓	✓
7. Concatenation	✓	✓
8. Reversal	✓	✓
9. Subset	X X	X
10. Substitution	X X	✓
11. Homomorphism	X X	✓
12. E-free Homomorphism	✓	✓
13. Finite Union	✓	✓
14. Finite Intersection	✓	✓
15. Finite Difference	✓	✓
16. Finite Concatenation	✓	✓
17. Inverse Homomorphism	✓ ✓	✓
18. Finite subset	✓ ✓	✓
19. Finite Substitution	✓	✓
20. Infinite Union	X	X
21. Infinite Intersection	X	X
22. Infinite Difference	X	X
23. Infinite Concatenation	X	X
24. Infinite subset	X	X
25. Infinite Substitution	X	X
26. Regular Closure		

Recursive language (Not closed)

Substitution, Homomorphism

Recursively Enumerable language (Not closed)

Complement, Difference

Note:

1. $\subseteq$ for all not closed.
2. $\text{Infinite}(U, \cap, \dots)$ for all not closed.
3. Inverse Homomorphism for all closed.
4. $\subseteq$ finite for all closed.

Note:

$\subseteq$	$\infty(U, \cap, \dots)$	h^{-1}	$\subseteq_{fin}$
x	x	✓	✓

✓ $D: \bar{L}, \text{prefix}(L)$

x $C: \cap, \bar{L} \rightarrow L_1 - L_2, \text{fin } \cap, \text{fin diff}$

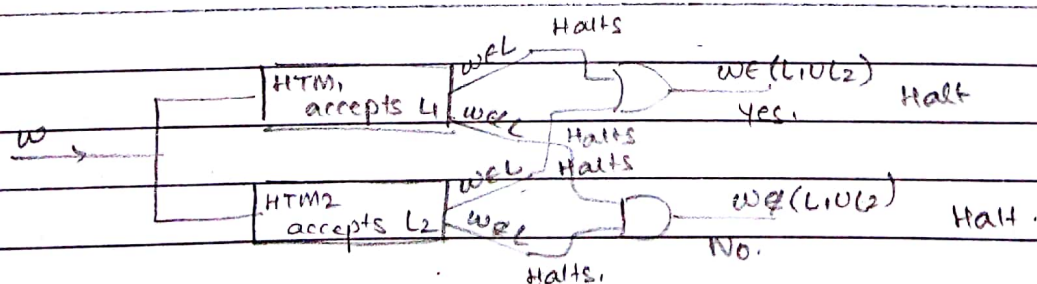
x $\text{Rec}: f(L) \rightarrow h(L)$

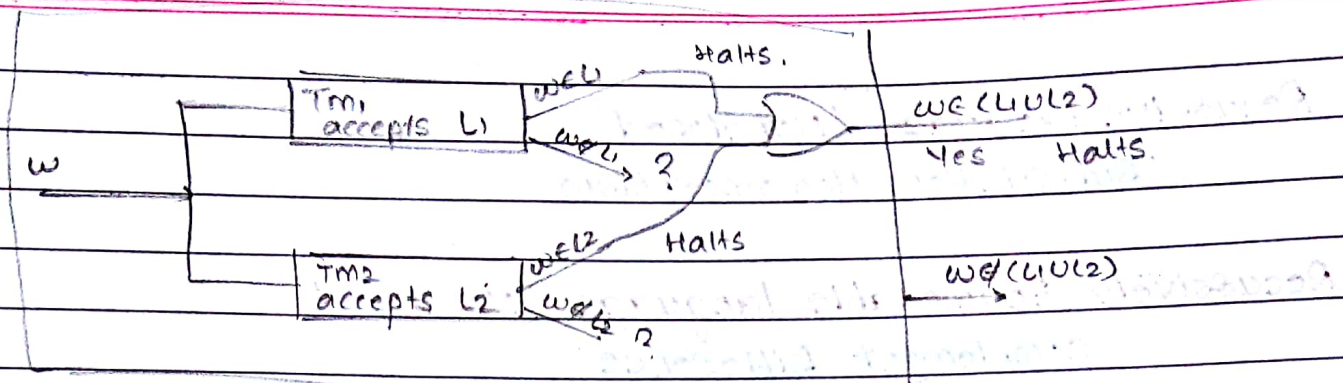
x $\text{REL}: \bar{L} \rightarrow L_1 - L_2$

Union

$\text{Rec}_1 \cup \text{Rec}_2 = \text{Recursive}$

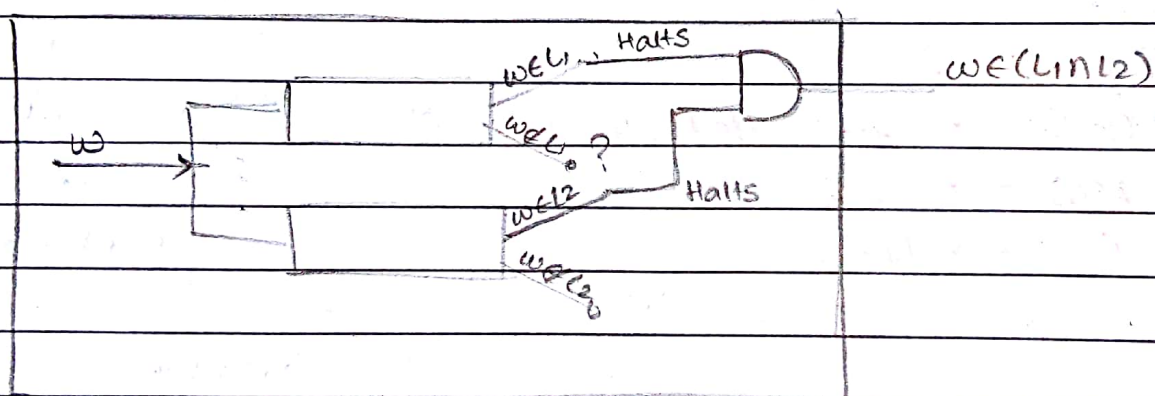
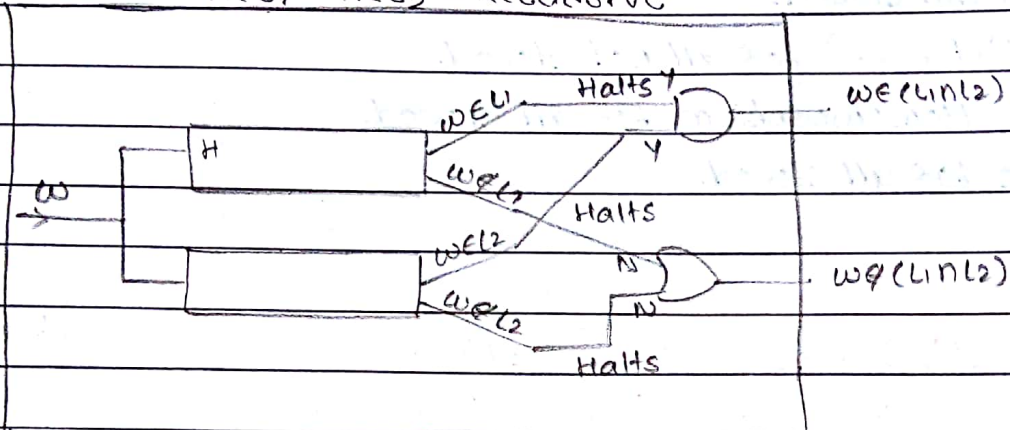
$\text{REL}_1 \cup \text{REL}_2 = \text{REL}$





Intersection

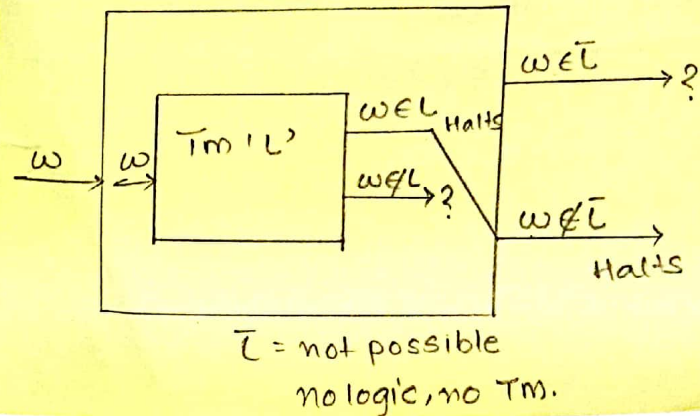
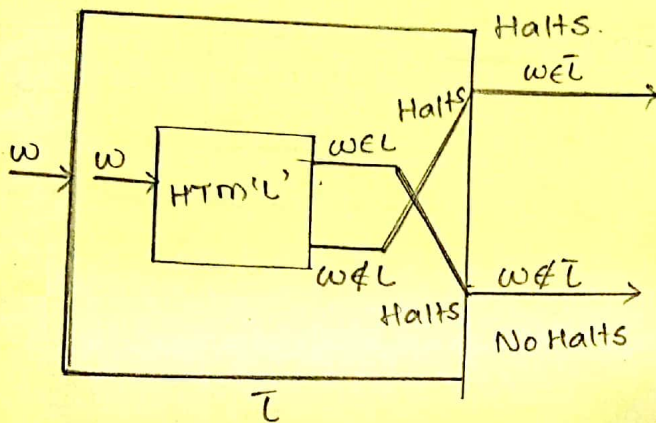
$Rec_1 \cap Rec_2 = Recursive$



Complement:

Recursive language = Recursive

Recursively enumerable language = Need not be REL

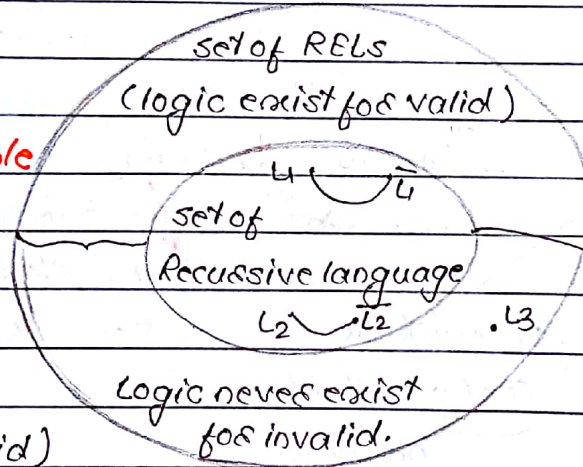


NOT RELs

D → decidable

SD → Semidecidable

Set of RELs but
not recursive
(Semidecidable)
(logic exist for valid)



$\overline{REL}$ = Need not be REL

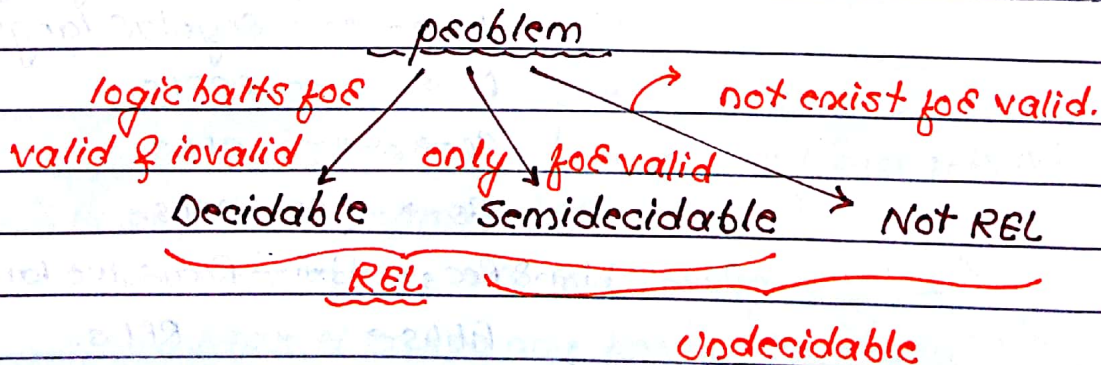
$(\overline{D \text{ or } SD}) = D \text{ or not REL}$

$\overline{REL}$ = never be

semidecidable

→ logic exist for valid
and invalid.

• $\bar{L}_3$ (logic never exist
valid of $\bar{L}_3$)



Every not REL → undecidable

Undecidable → not or need not be SD, REL.

Types of Turing Machine :

Turing machine $\cong$ single tape Turing machine

$\cong$ one way tape Turing machine
(unbounded)

$\cong$ multi-tape Turing machine

$\cong$ multi-head Turing machine

$\cong$ multi-tape & multi-head Turing machine

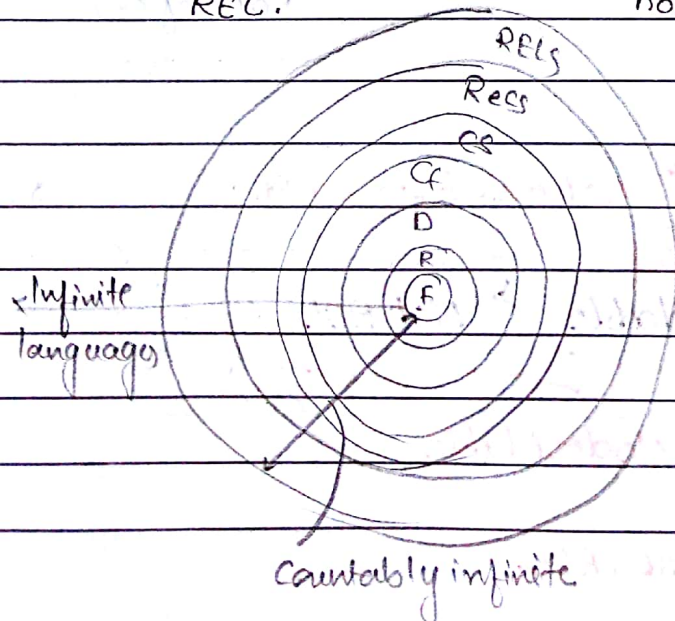
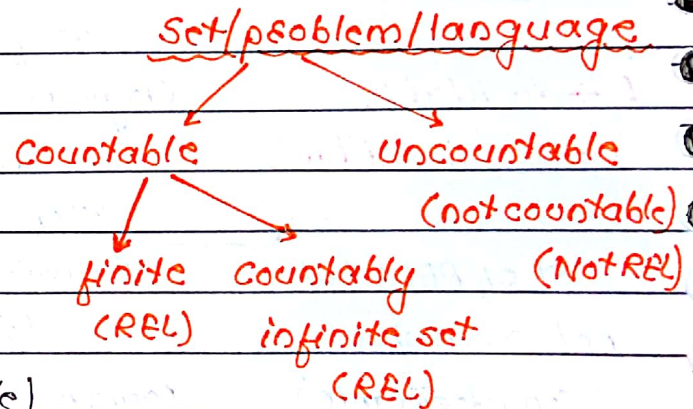
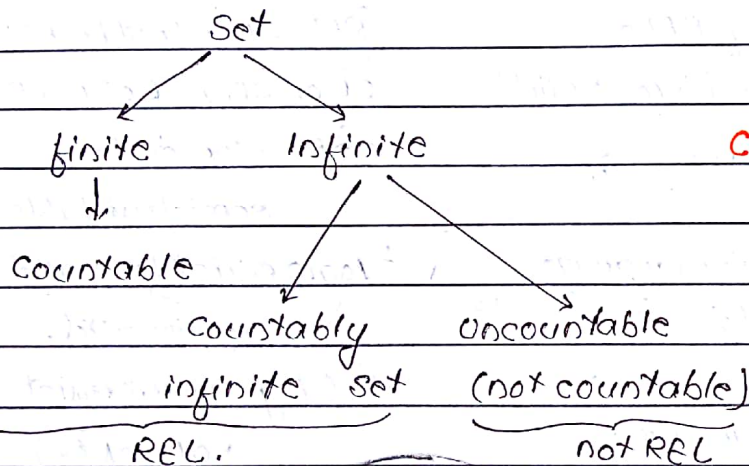
$\cong$ Multi-dimensional tape Turing machine

$\cong$ two stack PDA

$\cong$ Counter machine

$\cong$ Universal Turing machine (programmable)

Countable and Uncountable



$F \rightarrow$ set of finite language

$R \rightarrow$ — " — regular language

$D \rightarrow$ — " — DCFLs.

$C_f \rightarrow$ — " — CFLs.

$C_s \rightarrow$ — " — CSLs.

$Rec_s \rightarrow$ — " — Recursive language

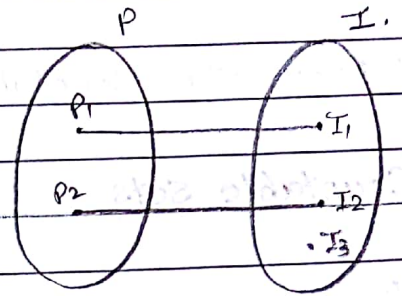
$REL_s \rightarrow$ — " — RELs.

Countably Infinite Set:

Injective function

Every two elements of $P(a, b)$, should have different images in I .

$$\left. \begin{array}{l} \forall a, b \in P, \\ a \neq b \end{array} \right\} \begin{array}{l} f(a) = I_1 \\ f(b) = I_2 \end{array} \quad I_1 \neq I_2$$



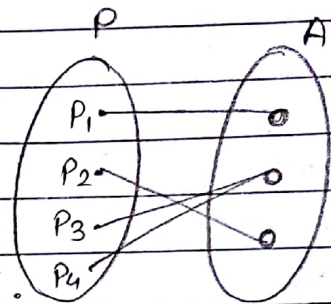
Surjective function

$$\forall a \in A, \exists p \in P$$

$$f(p) = a$$

pre-image

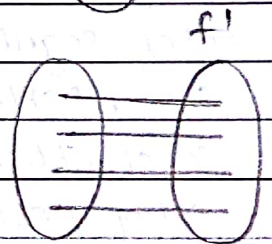
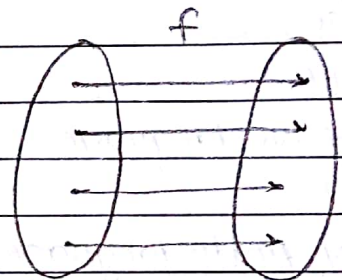
Every element of codomain has at least one pre-image.



Bijjective function (one to one correspondence) (Invertible)

$$f: A \rightarrow B \quad f: B \rightarrow A$$

- f is injective and surjective.
- f and f^{-1} are functions.
- f is a function, every codomain element has exactly one pre-image.



Set of Natural numbers (countably infinite set)

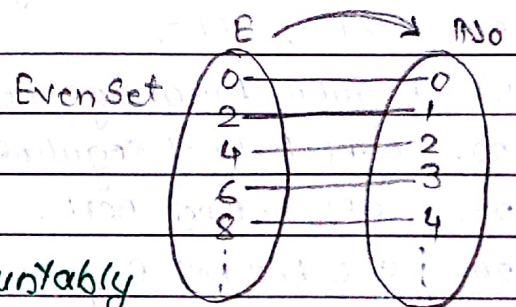
$$L = \{0, 1, 2, \dots\} \rightarrow \mathbb{N}_0$$

$$= \{1, 2, \dots\} \rightarrow \mathbb{N}$$

X is countably infinite set iff

$$f: X \rightarrow \mathbb{N} \text{ is bijective}$$

where $\mathbb{N}$ can be any known countably infinite set.



$$f(x) = x/2$$

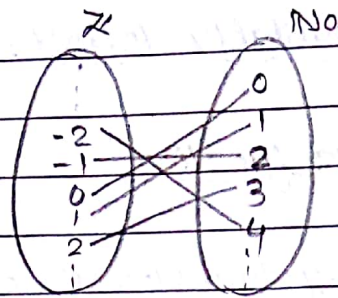
$$x \in E, f'(x) = 2x \\ x \in \mathbb{N}_0$$

$$x \in \mathbb{Z} \quad f(x) = 0 ; x = 0$$

$$= 2x - 1 ; x = +ve$$

$$= -2x ; x = -ve$$

"Even Set is Countable".



Countable Sets (REL)(TM)

1. N
2. $\mathbb{Z}$
3. Even set
4. Odd set
5. prime set
6. Rational set ($\mathbb{Q}$)
7. Finite set
8. Regular set
9. CFL
10. CSL
11. DCFL
12. Recursive language
13. REL
14. Set of finite languages
15. Set of regular languages
16. Set of DCFLs.
17. Set of CFLs.
18. Set of CSLs.
19. Set of Recursive language.
20. Set of RELs.
21. Regular language but not finite.
22. DCFL but not regular
23. CFL but not DCFL.
24. CFL but not Regular
25. Complement of regular language.
26. Complement of CFL.
27. Complement of DCFL.
28. Complement of recursive language.
29. Complement of finite language.
30. Union of two countable sets.
31. Intersecting 2 countable sets.
32. Every subsets of countable set.
finite over $\{a, b\}$
33. Cartesian product of 2 countable sets.
34. Every language over $\Sigma = \{a, b\}$
35. Language L is subset of Σ^*
where $|\Sigma|$ is finite.

Uncountable sets (Not RELs) (no logic)

01. Set of real numbers.
02. Set of irrational numbers.
03. Set of not natural numbers.
04. Set of infinite language.
05. Infinite language
06. Formal language
07. Set of formal language.
08. Not regular language.
09. Not DCFL
10. Not CFL
11. Not REL
12. Not CSL
13. Set of languages = 2^{Σ^*}
14. Set of not regulars = { DCFLs but not regulars, CFLs but not regulars, CSLs but not regulars, RELs but not regulars }
15. Set of not DCFLs.
16. Set of not CFLs.
17. Set of not CSLs.
18. Set of not RELs.
19. { L1L! DCFL but not regular }

Equivalence

• Recursive language $\cong$ Decidable $\cong$ Lexicographically Enumerable

$\cong$ Total function

$\cong$ REL but not undecidable

$\cong$ REL but not semidecidable

$\cong$ Countable but not semidecidable

$\cong$ Computable but not semidecidable

• REL but not recursive $\cong$ semidecidable

$\cong$ partial decidable $\cong$ REL but not decidable

$\cong$ partial function but not total

$\cong$ countable but not decidable

$\cong$ undecidable but REL

$\cong$ undecidable but partial decidable.

• Not Recursively Enumerable language $\cong$ Uncountable

$\cong$ undecidable but not semidecidable

$\cong$ undecidable but not REL.

$\cong$ no function $\cong$ not computable

$\cong$ not Enumerable

$\cong$ not acceptable

• Recursively Enumerable language $\cong$ countable

$\cong$ computable

$\cong$ partial function

$\cong$ enumerable

$\cong$ Turing recognizable

$\cong$ either decidable or semidecidable.

- Undecidable $\cong$ not decidable $\cong$ either semidecidable or not REL
 $\cong$ not recursive, $\cong$ not total function

Recursive language:

1. $L \rightarrow$ unknown

If $\bar{L}$ is decidable, then L is decidable.

2. If L is decidable then $\bar{L}$ is decidable.

REL but not Recursive:

* If L is semidecidable, then $\bar{L}$ is not REL.

If $\bar{L}$ is semidecidable then L is not REL.

Not Recursively Enumerable Language:

• If L is not REL, then $\bar{L}$ is semidecidable or not REL. undecidable

• If $\bar{L}$ is not REL, then L is semidecidable or not REL. undecidable

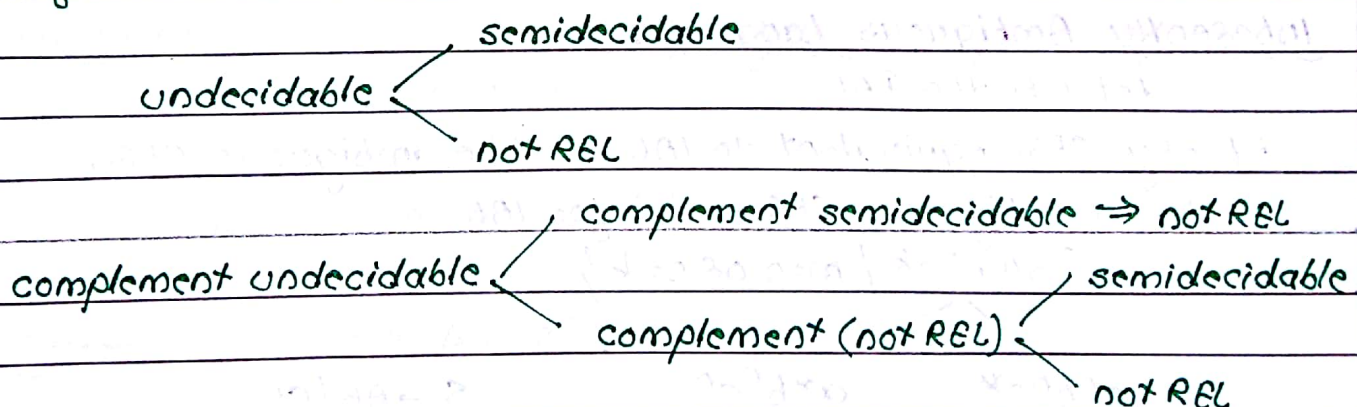
Recursively Enumerable Language:

• If L is REL, then $\bar{L}$ is decidable or not REL.

• If $\bar{L}$ is REL, then L is decidable or not REL.

Undecidable:

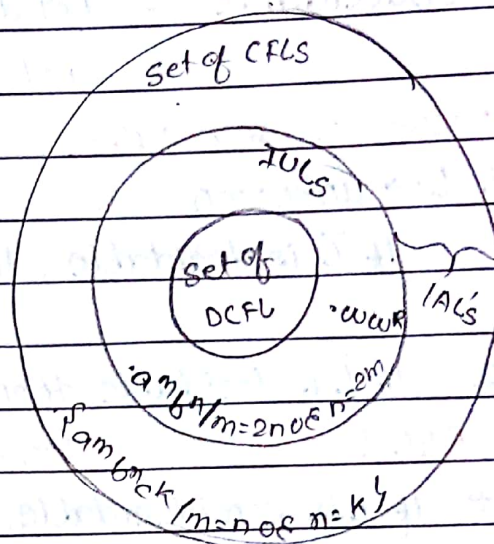
• If L is undecidable then $\bar{L}$ is undecidable.



• If $\bar{L}$ is undecidable then L is undecidable.

Inherently Unambiguous Language

If there exist unambiguous CFGs for any language, then it is Inherently Unambiguous language (IUL).



Find language generated by CFG?

$S \rightarrow A1a$ $L = \{a^n\} \rightarrow \text{IUL}$

$A \rightarrow a$ L produced by given CFG has some unambiguous CFG, so, L is IUL.

Eg. $\{a^m b^n / m=2n \text{ or } n=2m\} \rightarrow (\text{CFL but not DCFL})$

$a^{2n} b^n$

$a^n b^{2n}$

$A \rightarrow aaAb / \epsilon$

$B \rightarrow aBbb / \epsilon$

$S \rightarrow A/B$

$A \rightarrow aaAb / \epsilon$

$B \rightarrow aBbb / \epsilon$

$\therefore$ unambiguous CFG.

• ww^R , $S \rightarrow asa / bsbl / \epsilon$ $\rightarrow$ Unambiguous CFG.

Inherently Ambiguous Language

Let L be the IAL

• Every CFG equivalent to IAL must be ambiguous CFG.

• No unambiguous CFG exist for IAL

$\{a^m b^n c^k / m=n \text{ or } n=k\}$

$a^n b^n c^k$

$a^k b^n c^n$

$S \rightarrow AB / CD$

$A \rightarrow aAb / \epsilon$

$B \rightarrow cB / \epsilon$

$C \rightarrow aC / \epsilon$

$D \rightarrow bDc / \epsilon$

Note: Every CFG is Ambiguous

Regular closure

	$Reg_{sc(L)}$	$DCFL_{sc(L)}$	$CFL_{sc(L)}$	$CSL_{sc(L)}$	$Rec_{sc(L)}$	$REL_{sc(L)}$
$L \cup Reg$	Reg	DCFL	CFL	CSL	Rec	REL
$L \cap Reg$	Reg	DCFL	CFL	CSL	Rec	REL
$L - Reg$	Reg	DCFL	CFL	CSL	Rec	REL
$Reg \cup L$	Reg	DCFL	CFL	CSL	Rec	REL
$Reg \cap L$	Reg	DCFL	CFL	CSL	Rec	REL
$Reg - L$	Reg	DCFL	?	CSL	Rec	?

$$Reg - DCFL = Reg \cap \overline{DCFL}$$

$$= Reg \cap DCFL$$

$$= DCFL.$$

$\cup$

$$X \rightarrow \cap Reg = X.$$

$-$

Simplified CFG:

If CFG₁ has i. No unit ii. No Null iii. No useless production then it is simplified CFG₁.

I. Reduced CFG:

If CFG₁ has no useless productions then it is reduced.

II. Simplification of CFG: (CFG₁ $\Rightarrow$ simplified CFG₁)

Step 01: Delete Null production

Step 02: Delete unit production

Step 03: Delete useless production.

III. Deleting Null production $a(x \rightarrow \epsilon)$

1. $S \rightarrow a(\epsilon)$

$\Downarrow$ delete $S \rightarrow \epsilon$

$S \rightarrow a$ but empty string not possible in result, so that it should be noted.

"If many NULL productions, delete one by one."

2. $S \rightarrow Aa|Ab$ $\Rightarrow$ delete ϵ $S \rightarrow Aa|Ab|a|b$
 $A \rightarrow b|\epsilon$ $A \rightarrow b$

"put ϵ in place of A to produce new productions."

Note: If given CFG is not generating ' ϵ ' then after deleting Null productions, the resultant CFG is equivalent to given CFG.

IV. Deleting UNIT productions: ($X \rightarrow Y$ is unit production)

4. $S \rightarrow A|Ab$ $\Rightarrow$ delete $S \rightarrow A$ $S \rightarrow Ab|alsb$
 $A \rightarrow alsb$ $A \rightarrow alsb$

place all A productions in RHS side of $S \rightarrow A$.

"If many UNIT productions, delete one by one."

V. Deleting Useless productions

Step 1: Delete the productions which cannot derive any string.
 $(C \rightarrow Cd)$

Step 2: Delete the productions which are not reachable from start symbol.

Given:

$S \rightarrow aS|AB$

$A \rightarrow ac|aEb|g$

$B \rightarrow b$

$C \rightarrow dC$

$F \rightarrow e$

Step 1: Find productions which cannot derive any string.

'C' cannot derive ϵ 'E' also.

So delete all productions where C & E are present.

$S \rightarrow aS|AB$, $A \rightarrow g$, $B \rightarrow b$, $F \rightarrow e$

Step 2: Delete from productions which are not reachable.

Reachable set = $\{S, A, B\}$

So, F is not reachable.

$S \rightarrow aS \mid AB, A \rightarrow g, B \rightarrow b$

Null productions: $\alpha \rightarrow \epsilon \quad \alpha() \{ \}$

UNIT productions: $\alpha \rightarrow \gamma \quad \alpha() \{ \gamma() \}$

USELESS productions: $\alpha \rightarrow a\alpha \quad \gamma \rightarrow ab$

α never ends

γ is not reachable from S.

Examples:

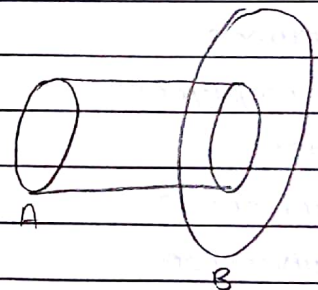
01. $S \rightarrow aA \Rightarrow$ simplified CFG = $\{ \}$ $L = \emptyset$

02. $S \rightarrow aA \mid bB \Rightarrow$ simplified CFG = $S \rightarrow aA, A \rightarrow a$
 $A \rightarrow a \quad \therefore L = \{a^n\}$

Reducibility

$A \leq B$

A is reducible to B.



B is at least as hard as A.

(more) (equal)

$A \leq_p B$ A is (polynomially) reducible to B.

$A \leq_m B$ A is many to one reducible to B.

$A \leq B$

1. If B is decidable, A is also decidable.

$A \leq B$

2. If A is undecidable, B is undecidable.

$A \leq B$

3. If B is finite, A is finite.
4. If A is infinite, B is infinite.
5. If B is REL, A is REL (fin/seg/OCFL/CFL/CSL/dec/REL).
6. If A is not REL, B is not REL.
7. If A is $O(n^3)$, then B is $\Omega(n^3)$. (min) $\rightarrow$ lower bound.
 $B \geq A, B \geq n^3, B = \Omega(n^3)$
8. If B is $O(n^3)$, then A is $O(n^3)$. (max) $\rightarrow$ upper bound.
 $B \geq A, A \leq B, A \leq O(n^3), A = O(n^3)$
9. If B is total function, A is total function.
10. If B has Turing machine, then A also has Turing machine.
11. If B is Regular then A is also regular (may finite/infinite).
12. If $\bar{B}$ is decidable (B is decidable) then A is decidable.
13. If $\bar{A}$ is undecidable (A is undecidable) then B is undecidable.

Decidable and Undecidable problem

• membership problem

Emptiness

Non-emptiness

Finiteness

Infiniteness

Equivalence

Totality

Containment

Ambiguity

Halting

disjoint.

"

Finite automata (seg lang) (seg gsa)
(seg.exp)

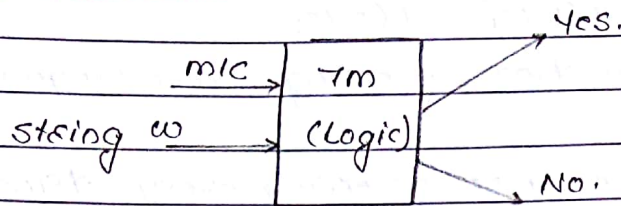
DPDA (OCFL)

PDA (CFL) (NCFL) (NPDA) (CFG)

HTM (recursion) (decidable lang)

TM (REL) "

Membership problem



Membership for machine

Is w accepted by machine?

Is $w \in L(m/c)$?

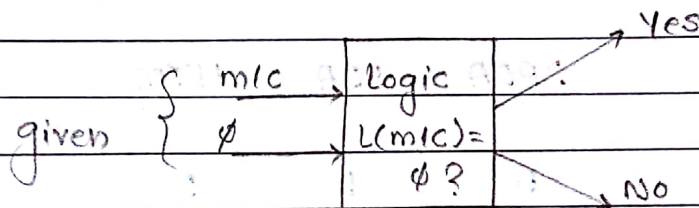
Is $w \in L(\text{grammar})$?

Is $w \in L$?

membership for corresponding grammar language

} Same

Emptiness problem



Is given machine accepts empty language?

Is $L(m/c) = \emptyset$?

Is $L(\text{grammar}) = \emptyset$?

Is given machine rejecting every string?

Non-Emptiness problem

Is $L(m/c) \neq \emptyset$?

Is given machine accepting atleast one string?

Finiteness problem:

Is $L(m/c)$ finite?

Infinite^{ness} problem:

Is $L(m/c)$ infinite?

Equivalence problem

$$\text{Is } L(M_1) = L(M_2)$$

Is given two machines accepting same language?

Totality problem

Is given machine accepting every string?

Containment problem (subset checking)

$$\text{Is } L(M_1) \subseteq L(M_2)?$$

$$L_1 \subseteq L_2 \text{ iff}$$

$$1. L_1 \cap \bar{L}_2 = \emptyset \quad 2. L_1 - L_2 = \emptyset$$

Ambiguity problem

Is grammar ambiguous?

Halting problem

Is given machine halts?

Disjoint problem

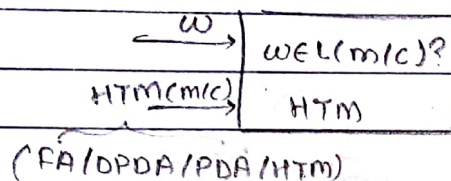
$$\text{Is } L(M_1) \cap L(M_2) = \emptyset$$

<u>Problems</u>	FA	DPDA	PDA	HTM	TM
• Membership	↑	D	D	D	↑
• Emptiness		D	D		
• Non-Emptiness		D	D		
• Finiteness		D	D		undecidable
• Infiniteness		D	D		(HTM not
• Equivalence		D	≠		exist)
• Totality		D			
• Containment					
• Ambiguity					
• Halting		D	D	D	
• Disjoint					↓

Decidable
(HTM exist)

Membership $\Rightarrow$ CYK algorithm, (FA, DPDA, PDA) passing, Bottom-up passing, $O(n^3)$, dynamic programming

Algo. 1 $\Rightarrow$ FA } $\Rightarrow$ CFG $\Rightarrow$ Chomsky Normal Form (CNF) CFG $\Rightarrow$ DPDA } Dynamic programming

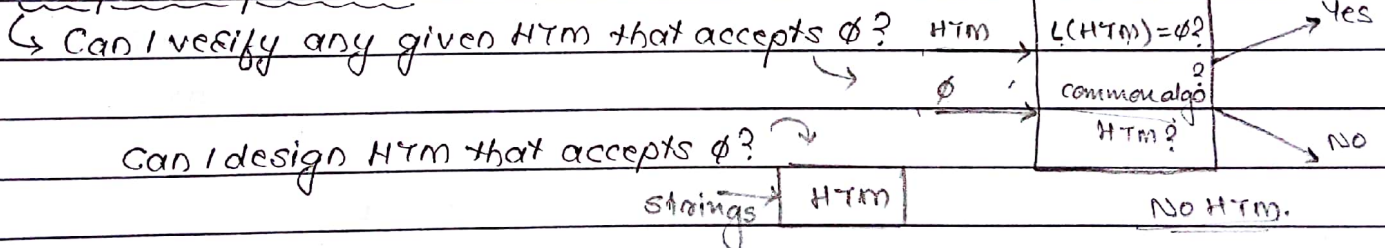


Is $prog1 \equiv prog2$?

Is given two programs logically equivalent?

Equivalence problem of TM.

Emptiness problem



Emptiness / Non-emptiness

FA } $\Rightarrow$ CFG $\Rightarrow$ Simplify

DPDA

PDA

At least 1 product $L \neq \emptyset$

No product $L = \emptyset$

Finiteness / Infiniteness (dependency graph)

FA } $\Rightarrow$ CFG $\Rightarrow$ dependency graph $\Rightarrow$ check loops

DPDA

PDA

Equivalence

Is $FA_1 \cong FA_2$?

$FA_1 \Rightarrow \text{min DFA}_1$

$FA_2 \Rightarrow \text{min DFA}_2$

} Isomorphic (same)

Containment

Is $R_1 \subseteq R_2$?

DPDA/PDA

HYM

$R_1 \cap \bar{R}_2 = \emptyset$

Is $D_1 \subseteq D_2$

Is $Rec_1 \subseteq Rec_2$

$D_1 \cap D_2 = \emptyset$?

$Rec_1 \cap Rec_2 = \emptyset$?

Is $HYM = \emptyset$?

Is $(HYM) = \emptyset$?

Finite automata $\Rightarrow$ all decidable

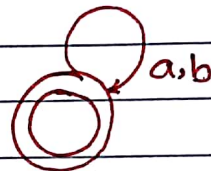
Turing machine $\Rightarrow$ all decidable

FA, PDA, DPDA (decidable)

(mem, $\emptyset$, not $\emptyset$, fin, Inf)

FA, DPDA (decidable)

(totality, equivalence)



FA, DPDA, PDA, HYM (decidable)

mem, halting

Decidable, Semidecidable & REL (Not REL)

Decidable, Semidecidable, Not Recursively Enumerable Language

Decidable:

L is decidable iff

- L has logic and $\bar{L}$ has logic.
- L is REL and $\bar{L}$ is REL.
- L has TM and $\bar{L}$ has TM.

Eg. L is regular language
 L is decidable

Semidecidable:

L is semidecidable iff

- L has logic and $\bar{L}$ has no logic.
- L is REL and $\bar{L}$ is not REL.
- L has TM and $\bar{L}$ has no TM.

Eg. $L =$ set of all regular languages
 L is semidecidable.

Not REL

L is not REL iff

- L has no logic.
- L is not REL.
- L has no TM.

Eg. L is the set of not regular languages
 $\downarrow$ no logic
 L is not REL.

Undecidable $\rightarrow$ Semidecidable, Not REL.

01. Finite language $\leftarrow L_1$ Decidable ($L_1 \rightarrow$ finite $\rightarrow$ TM, $\bar{L}_1 \rightarrow$ TM, $L_1 \rightarrow$ HTM)

02. Language over $\Sigma = \{a, b\}$ (finite alphabet) $\leftarrow L_2$
 $L \subseteq \Sigma^*$ over $\Sigma = \{a, b\}$ Decidable

03. Language over $(\Sigma = \{a, b, c, \dots\})$ finite alphabet $\leftarrow L_3$ Not REL
 $\Sigma^* = \{\epsilon, a, b, aa, \dots\}$

04. Σ^* over $\Sigma = \{a, b\} \leftarrow L_4$ - Reg Decidable
 $\bar{L}_4$ - Reg. $\exists$ HTM

05. Σ^* over $\Sigma = \{a, b, c, \dots\} \leftarrow L_5$ Not REL

06. Regular language $\rightarrow L_6$ Decidable
 07. CFL $\rightarrow L_7$ Decidable
 08. CSL $\rightarrow L_8$ Decidable
 09. Recursive language $\rightarrow L_9$ Decidable

trivial,

10. REL $\rightarrow L_{10}$ $L_{10} \rightarrow \text{TM}$ Semidecidable
 $\overline{L_{10}} \rightarrow \text{no TM.}$

$$\overline{\text{REL}} = \underbrace{\text{no logic}}_{\text{no logic}} \text{ or } \underbrace{\text{not REL.}}_{\text{no logic}}$$

11. Set of all regular languages. $\rightarrow L_{11}$ $\rightarrow$ Semidecidable
 $L_{11} \rightarrow \text{logic}$, $\overline{L_{11}} \rightarrow \text{no logic}$
12. Set of all CFLs. $\rightarrow L_{12}$ $\rightarrow$ Semidecidable
13. Set of all CSLs. $\rightarrow L_{13}$ $\rightarrow$ Semidecidable
14. Set of all DCFLs. $\rightarrow L_{14}$ $\rightarrow$ Semidecidable
15. Set of all recursive languages. $\rightarrow L_{15}$ $\rightarrow$ Semidecidable
16. Set of all RELs. $\rightarrow L_{16}$ $\overline{L_{16}} \rightarrow \text{no logic}$ $\rightarrow$ Semidecidable
17. Set of all DCFLs but not regulars. $\rightarrow L_{17}$ $\rightarrow$ Semidecidable
 $\overline{L_{17}} \rightarrow \text{all regulars and all not DCFLs}$
 no logic
18. Set of CFLs but not DCFLs. $\rightarrow L_{18}$ $\rightarrow$ Semidecidable
19. $\{ \langle \text{DFA} \rangle \mid L(\text{DFA}) = \emptyset \}$ $\rightarrow$ decidable

20. $\{ \langle \text{NFA} \rangle \mid L(\text{NFA}) = \emptyset \}$ $\Rightarrow$ decidable

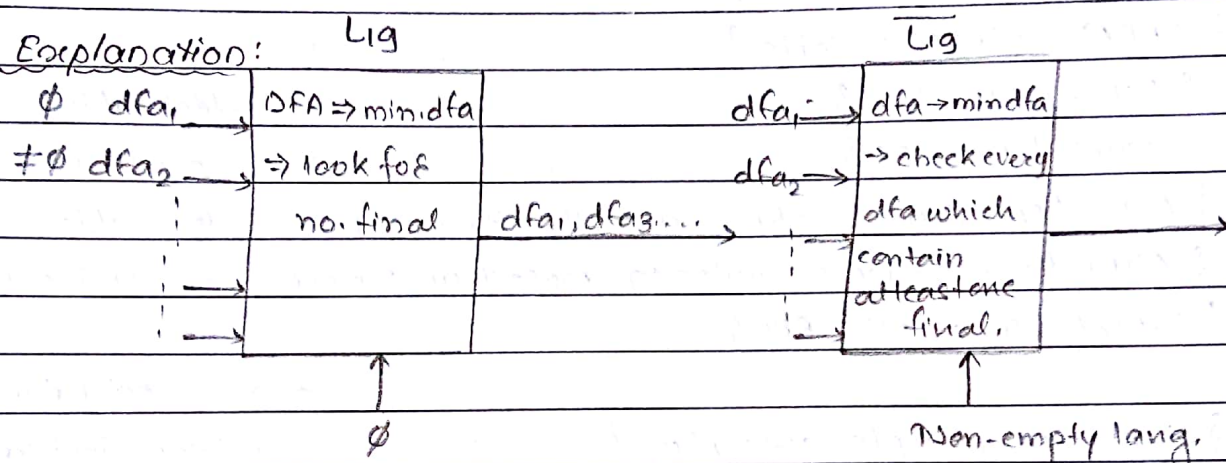
21. $\{ \langle \text{Regular Expression} \rangle \mid L(\text{Regular Expression}) = \emptyset \}$ $\rightarrow$ decidable

22. $\{ \langle \text{LLGI} \rangle \mid L(\text{LLGI}) = \emptyset \}$ $\rightarrow$ decidable

23. $\{ \langle \text{RLGI} \rangle \mid L(\text{RLGI}) = \emptyset \}$ $\rightarrow$ decidable

24. $\{ \langle \text{Regular Grammar} \rangle \mid L(\text{Regular Grammar}) = \emptyset \}$ $\rightarrow$ decidable

19 Explanation:



25. $\{ \langle \text{DFA} \rangle \mid L(\text{DFA}) \neq \emptyset \}$ $\rightarrow$ decidable

26. $\{ \langle \text{DFA}_1, \text{DFA}_2 \rangle \mid L(\text{DFA}_1) = L(\text{DFA}_2) \}$ $\rightarrow$ decidable

Equivalence

27. $\{ \langle \text{DFA} \rangle \mid \langle \text{DFA} \rangle = \text{finite} \}$ finiteness $\rightarrow$ decidable

28. $\{ \langle \text{DFA} \rangle \mid L(\text{DFA}) = \Sigma^* \}$ totality $\rightarrow$ decidable

29. $\{ \langle \text{DFA}_1, \text{DFA}_2 \rangle \mid L(\text{DFA}_1) \subseteq L(\text{DFA}_2) \}$ $\rightarrow$ decidable

containment

30. $\{ \langle \text{DFA}_1, \text{DFA}_2 \rangle \mid L(\text{DFA}_1) \cap L(\text{DFA}_2) = \emptyset \}$ disjoint $\rightarrow$ decidable

31. $\{ \langle \text{PDA} \rangle \mid L(\text{PDA}) = \emptyset \}$ $\rightarrow$ decidable

32. $\{ \langle \text{DFA} \rangle \mid \text{DFA accepts string 'w'} \}$ $\rightarrow$ decidable

$w \in L(\text{DFA})$

33. $\{ \langle \text{PDA} \rangle \mid \text{PDA accepts string 'w'} \}$ $\rightarrow$ decidable

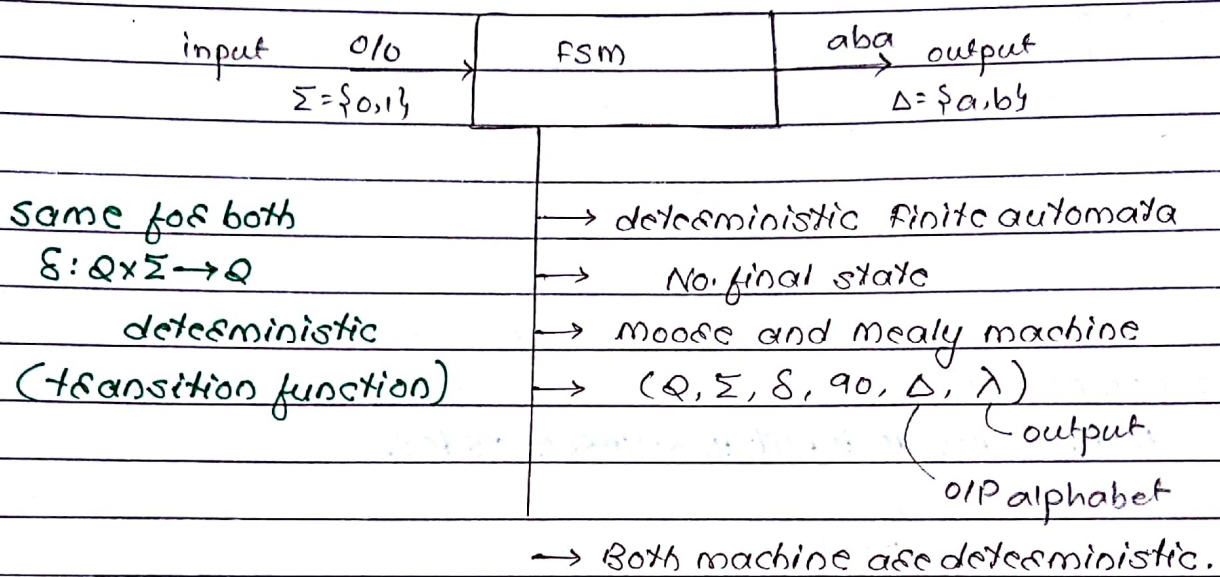
34. $\{ \langle \text{CFG}_i \rangle \mid \text{CFG}_i \text{ is ambiguous} \}$ $\rightarrow$ Semidecidable
35. $\{ \langle \text{CFG}_i \rangle \mid \text{CFG}_i \text{ is not ambiguous} \}$ $\rightarrow$ Not REL
36. $\{ \langle \text{PDA}_1, \text{PDA}_2 \rangle \mid L(\text{PDA}_1) = L(\text{PDA}_2) \}$ $\rightarrow$ Not REL
- $L(\text{PDA}_1) \neq L(\text{PDA}_2)$ $\rightarrow$ Not REL
37. $\{ \langle \text{DFA} \rangle \mid \text{DFA accepts regular} \}$ $\bar{L} = \emptyset$ $\rightarrow$ decidable
38. $\{ \langle \text{PDA} \rangle \mid \text{PDA accepts CFL} \}$ $\rightarrow$ decidable
39. $\{ \langle \text{DFA} \rangle \mid L(\text{DFA}) = a^* \}$ $\rightarrow$ decidable
40. $\{ \langle \text{PDA} \rangle \mid L(\text{PDA}) = a^* \}$ $\bar{L} \rightarrow$ not REL $\rightarrow$ Not REL
41. $\{ \langle \text{TM} \rangle \mid \text{size of TM} = 3 \}$ $\rightarrow$ decidable
42. $\{ \langle \text{TM} \rangle \mid \text{no. of states}$
42. $\{ \langle \text{PDA} \rangle \mid L(\text{PDA}) = \text{finite} \}$ $\rightarrow$ decidable
43. $\{ \langle \text{TM} \rangle \mid |\text{TM}| \geq 3 \}$ $\rightarrow$ decidable
44. $\{ \langle \text{TM} \rangle \mid |\text{TM}| \leq 3 \}$ $\rightarrow$ decidable
45. $\{ \langle \text{TM} \rangle \mid \text{TM reaches states 'q' within 3 moves} \}$ $\rightarrow$ decidable
46. $\{ \langle \text{TM} \rangle \mid \text{TM reaches states 'q' more than 3 moves} \}$ $\rightarrow$ Semidecidable
47. $\{ \langle \text{TM} \rangle \mid \text{TM accepts REL} \}$ $\bar{L} = \emptyset$ $\rightarrow$ decidable
48. $\{ \langle \text{TM} \rangle \mid \text{TM accepts string 'w'} \}$ $w \in L(\text{TM})$ $\bar{L} = \text{no. L} \rightarrow$ semidecidable
49. $\{ \langle \text{TM} \rangle \mid \text{TM accepts string '01'} \}$ $01 \in L(\text{TM})$ $\rightarrow$ semidecidable
50. $\{ \langle \text{TM} \rangle \mid \text{TM accepts only string '01'} \}$ $L(\text{TM}) = \{01\}$ $\rightarrow$ NOT REL (No logic)
51. $\{ \langle \text{TM} \rangle \mid \text{TM accepts some string} \}$ $L(\text{TM}) \neq \emptyset$ $\rightarrow$ Semidecidable
52. $\{ \langle \text{TM} \rangle \mid \text{TM accepts some every string} \}$ $L(\text{TM}) = \Sigma^*$ $\rightarrow$ Not REL
53. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) \leq \{00, 111\} \}$ $\rightarrow$ NOT REL.
54. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) \neq \{00, 111\} \}$ other than 00, 111. $\rightarrow$ Semidecidable
55. $\{ \langle \text{TM} \rangle \mid \text{TM accepts some string of length 2014} \}$ $\bar{L} = \text{No logic}$ $\rightarrow$ Semidecidable.
56. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = \text{Regular} \}$ $\rightarrow$ No logic $\rightarrow$ NOT REL
57. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = a^* \}$ $\rightarrow$ NOT REL
58. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = \text{CFL} \}$ $\rightarrow$ NOT REL
59. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = \text{DCFL} \}$ $\rightarrow$ NOT REL
60. $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = \text{ESL} \}$ $\rightarrow$ NOT REL
61. $\{ \langle \text{TM}_1, \text{TM}_2 \rangle \mid L(\text{TM}_1) = L(\text{TM}_2) \}$ $\rightarrow$ NOT REL.
62. (51). $\{ \langle \text{TM} \rangle \mid L(\text{TM}) = \emptyset \}$ $\rightarrow$ NOT REL

52. $\bar{L} = \{ \langle TM \rangle \mid L(TM) \neq \Sigma^* \}$ no logic $\rightarrow$ NOT REL.

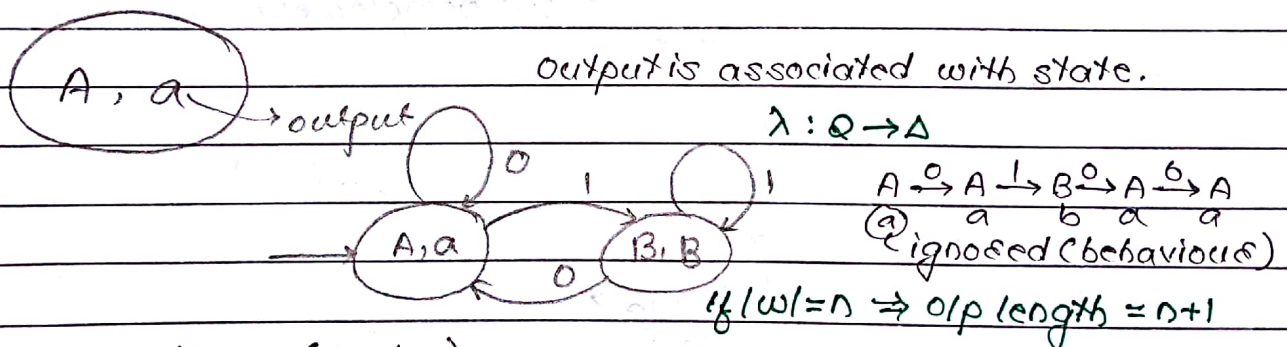
53. $L(TM) = \{ \epsilon, \{00\}, \{11\}, \{00, 11\} \} \rightarrow$ no logic.

$\{ \langle TM \rangle \mid TM \text{ halts on } w \} \rightarrow$ semidecidable. $\bar{L} \rightarrow$ No logic $\rightarrow$ Not REL

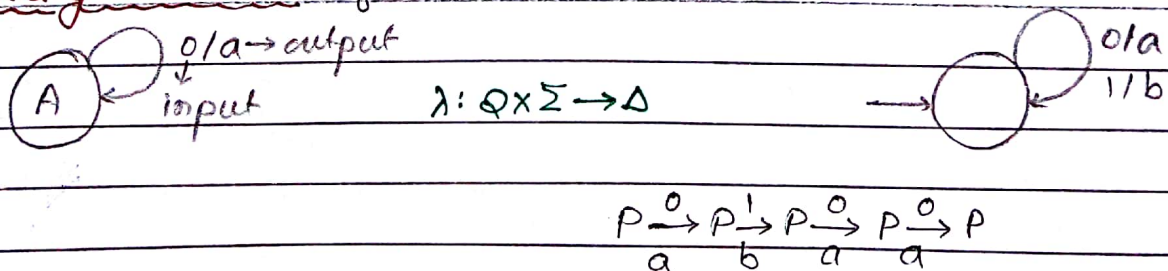
Finite State machine without Output:



Moore Machine



Mealy Machine (faster)



Construction of FSM with output:

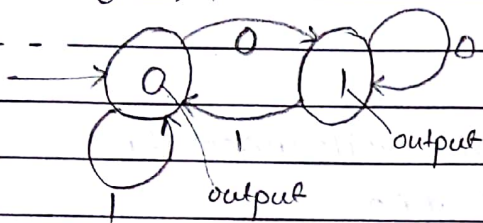
01. 1's complement of binary numbers.

moore machine

i/p o/p

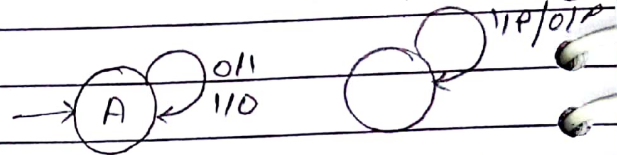
1 → 0

0 → 1



mealy machine

i/p



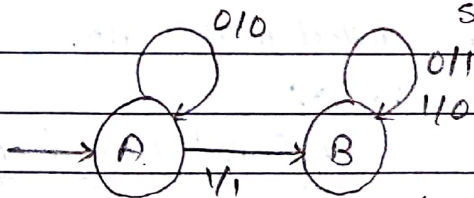
02. 2's complement

Note: Give an input in reverse order.

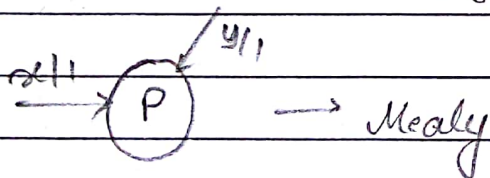
010100 → Given

101100 → 2's complement

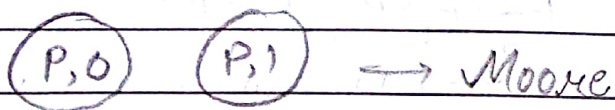
Same



Look
incoming
o/p's



Mealy



Moore

Applications:

- Designing digital circuits.
- 1's complement
- 2's complement
- Addition of any two binary nos.
- Increment
- Decrement
- Subtractors
- Adders
- Comparators, etc
- Multiplication of any binary number with any constant.
- Counting Sequence

03. Addition of two binary numbers.

without carry $\Rightarrow$

0	0	1	1
0	1	0	1
0	1	1	0

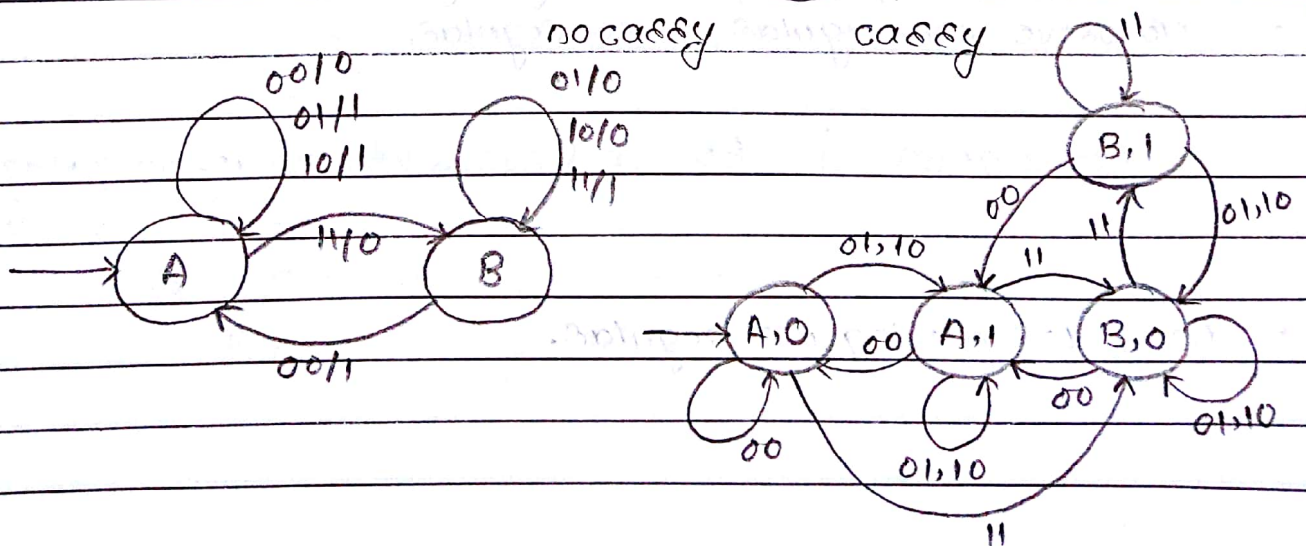
$\xrightarrow{\text{carry}}$

$\nearrow$ digit of 2nd no
 $\nwarrow$ output bit.

with carry $\Rightarrow$

0	0	1	1
0	1	0	1
1	0	0	1

digit of 1st no $\nwarrow$ output bit.

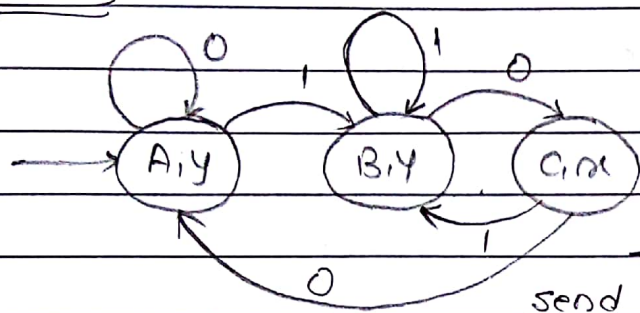


4. Count no. of 1's in binary number.

01011010111000010
 ↓ x x x x x x
 y

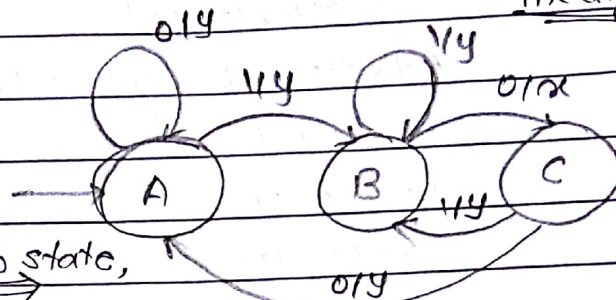
If '10' occurs produce 'x', else produce y.

Moore.



at each state,
 send o/p to all incoming.

Mealy



5. Identify binary number which starts with '00'.

If starts with '00' produce x else produce 'y'.

6. Identify binary number which ends with '00'.

7. Identify binary number which contain '00'.

8. Identify binary number which has atleast length 2.

Pumping Lemma

Pumping lemma for Regular

- It satisfy regular language.
- It is not used to prove for checking regular or not.
- To prove non-regular as non-regular.



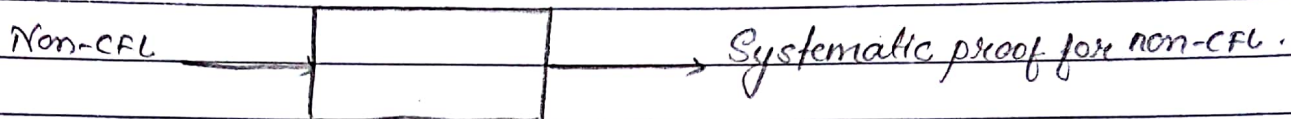
- Proof for proving not regular.

- If regular language is input to pumping lemma, it says regular as non-regular.

Never give regular as input to pumping lemma.

Pumping lemma for CFL.

- It satisfy CFL.
- Cannot be used for proving CFL or not.
- To prove not CFL.



- proof for proving not CFL.
- Do not give

- Pumping lemma uses proof by contradiction
- Pumping lemma uses pigeon hole principle.

Proof by Contradiction:

I/p: Non-regular (L)

⇓

Pumping lemma: It assumes L is regular.

⇓

Try to prove L as regular

⇓

I can't prove L as regular.

⇓

So, Contradiction occurs, Assumption failed,
So, L is not regular.