1 to 80: Regular Language,
80 to 112: Context free Language
112 to 147: Turing Machine

Introduction

Set: (language) : It is a collection of objects.
unordered
(well defined — member, cardinality (size of set → 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

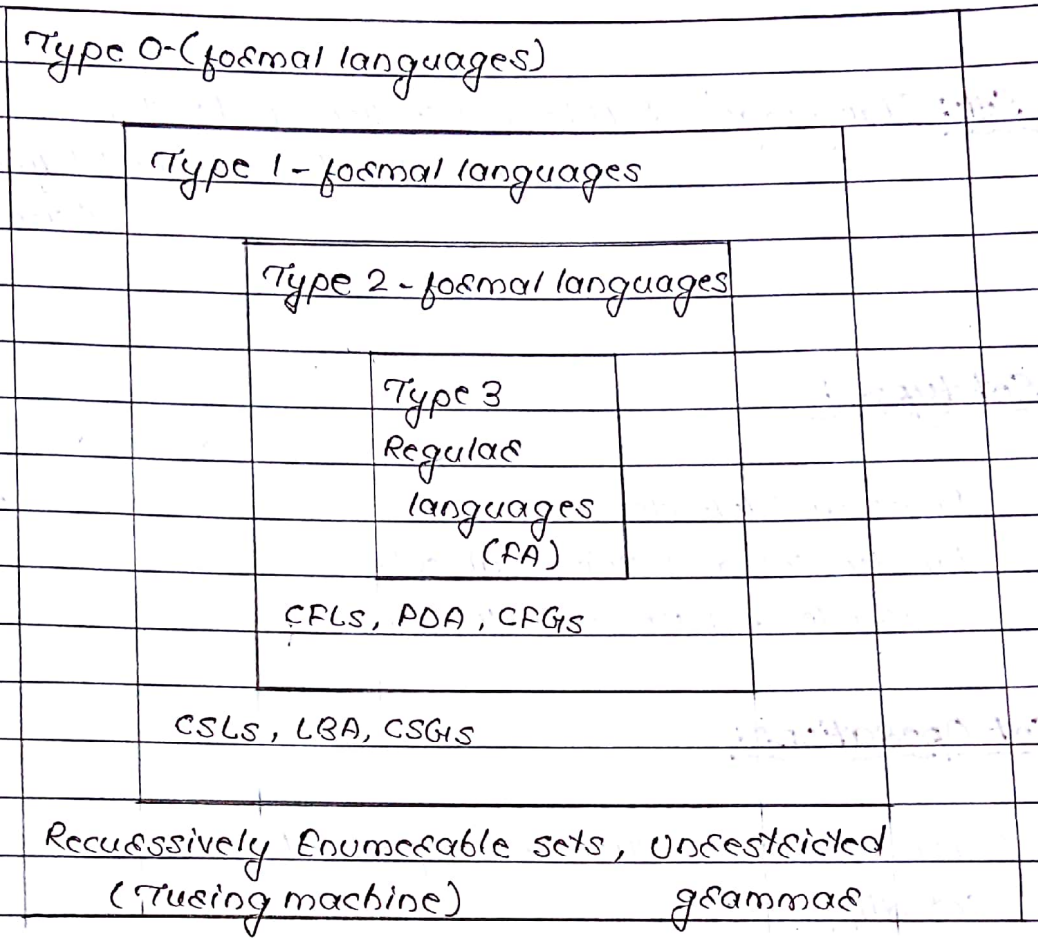
$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$ let $|A|=m, |B|=n$, $\max(m,n) \le |A \cup B| \le m+n$

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

$\overline{A} = \{x \mid x \notin A\} = \{x \mid x \in U, x \notin A\}$

$A - B = \{x \mid x \in A, x \notin B\} = A \cap \overline{B}$

Basics of TOC:



- set of all regular languages
  ⊂
  set of all context free languages
  ⊂
  set of all context sensitive language
  ⊂
  set of all Recursively Enumerable languages.

- Type 3 ⊂ Type 2 ⊂ Type 1 ⊂ Type 0
- FA < PDA < LBA < TM
- LS(FA) ⊂ LS(PDA) ⊂ LS(LBA) ⊂ 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.

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

$\|w\|=0$	$\|w\|=1$	$\|w\|=2$	$\|w\|=3$	
λ or ϵ	a	aa	aaa	aaaa
	b	ab		
		ba		
		bb	bbb	bbbb

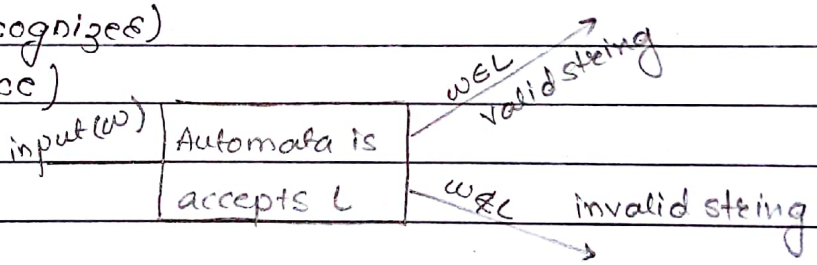
Language: Set of strings. $|\{\epsilon\}| = 1$ $|\epsilon| = 0$ $|\{\}| = 0$
(set ↙, ↳ string, size of set)

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

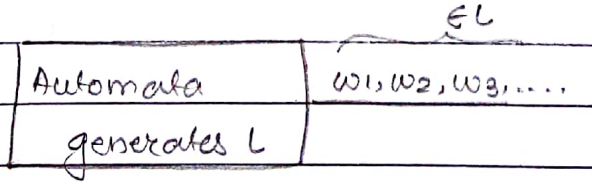
- Four types of languages have logic to compute.

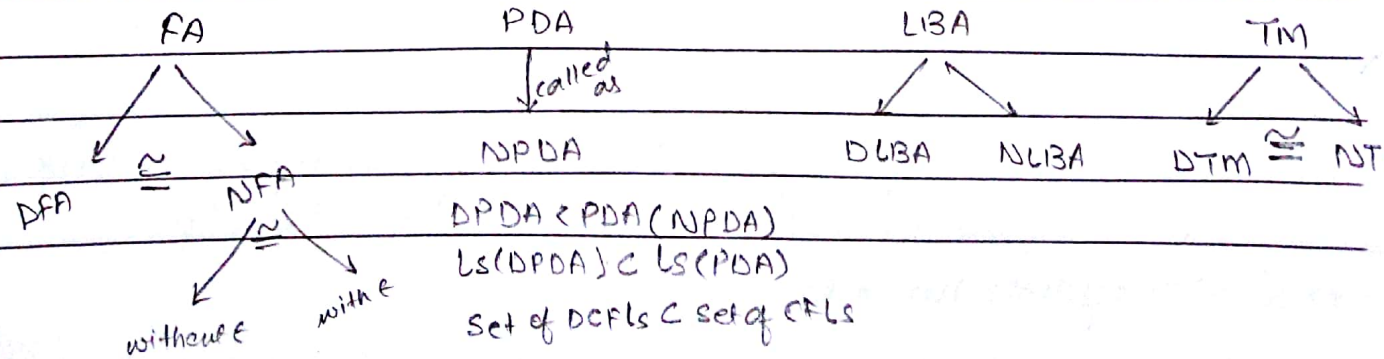
Automata:

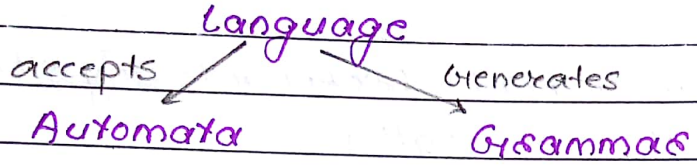
1. Acceptor (Recognizer) (Universal acceptance)



2. producer / Generator / Enumerator / Transducer







Grammar = (V, T, P, S)

V → set of Variables (Non-Terminals) V = {S, A, B}

T → set of Terminals T = {a, b}

P → set of rules (production) P = {S → AB, A → a}

S → Start Symbol.

Grammar is set of rules which generates some rules.

Automata is a machine which accepts some language.

Finite Automata

1. Regulas Expression
2. Finite Automata (***DFA, NFA)
3. Regulas Gisammas (LLG1, RLG1)
4. Equivalences (RE ≅ FA ≅ Reg. Gisammas)
5. How to identify a segulas languages?
6. Closusse psopesties of segulas languages.

Regulas Languages (Regulas sets)

A → Accept
G1 → Gienesates.

G ≅ A ≅ A ≅ A ≅ G1 ≅ G

R.E. DFA NFA ε-NFA LLG1 RLG1

Regulas Expsession

- It sepsesents segulas language.
- It genesates a segulas language.
  L(R.E) = Regulas Language
- It has a declasative way to sepsesent a segulas language.
- Fous opesatoss: * + . +
  unasy Binasy

operatos → Logic → progsam → functh → selatn set.

* → kleene closusse
+ → positive closusse
. → Concatenation
+ → OR.

1. + (OR) set.
   a+b = {a} ∪ {b}
   ∴ ab+bba = {ab, bba}

2. concatenation (.) symbol.
   a.b = ab
   (aaa).(bb) = aaabb (string)
   a*.b* expression

3. Kleene closusse (*)
   a* = a^≥0 = a^any
   a* = {a^n | n ≥ 0}
   = set of all strings ovee Σ = {a}

4. Positive closure (+)

$a^+ = a^{\geq 1}$ ← atleast one

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

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

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

	Regular Expression	Set
1.	ϕ ↳ empty exp.	$\{\} = \phi$ ↳ 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^*	$\{\epsilon, a, aa, \ldots\} = \{a^n \mid n \geq 0\}$ $= \{a^n\}$
9.	a^+	$\{a, aa, \ldots\} = \{a^n \mid n \geq 1\}$ $= \{a^{n+1} \mid n \geq 0\}$

0. Simplying the following Regular Expression.

$\phi.any = \phi$

$any.\phi = \phi$

1. $\epsilon + \epsilon = \epsilon$
2. $\epsilon.\epsilon = \epsilon$
3. $\phi.\phi = \phi$
4. $\phi.\epsilon = \phi$
5. $\epsilon.\phi = \phi$
6. $a.a = aa$
7. $\epsilon^* = \epsilon$ $[any]^0 = \epsilon$
8. $\phi^* = \epsilon$
9. $\phi^+ = \phi$
10. $\epsilon^+ = \epsilon$
11. $\phi + \phi = \phi$
12. $\phi.(a+\epsilon) = \phi$
13. $a+a = a$
14. $a.a = aa$
15. $(\epsilon+\phi)^+ = (\epsilon)^+ = \epsilon$
16. $(\epsilon+\phi)^* = (\epsilon)^* = \epsilon$
17. $(\epsilon.\phi)^+ = \phi^+ = \phi$
18. $(\epsilon.\phi)^* = \phi^* = \epsilon$
19. $(\epsilon.\epsilon)^+ = \epsilon^+ = \epsilon$
20. $(\phi.a)^* = \phi^* = \epsilon$
21. $a+\epsilon = a+\epsilon$
22. $a.\epsilon = a$
23. $\epsilon.a = a$
24. $\epsilon.a.\epsilon = a$
25. $a.\epsilon.a = aa$
26. $\epsilon.\epsilon.a = a$
27. $a.a.\epsilon = aa$
28. $a.\epsilon.b = ab$
29. $ab.\epsilon.b = abb$
30. $(a+\epsilon).b = ab+b$

Properties of operators ('or' and '.')

1. closed (closure, operation, operator)
   (Domain, operation) is closed iff "$\forall x, y \in D \Rightarrow x \text{ op } y \in D$"

   i. or → closed for regular expressions
   ii. concatenation ⇒ closed
   iii. kleene closure ⇒ closed
   iv. positive closure ⇒ closed

2. Associative
   i. $[(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3)]$
   L→R left, R→L right.

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

or(+) satisfies associative

   ii. $(R_1 . R_2) . R_3 = R_1 . (R_2 . R_3)$
   $(a \cdot b)c = a(bc)$
   $\{a,b,c\}$ $\{a,b,c\}$

Concatenation (·) satisfies associative.

3. Identity
   or: $R + \phi = \phi + R = R$
   ϕ (empty exp.) is identity for or.

   concatenation: $R + \epsilon = \epsilon . R = R$
   ϵ (empty string) is identity for concatenation.

4. Inverse (always depends on identity)
   or: $R + [\,?\,] = [\,?\,] + R = \phi$ "inverse not exist for 'or'"
   concatenation: $R . [\,] = [\,] . R = \epsilon$ "inverse not exist for concatenation"
   (identity of concatenation)

5. Commutative:

OR: $R_1 + R_2 = R_2 + R_1$ Eg. $a+b = b+a$ ∴ satisfies.

concatenation: $R_1 . R_2 \neq R_2 . R_1$ Eg. $ab \neq b.a$ ∴ Not-satisfies

$R_1 . R_2 = R_2 . R_1$ for some R_1, R_2 1. $R_1 = R_2$ 4. $R_1 = \epsilon$

2. $R_1 = \phi$ 5. $R_2 = \epsilon$

3. $R_2 = \phi$

6. Distributive

OR over concatenation: $R_1 + (R_2 . R_3) = (R_1 + R_2) . (R_1 + R_3)$ ∴ Not satisfies.

concatenation over OR: $R_1 . (R_2 + R_3) = (R_1 . R_2) + (R_1 . R_3)$ ∴ satisfies

OR cannot be distributed over concatenation but concatenation can be distributed over OR.

Note: Dominant

$R . [\phi] = [\phi]$ dominator for any exp. in concatenation

$R + (\Sigma^*) = (\Sigma^*)$ ↗ universal set

Understanding Regular Expression (Language generated by expression)

1. $R = \phi$

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

2. $R = \epsilon$

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

3. $R = a^*$

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

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

= set of all strings over a's.

4. $R = a^+$

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

= set of all strings over a's except empty string.

5. $R = a+b$

$L(R)$ = Atleast one 'a' followed one 'b'

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

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

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

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

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

all strings covered in $(a+b)^*$

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

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

$= (a+b+aabab)^*$ 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'. → $(a+b)^*b$

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

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

7. 2nd symbol from beginning is a. → $(a+b)^*a(a+b)^*$

starts with 'aa' or 'ba' → $(aa+ba)(a+b)^*$

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

9. 2nd symbol is 'a' and 4th symbol is 'b'

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

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

(aaab, aabb, baab, babb) - starts.

10. Empty language - ϕ

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

$aax + bbx = (aa+bb)x = (aa+bb)(a+b)^*$ min = aa or bb

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

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

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 contain 'b'.

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

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

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

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

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

18. Each string starts with 'a' and contain '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 strings ends with 'a' and contains 'b'. $(a+b)^*b(a+b)^*a$

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

22. Each string ends with 'a' and contain 'a', → $(a+b)^*a$

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'

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

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

~~ea~~

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.

$(a+b)(a+b)^2 = (a+b)^2(a+b)$ or $(a+b)(a+b)(a+b)$

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

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

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

$\epsilon + (a+b) + (a+b)^2 + (a+b)^3$ 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 ~~4~~ 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.

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

$|w| \neq 3n$

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

35. Each string has exactly two a's. $b^*ab^*ab^*$

36. Each string has atleast two a's. ~~$b^*ab^*a(a+b)^*$~~

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

37. Each string has atmost two a's.

$b^* + b^*ab^* + b^*ab^*ab^*$ 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.

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

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

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

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 contain 'aaa' as a substring.

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

44. Each string not containing 'aaa' as a substring. ↘ FA ⇒ 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 $(b^*ab^*ab^*)^*b^*$

$b^*(b^*ab^*ab^*)b^*$

$b^*(ab^*ab^*)^*b^*$

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

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's of a's. $b^*ab^*(b^*ab^*ab^*)^*$

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

Languages over one symbol:

50. $(11+1111)^*$, $\Sigma = \{1\}$ $= \{\epsilon, 1^2, 1^4, 1^6, \ldots\}$

$= (11)^*$

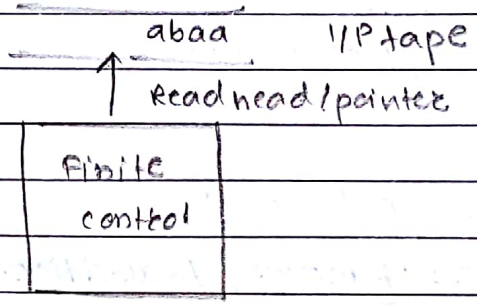
51. $(11+111)^*$, $\Sigma = \{1\}$ $= \{\epsilon, 1^2, 1^3, 1^4, 1^5, \ldots\}$

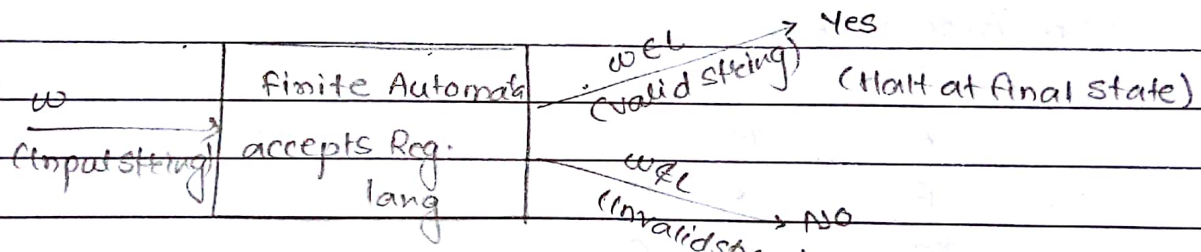
$= \epsilon + 111^* = \epsilon + 11^* = \epsilon + 1^*1$ $\{1^n \mid n \neq 1\}$

Finite Automata :

- Donot require any memory.
- It accepts regular language.
- FA → DFA, NFA, ε-NFA } equivalent.
- L(DFA) = L(NFA) = L(ε-NFA) = 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

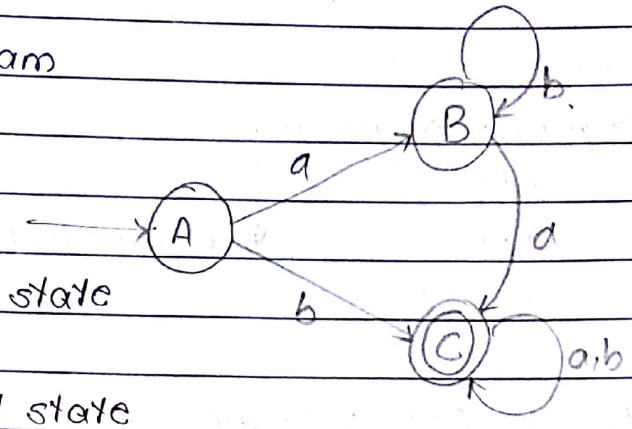
δ = Transition function

q_0 = start state $\quad q_0 \in Q$

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

Representation of Finite Automata:

01. State diagram



: Initial state

: Final state

: Non-final state

: transition on input '_'

How to run a string on machine?

(ω)

abba:

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

Sequence of moves / transitions / path

02. Transition Table

δ	a	b
→ A	B	C
B	C	B
*C	C	C

03. Mathematical model (set / relation / function)
(transition function)

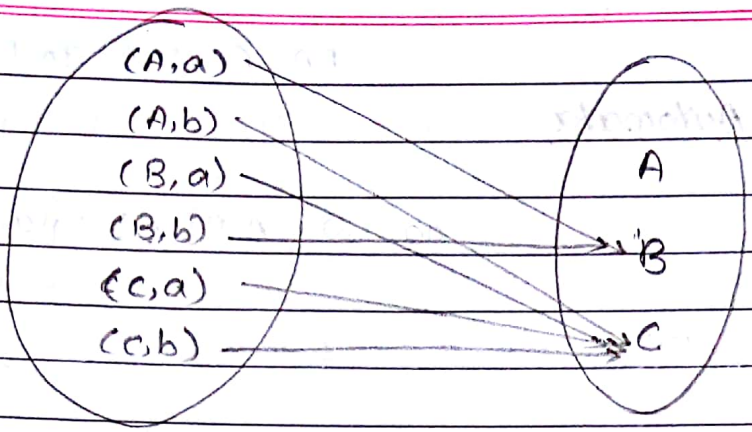
$Q = \{A, B, C\}$, $\Sigma = \{a, b\}$, $q_0 = A$, $F = \{C\}$

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

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

$\delta(A, b) = C$ $\quad \delta(C, a) = C$

$\delta(B, a) = C$ $\quad \delta(C, b) = C$

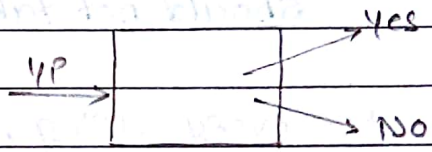


(may change as per type of FA codomain.

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

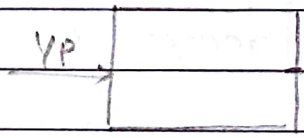
Finite Automata, Finite machine, Finite state machine are same.

- 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, Σ, δ, q0, F)

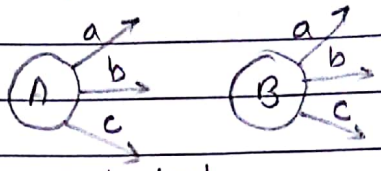
Deterministic Finite Automata

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

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

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

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



$|\delta| = |Q| * |\Sigma|$

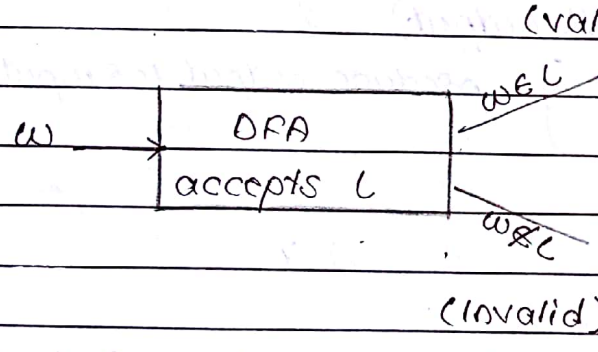
From each state, for every I/P symbol, exactly one transition to the next state.

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.

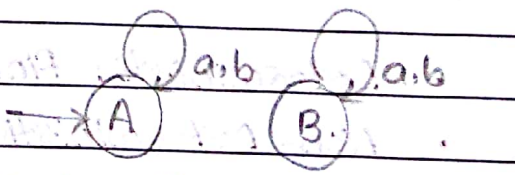
04.



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

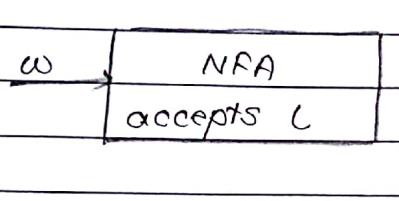
$2^Q = P(Q)$

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

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

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

04.



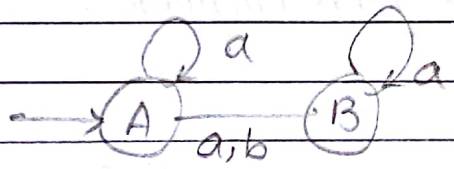
Atleast one path that goes to final state

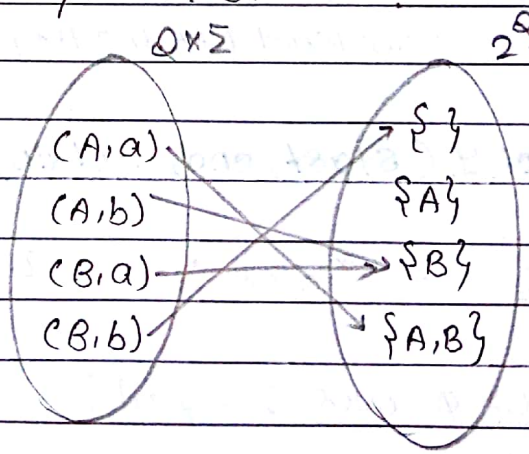
No path halts at final.

No path or every path goes to non-final.

05. For every valid strings, # paths ≥ 1

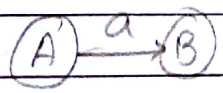
for every invalid string, # paths ≥ 0.



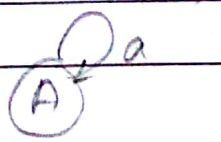


$\delta(A, a) = \{\}$

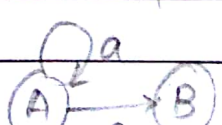
$\delta(A, a) = \{B\}$



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

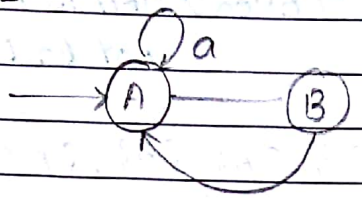


$\delta(A, a) = \{A, B\}$



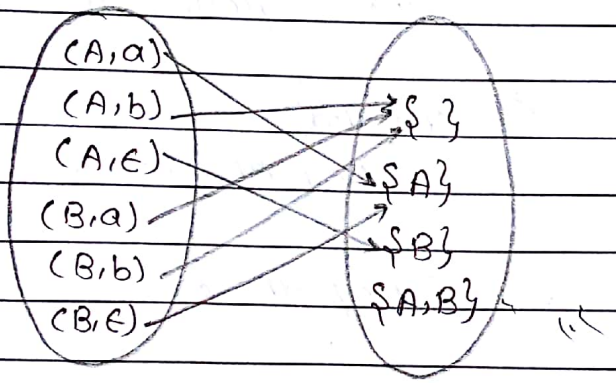
- **ϵ-NFA (Non-Deterministic Finite Automata with ϵ moves)**

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



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

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

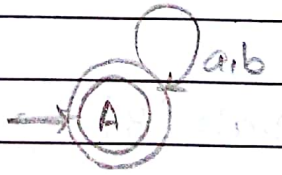


Construct NFA:

Construct NFA for the following regular languages.

Model 1 (start, end, contain)

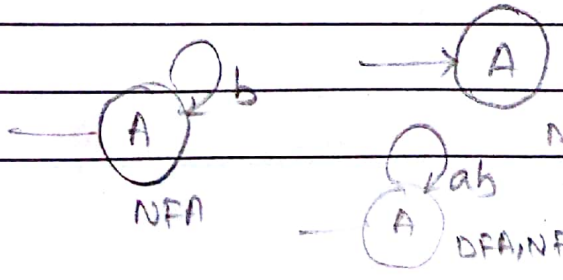
1. $L = (a+b)^*$ over $\Sigma = \{a,b\}$



ϵ: A

a: $A \xrightarrow{a} A$

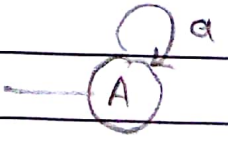
2. $L = \emptyset$ over $\Sigma = \{a,b\}$



ϵ: A

a: $A \xrightarrow{a} ?$

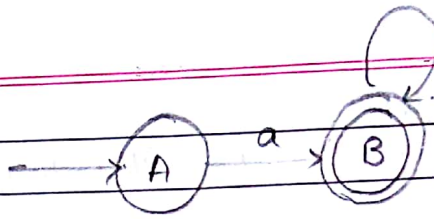
NFA but not DFA.



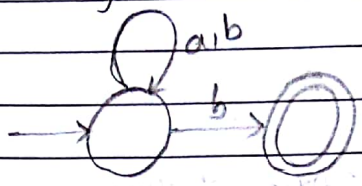
NFA (but not DFA)

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

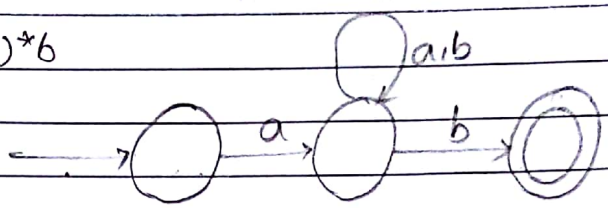
$= \{ aw \mid w \in (a+b)^* \}$



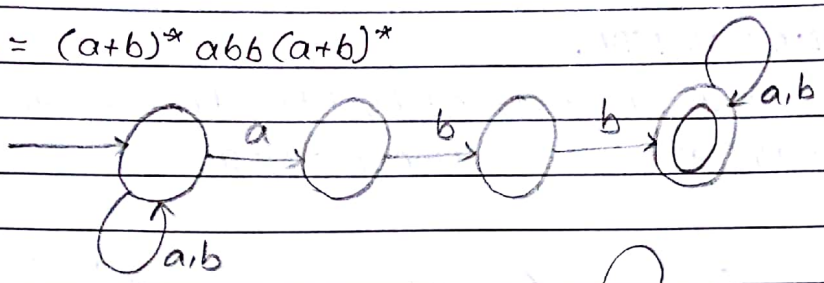
4. $L = (a+b)^* b$



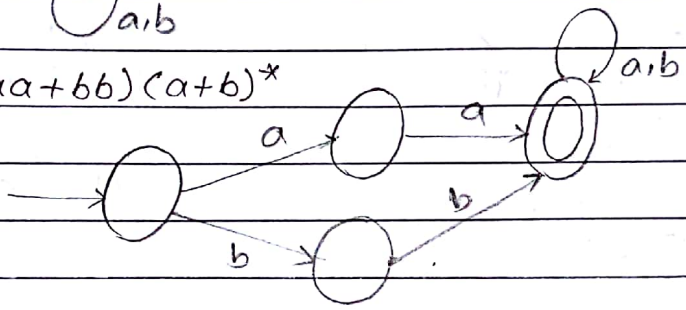
5. $L = a(a+b)^* b$



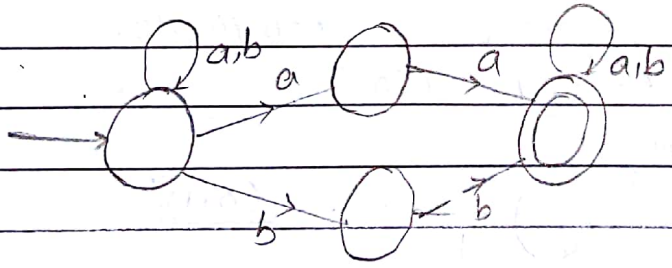
6. $L = (a+b)^* abb(a+b)^*$



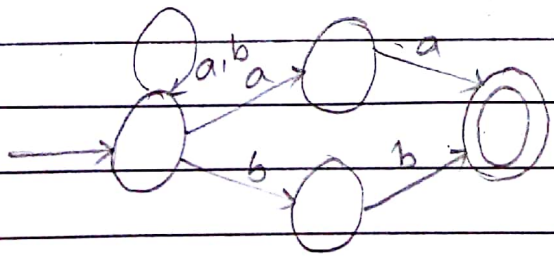
7. $L = (aa+bb)(a+b)^*$



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

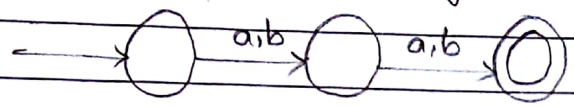


9. $L = (a+b)^*(aa+bb)$

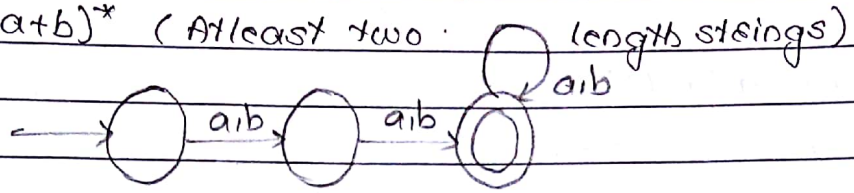


Model 02:

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

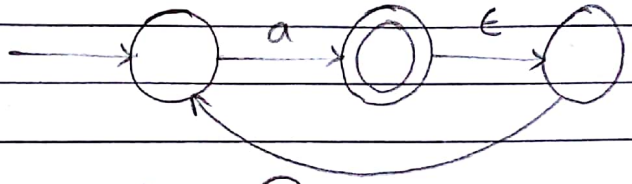


11. $L = (a+b)^2(a+b)^*$ (Atleast two length strings)



Model Questions of NFA.

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

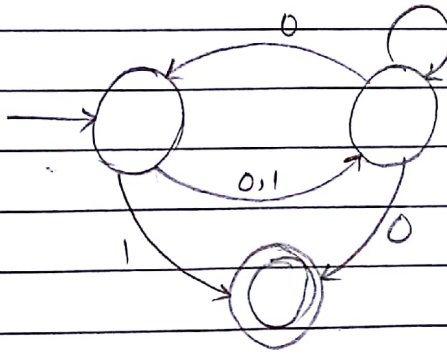


$L = \{a, aa, \ldots\}$

$L = \{a^+$

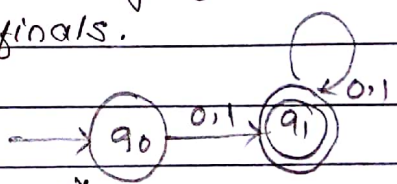
$\therefore L' = \{\epsilon\}$

02



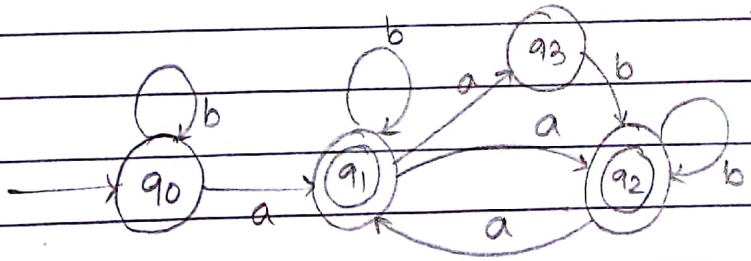
M' obtained by interchange finals & non-finals.

$L(M') = ?$



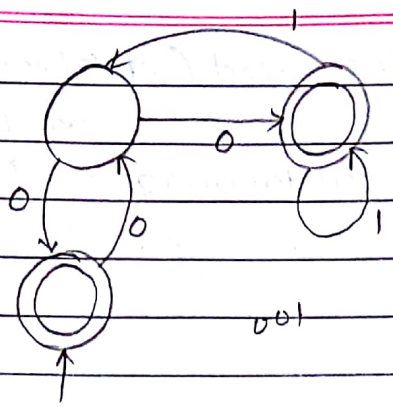
$(0+1)^*$

03.

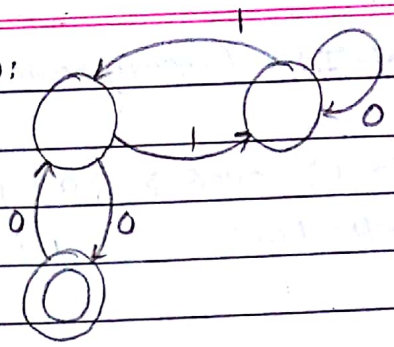


c. $b^*a(a+b)^*$

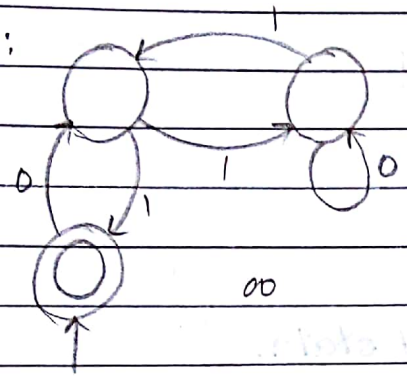
04. (P):



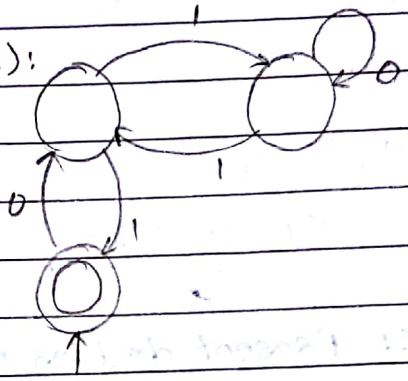
(Q):



(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.

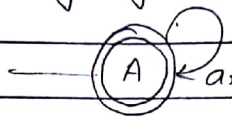
Construction of NFA:

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 deadstate.

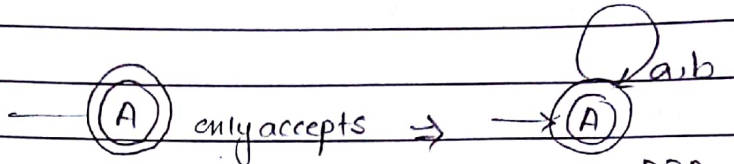
Model I: Purely minimum string based (start, end, contain conditions)

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

method 01: min $=\epsilon$



method 02: min $=\epsilon$



i. $A \xrightarrow{a} |\ \square$

$a \in L$ goto final

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

ii. $A \xrightarrow{b} |\ \square$

$b \in L$ goto final

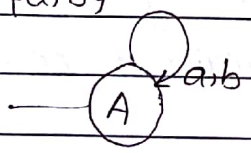
step 1: Take min string

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

Step 3: Find all missing transitions to make DFA.

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

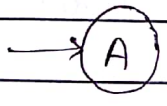
method 01:



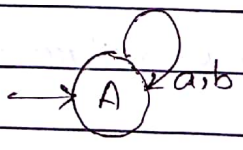
$\in$: A

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

method 02:



DFA



i. $A \xrightarrow{a} |\ \square \Longrightarrow$ final

present: a — future: $\in L$

ii. $A \xrightarrow{b} |\ ? \Longrightarrow$ final

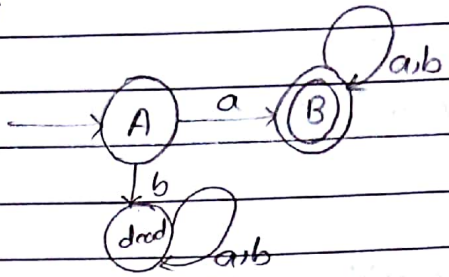
$b \notin L$ — future: no future exist goto dead state

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

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

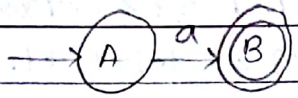
method 01:

min a



method 02:

min a



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

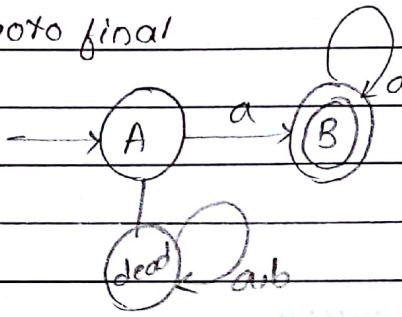
$b \notin L$ present, future not able to decide goto dead.

ii. $a\ B \xrightarrow{a} [\ ?\ ]$

$aa \in L$ goto final

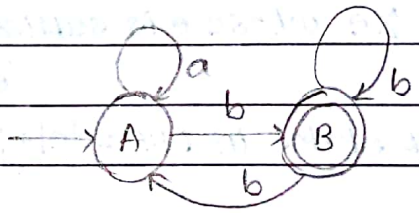
iii. $a\ B \xrightarrow{b} [\ ]$

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

$a \in L$

$A \xrightarrow{a} [\ ?\ ] \xrightarrow{b}$ final

$ab \in L$

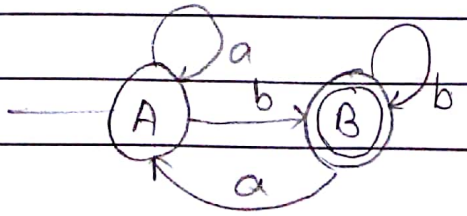
$= A.$

ii. $ba\ B \xrightarrow{a} [\ ?\ ]$

$baab.$

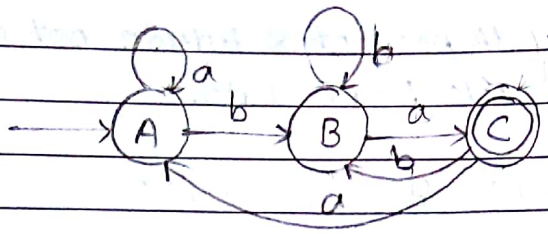
iii. $bb\ B \xrightarrow{b} ?$

$bb \in L.$



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

min: aba



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

min string: abb — 4 states

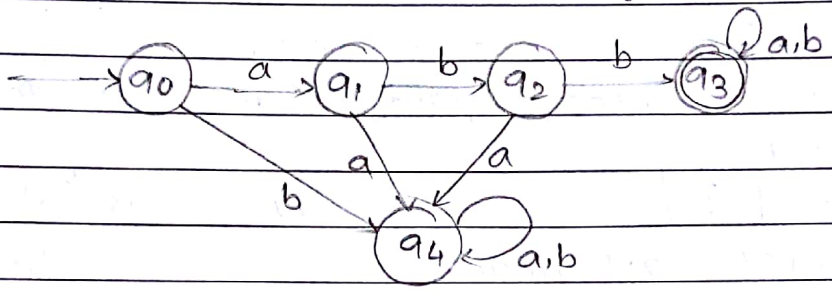
1 dead state

∴ Total: 5 states

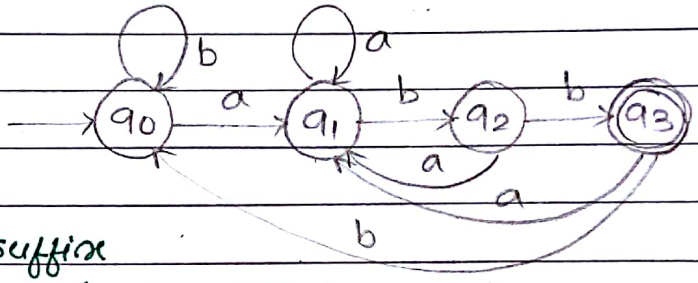
$L = sX$ where s is prefix (start condition)

⇓ No. of states in DFA = $(|s|+1)+1$.

(+1 of dead state)



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

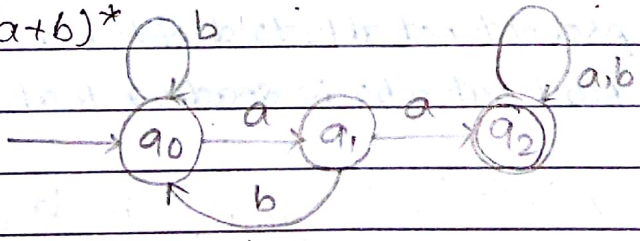


$L = Xe$ where e is suffix

⇓ (end condition)

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

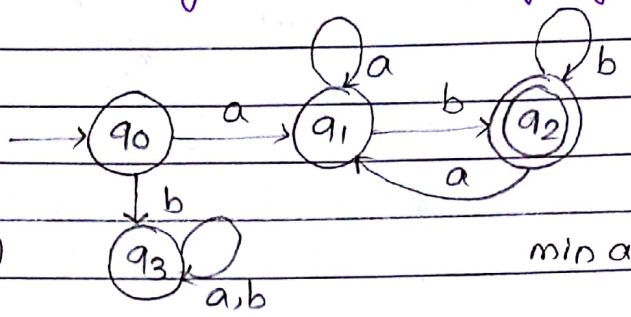
8. $L = (a+b)^*aa(a+b)^*$



Module 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')



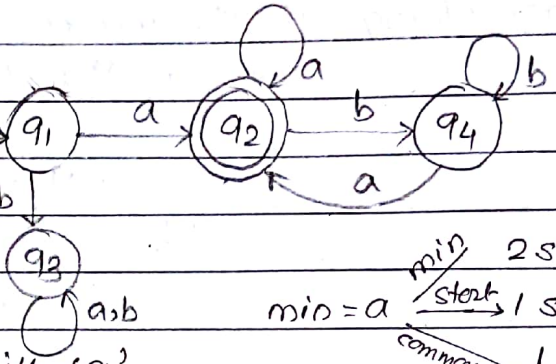
min ab → 3 states

1 dead

Total states: 4.

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

min = a (common symbol) will be duplicated.



Each string starts and ends with 'a'.

min = a → min / start → 1 states; common symbol → 1 states → 2 states

Total states: 4.

11. Each string starts with 'a' and ends with 'ab'.

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

min ab → 3 states

min / start → 1 states

common → 1 states

Total: 4 states

12. Each string starts with 'aba' and ends with 'bab'.

$abab + aba(a+b)^*bab$

min = abab → 5

min / start → 1

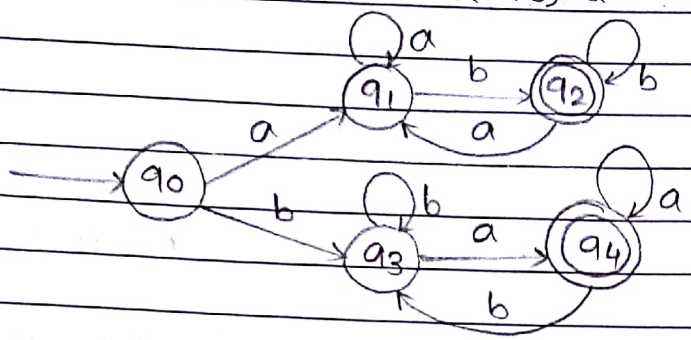
c.s.(ba) → 2

1

8

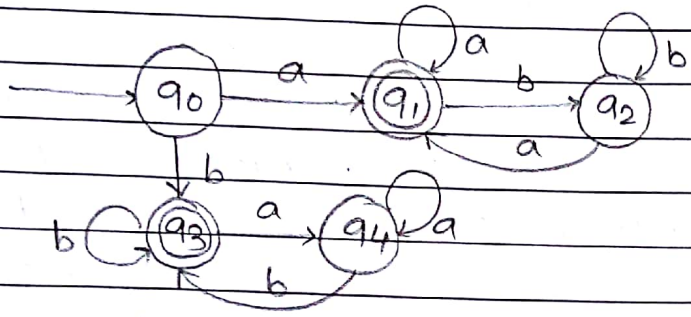
13. Each string starts and ends with different symbols.

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



14. Each string starts and end with same symbols.

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



Union = Σ^*

$q_2 = a \times b$

$q_4 = b \times a.$

$q_0 = \epsilon$

$q_1 = a + a \times a$

$q_3 = b + b \times b.$

Note:

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

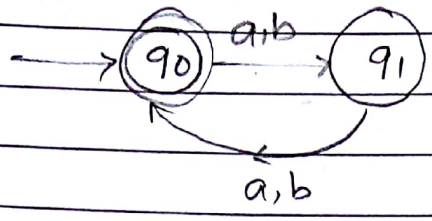
If every state is non-final in DFA, then it accepts empty language.

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 = remainder 0

q_1 = remainder 1

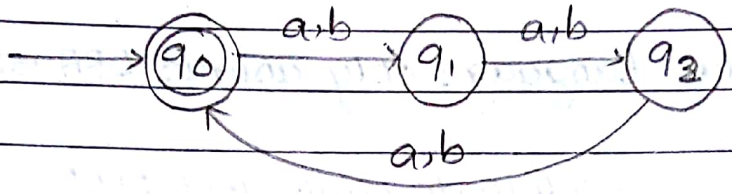
16. Length of $|w| = 0 \bmod 3$

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

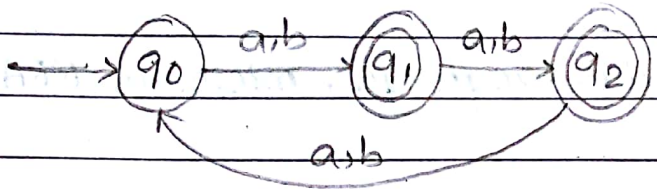
q_0 : remainder 0

q_1 : remainder 1

q_2 : remainder 2



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



Any Regular Language (L)

⇓

DFA accepts language

⇓ Interchange finals and non-finals

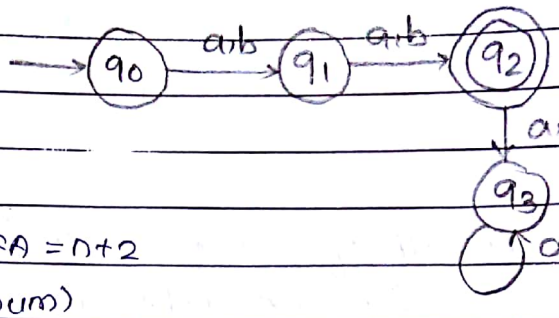
modified DFA

⇓

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 $= n+1$

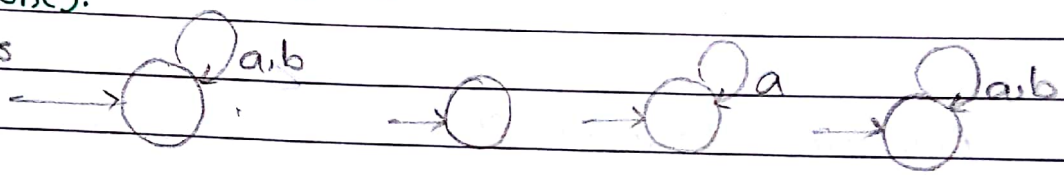
Note:

For every Regular Language only unique DFA exist.

Regular Language (but only one mini) $\begin{cases} \rightarrow \text{Infinite equivalent DFA's} \\ \rightarrow \text{Infinite equivalent Regular expression} \\ \rightarrow \text{Infinite equivalent NFA's} \end{cases}$

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

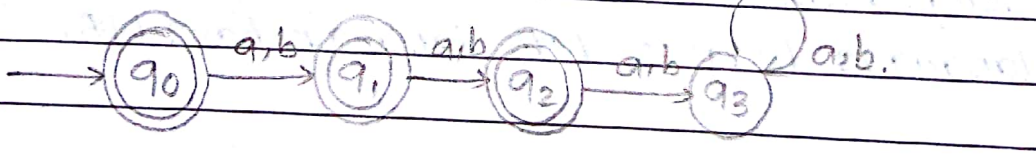
minimum NFA's



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

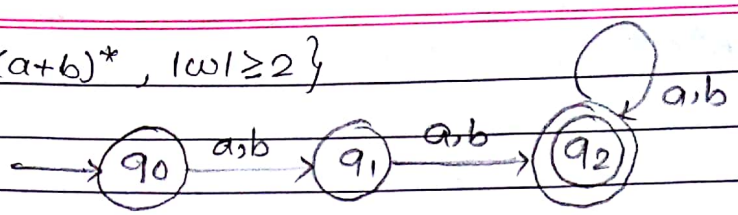
length of each string is atmost 2.



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.

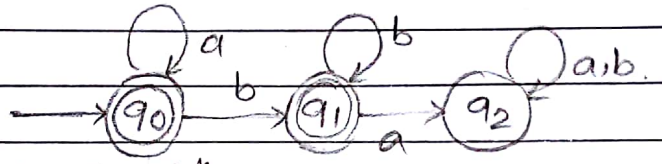
Model 4: Sequence

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

[A] = a^*

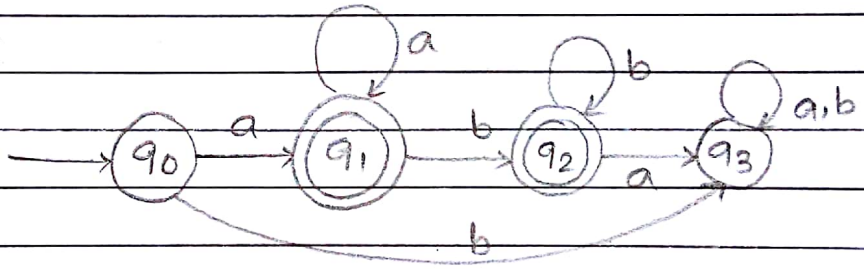
[B] = a^*b^+

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

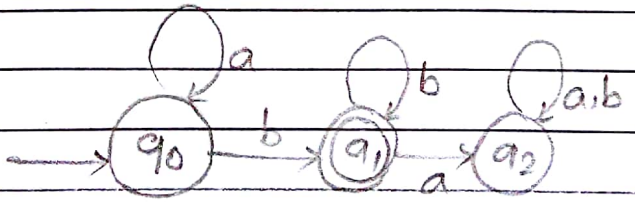


22. $L = a+b^*$

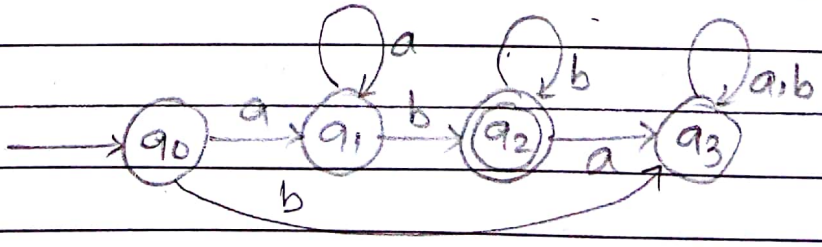
min = a



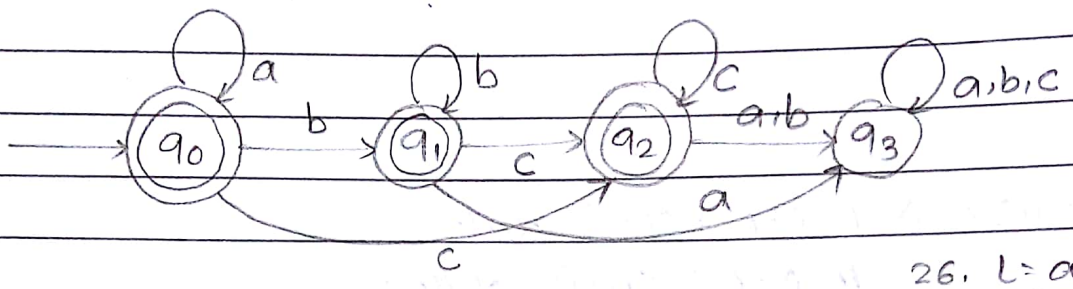
23. $L = a^*b^+$



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



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



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

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

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

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

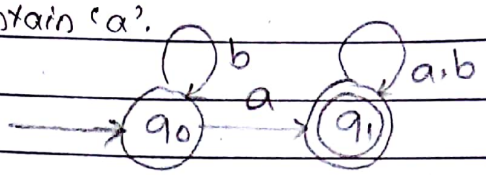
30. $a^+b^*c^+$

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

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

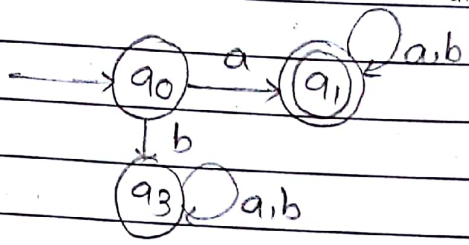
Model 5: (Subset / Super set) (Two conditions will be converted to one condition)

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

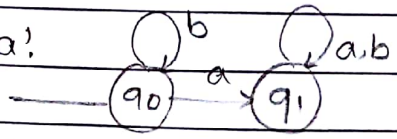


$(b^*a(a+b)^*)$

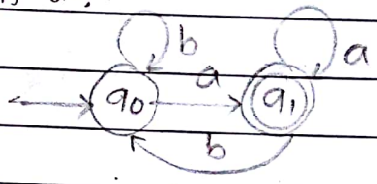
34. Each string starts with 'a' and contain 'a'.



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



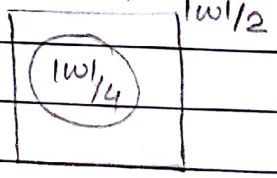
36. Each string ends with 'a' and contain 'a'.



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

$|w| = 0$ modulo 2

$\therefore$ No. of states: 2

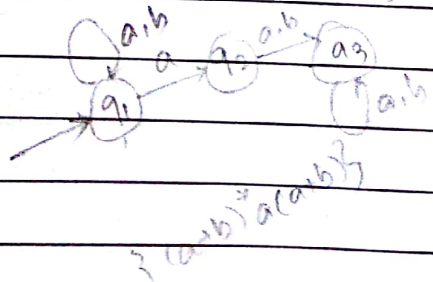


38. $\{w \mid w \in (a+b)^*, |w| = 0 \text{ modulo } 2, |w| = 0 \text{ modulo } 4\}$

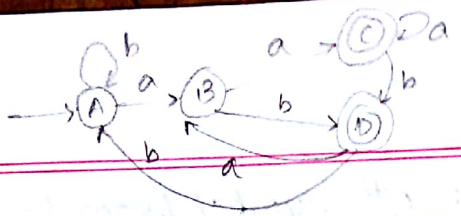
same as $|w| = 0$ modulo 4

$\therefore$ No. of states = 4

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



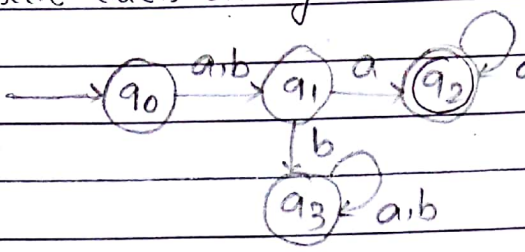
$(a+b)^*a(a+b)$



Model 6: Position based.

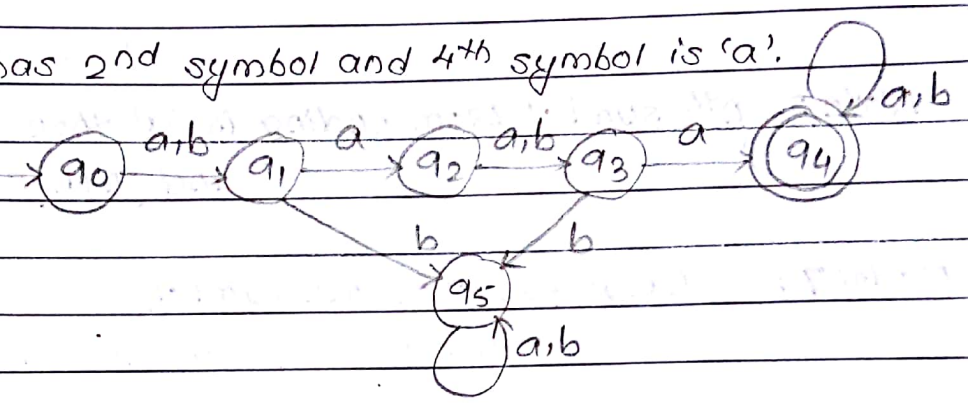
39. Set of all strings where each string has 2nd symbol is 'a'.

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



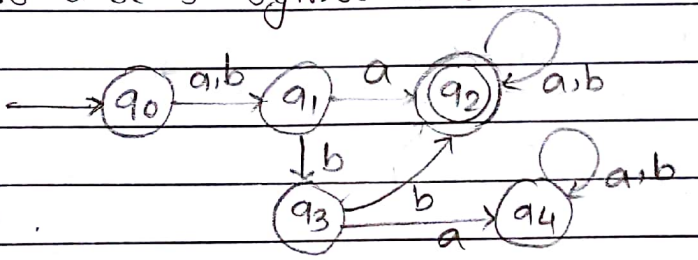
40. Each string has 2nd symbol and 4th symbol is 'a'.

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

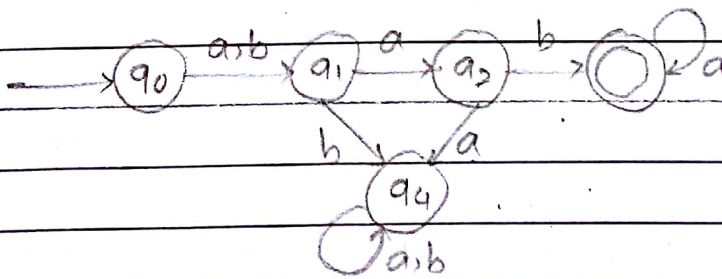


41. 2nd symbol is 'a' or 3rd symbol is 'b'.

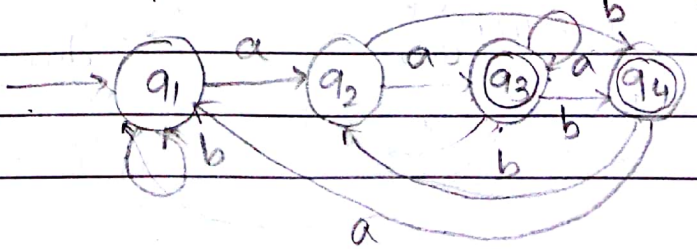
$\alpha aX + \alpha bbX$



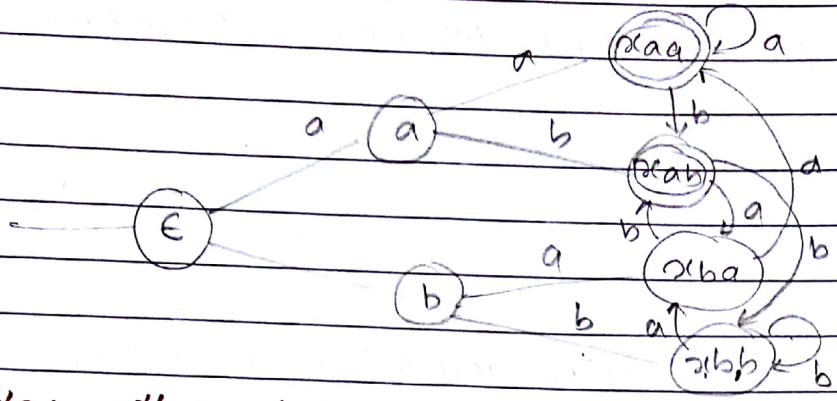
42. Set of all string where each string has 2nd symbol is 'a' and 3rd symbol is 'b'.



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



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



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

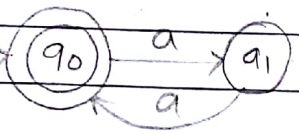
Model 3: (Language over one symbol)

It should form Arithmetic Progression.

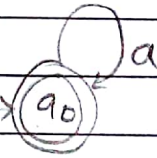
44. $L = (aa)^*$ $\{\epsilon, aa, aaaa, \ldots\}$ $\Sigma = \{a\}$

$(a^2)^*$

$(a^2)^n$

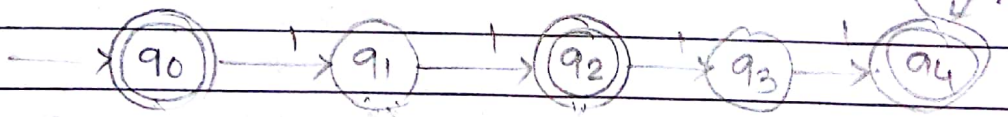


45. $L = a^*$ $\{a^0, a^1, a^2, \ldots\}$

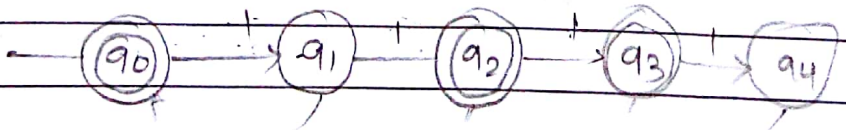


Note: $(a^2)^* \neq (a^*)^2 = a^*.a^* = a^*$

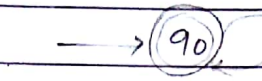
46. $L = (11+11111)^*$ $\{\epsilon, 1^2, 1^4, 1^5, \ldots\}$



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



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

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

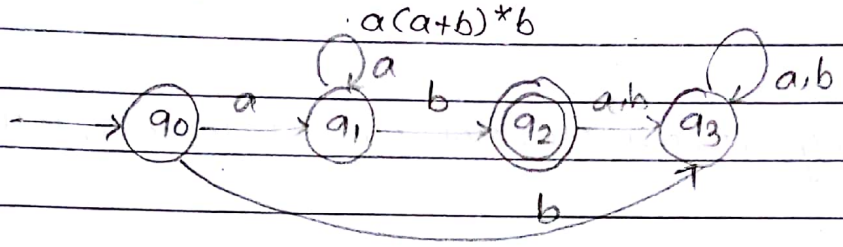
50. $L=(1111+11111)^* = \{\epsilon, 1^4, 1^5, 1^8, 1^9, 1^{10}, 1^{12}, 1^{13}, \ldots\}$

13 states

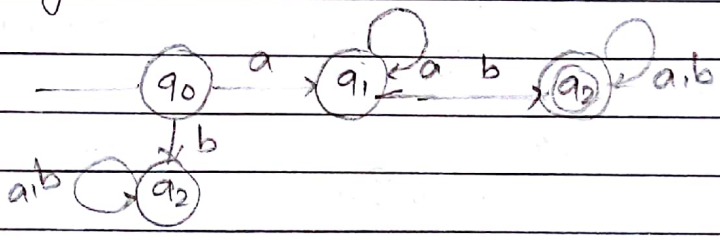
Model 8: (More conditions but creates sequence)

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

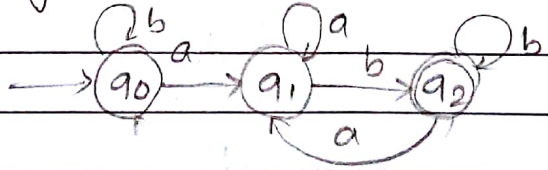
a(a+b)*b



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



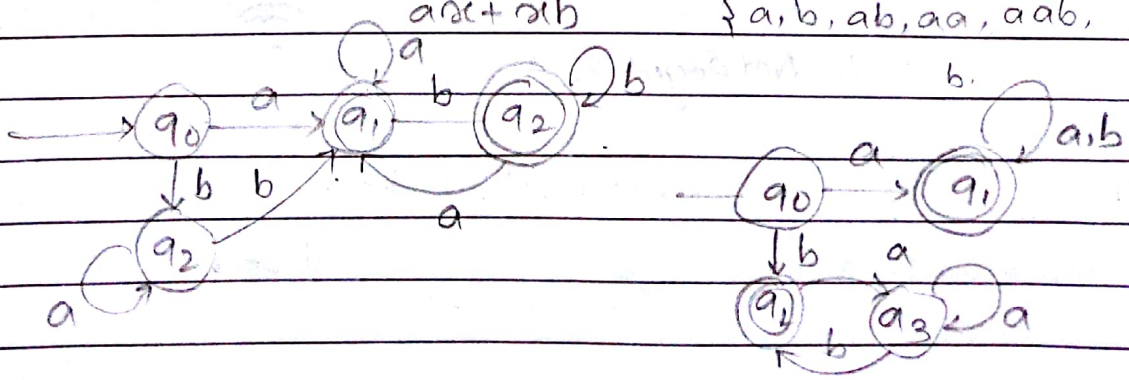
53. Where each string ends with 'b' and contain 'a'.



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

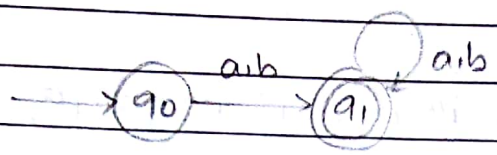
aa+ab

{a, b, ab, aa, aab,



55. where each string starts with 'a' or contain 'b'. $(a+b)^+$

equal. ↑ ends with 'a' or contain 'b'.



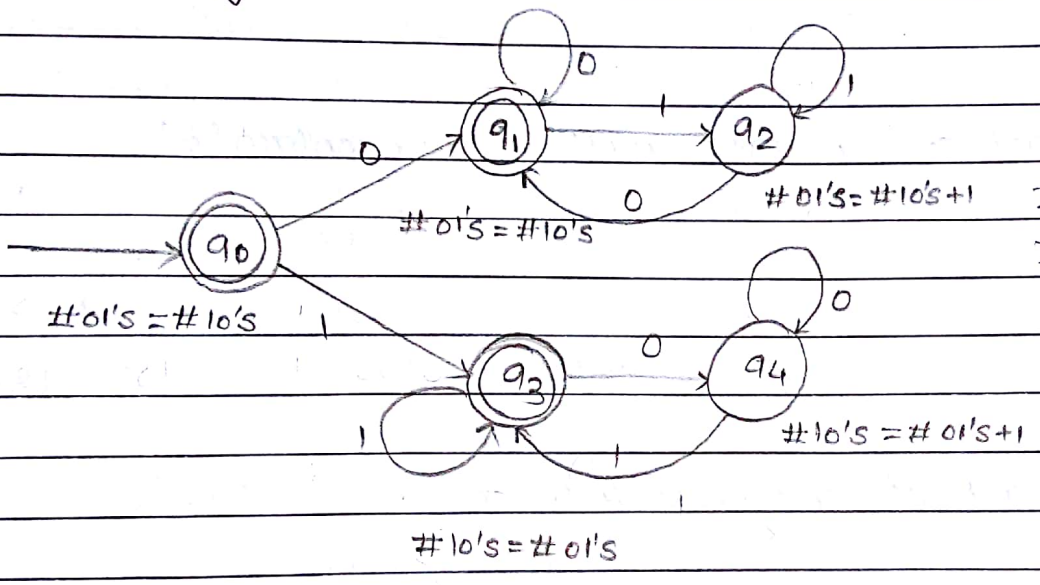
Model 9: Pattern Based

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

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

$\{w \mid n_0(w) = n_1(w),\ w \in (0+1)^*\}$ → impossible to design FA.

valid string = {ϵ, 0, 1, 01̄0, 01̄01, 0100001, 1000010, 101̄111110, ...}



58. $\{w \mid n_0(w) = n_1(w),\ w \in (0+1)^*\}$

Impossible to design finite Automata.

∴ Not-Regular.

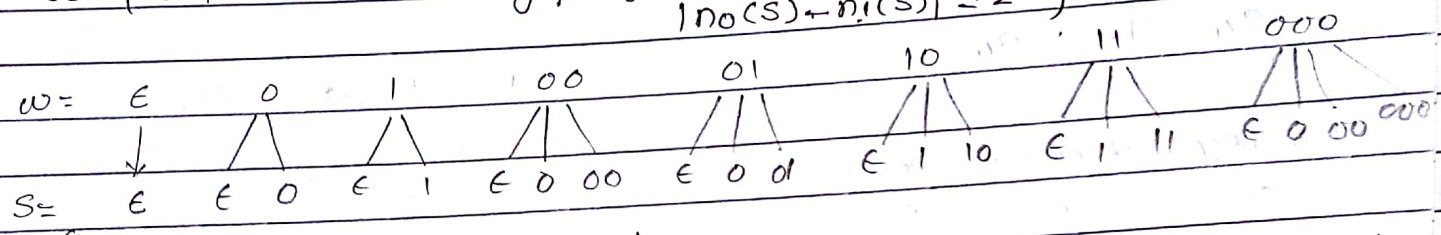
59. $\{w / n_{011}(w) = n_{110}(w),\ w \in (0+1)^*\}$

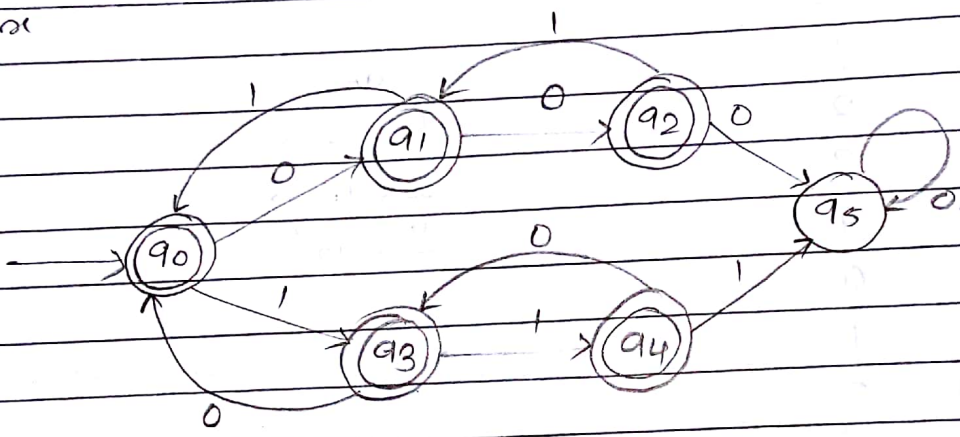
011011, 01100000111, 011111011, 110110, 1100000110

∴ Language is regular.

Model 10: Prefix Based.

60. $\{w / w \in (0+1)^*$, every prefix s of w 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: dead state.

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

$L = \{\in\}$

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

using Q.58

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

$n_0(w) = n_1(w)$ ∴ Not regular,

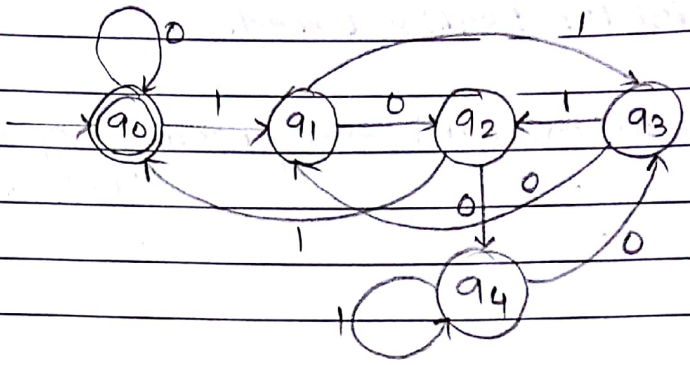
so no finite Automata exist.

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

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

Binary alphabet.

remainder	0	1
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 ⇒ ends with 0 → 2 states
2. Divisible by 4 (2^2) → ends with 00 → 3 states
3. Divisible by 8 (2^3) → ends with 000 → 4 states

n. divisible by 2^n → ends with n zeros → (n+1) state.

divisible by k ($k = 2^n$) → ends with $\log_2 k$ zeros → ($\log_2 k + 1$) states

Decimal equivalent of Binary numbers is divisible by k.

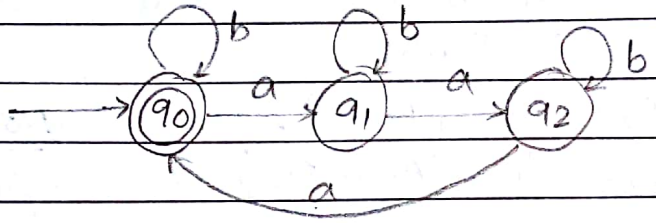
I. If k is in power of 2 $\Rightarrow$ No. of states = $\log_2 k+1$
$k = 2, 4, 8, 16, \ldots\ldots$

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

III. $k \neq 2^{any}$, $k \neq$ prime, $k \neq$ odd then $\Rightarrow$ Apply minimization to answer after constructing with k states.
$k = 6, 10, 12, 14, 18, \ldots\ldots$

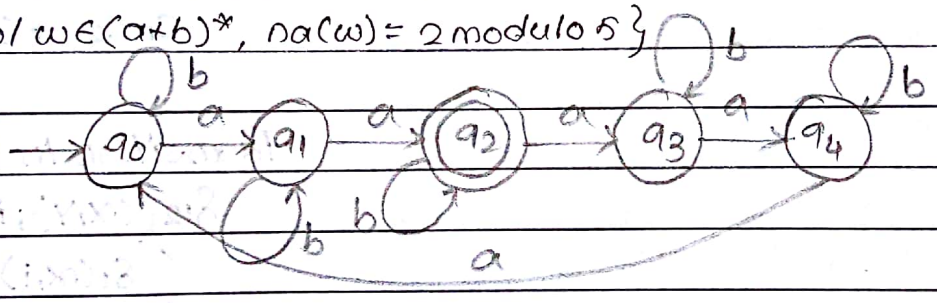
Model 12: Symbol based but divisible

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



3 states

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



5 states.

Model 13: (Many conditions but no condition)

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

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

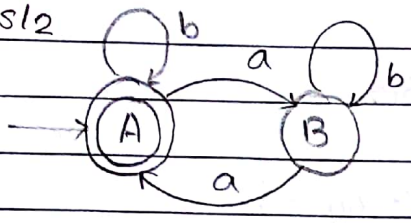
68. $\{w \mid w \in (a+b)^*$, w ends with 'a' or 'b' $\} = (a+b)^+$

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

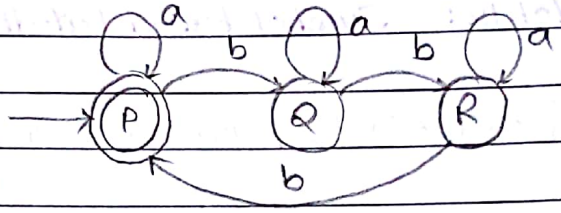
Model 14: Many conditions ⇒ Composition (divide and conquer)

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

FA_1: #a's|2



FA_2: #b's|3

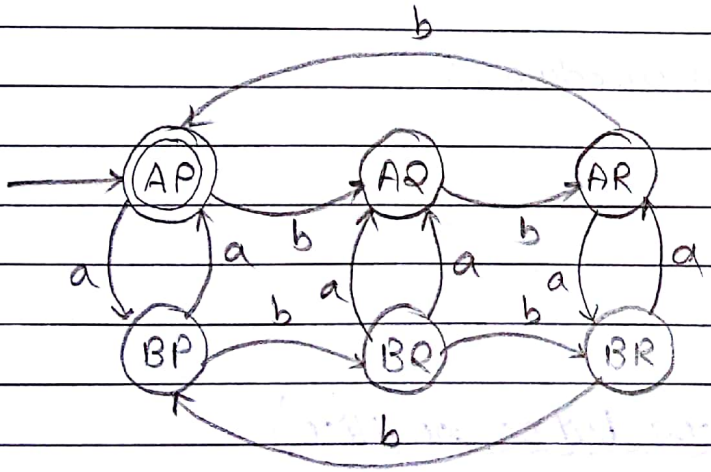


$FA_1 = (Q_1, \Sigma, \delta_1, A, \{A\})$ $Q_1 = \{A, B\}$

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

$FA_2 = (Q_2, \Sigma, \delta_2, P, \{P\})$ $Q_2 = \{P, Q, R\}$.

$FA_1 \times FA_2 = (Q_1 \times Q_2, \Sigma, \delta_{12}, (A,P), F_{12})$



Note:

$C_1 \rightarrow m$ states

$C_2 \rightarrow n$ states

$C_1, C_2 \rightarrow (mn)$ states

Transition in $FA_1 \times FA_2$

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

#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

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

73. #b's%3 but not #a's%2

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

$a^0b^0 \Rightarrow n_a(w) = 0 \text{ modulo } 2$

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

$a^ib^j = n_a(w) = i \text{ modulo } 2$

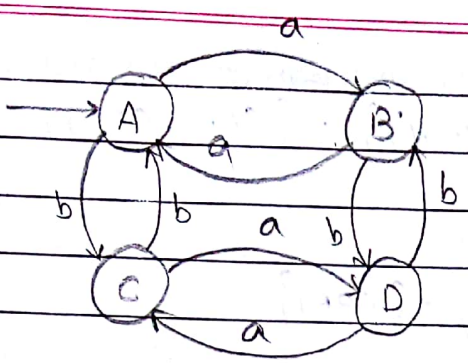
$n_b(w) = j \text{ modulo } 3.$

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

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.

Reg1 ∪ Reg2 = Regular.

FA1 FA2, FA1 × FA2 ∪



A : even a's & even b's
B : odd a's & even b's
C : even a's & odd b's
D : odd a's & odd b's

Minimization of DFA

Requires 'state equivalence' & 'states distinguishable'

complete DFA. → [minimization algo.] → minimized DFA.

- It may reduce no. of states.
- It can save space.
- It works based on "partition".

Partition on a Set

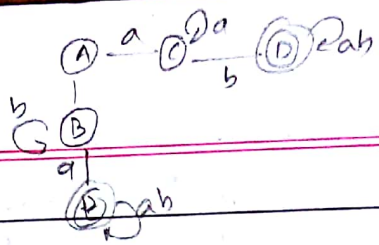
- $A_1 \cup A_2 \cup A_3 \cup \ldots A_n = A$
- Every 2 subsets are disjoint (intersection is empty)

$\{A_1, A_2, A_3, \ldots A_n\}$ is a partition on set A.

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

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

Note. Every partition induces one equivalent relation.



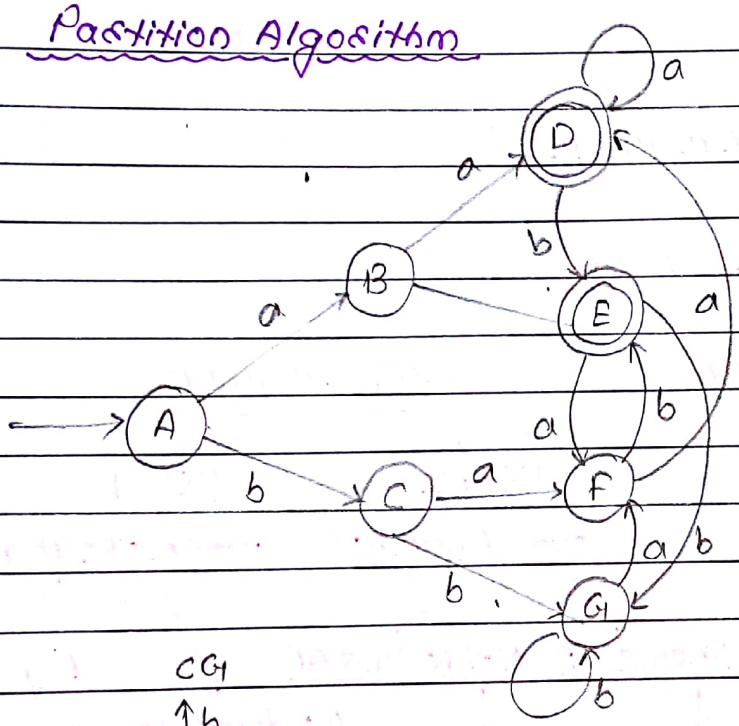
Equivalent States

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: {ABCDEFG} {DE}
|w|=0 set of all non-final set of all final.

Step 2: {ACG} {BF} {D} {E}
|w|=1

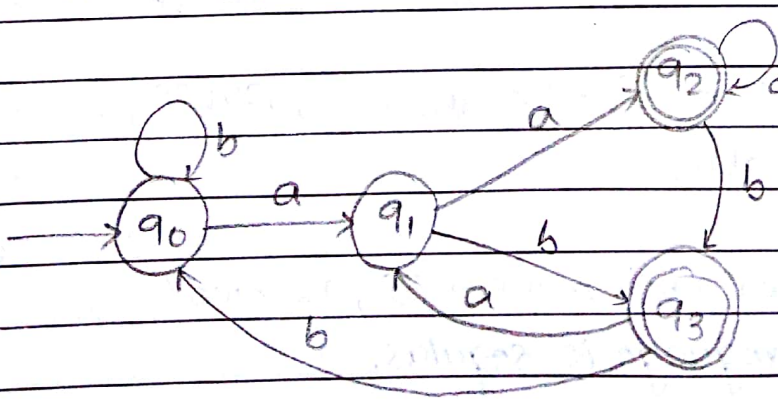
Step 3: {ACG} {BF} {D} {E}
|w|=2 q_0 q_1 q_2 q_3

A, C, G are equivalent.

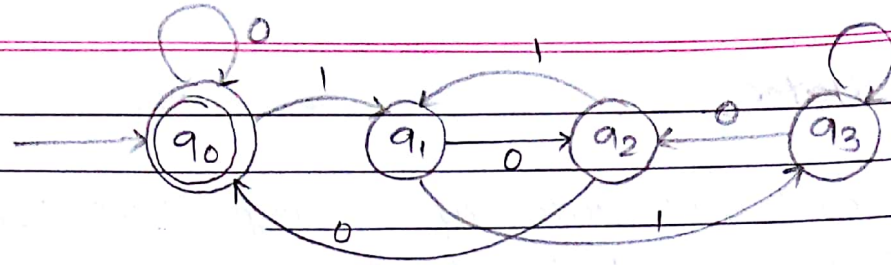
B, F are equivalent.

CG
↑b

AB	AC	AF	AG	CG	DE	BF	EE
↓a	↓a	↓a	↓a		a↓	a↓	
BD	BF	BD	BF		DF.	DD	



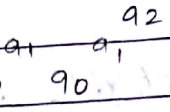
Q.

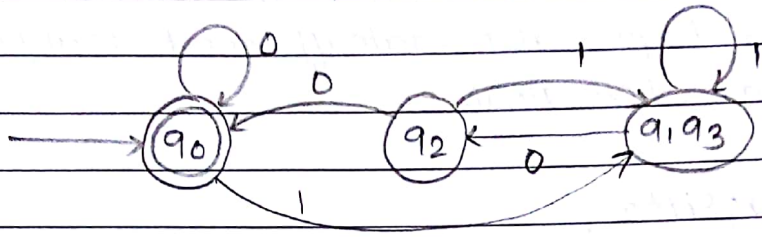


Step 1: $\{q_1 q_2 q_3\}$ $\{q_0\}$

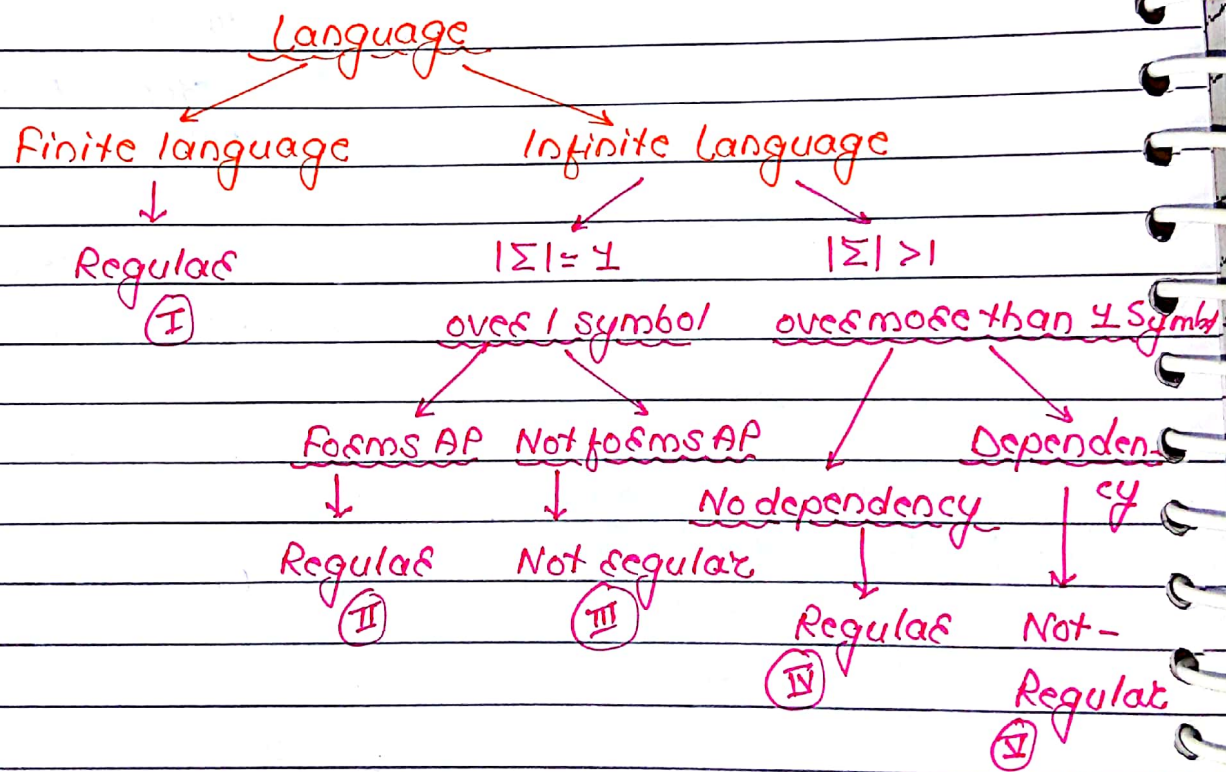
Step 2: $\{q_1, q_3\}$ $\{q_2\}$ $\{q_0\}$

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





Identification of Regular Language



$\{a^n b^n / n < 100\} = \{\epsilon, ab, \ldots\ldots, a^{99} b^{99}\}$

$\{a, abb\}$

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

Note: Every finite language is regular.

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

III. $|\Sigma|=1$ and not form AP.
$a^{n!}$, a^{3n+2^n}, a^{Prime}

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

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

1. $\{a^mb^n\} = a^*b^*$ (IV) → Regular
2. $\{a^mb^n / m=n \text{ or } m\neq n\}$
→ a^*b^* (Regular)
3. $\{a^mb^n / m=n\}$
→ a^nb^n (V) → Not-Regular
4. $\{a^mb^n / m=2n\}$
→ $a^{2n}b^n$ (V) → Not-Regular
5. $\{a^ma b^n\}$
→ a^*ab^* → Regular (a^+b^*) (IV)
6. $\{a^mb^n / m\geq 2, n\geq 1\}$
→ $aaa^*bb^* = aa^+b^+$ (Regular)
7. $\{a^mb^n / m<n, n<100\}$
→ finite. (Regular) (I)
8. $\{a^mb^n / m>n, m<100\}$ → finite → Regular (I)
9. $\{a^mb^n / m=n, m\neq n\} = \phi$ (I) → Regular (I)

10. $\{a^mb^n / LCM(m,n)=1\}$ → $\{ab\}$ → Regular (I)

11. $\{a^mb^n / gcd(m,n)=1\}$ = (IV) → Not-Regular.

12. $\{a^mb^n / LCM(m,n)<100\}$ = IV- Regular
13. $\{a^mb^n / LCM(m,n)=0\}$ — IV. Regular
14. $\{a^mb^n / 0<LCM(m,n)<100\}$ = I. Regular
15. $\{w_1,w_2 / w_1,w_2\in(a+b)^*\}$ → Regular

$(a+b)^*(a+b)^* = (a+b)^*$ $w_1 = w_2$ IV. Reg

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

17. $\{w_1 w_2 \mid w_1 = w_2,\ w_1, w_2 \in (a+b)^*\}$ V. 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
$|w| = 0 \bmod 3$.

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

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

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

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

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

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

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

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

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

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

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

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

32. $\{wwx / w, x \in (a+b)\}$ I. Regulas

33. $\{wwx / w, x \in a^*\}$ a^* Regulas

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

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

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

37. $\{xww / w \in (a+b)^*, x \in (a+b)^+\}$ Regulas

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

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

40. $\{xww / w \in (a+b), x \in (a+b)^*\}$ $xaa+xbb$ ∴ Regulas

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

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

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

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

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

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

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

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

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

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

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

52. $\{wwR x \mid w, x \in (a+b)^*\}$ $x = (a+b)^*$. ∴ Regular

53. $\{wwR x \mid w, x \in (a+b)^+\}$ ∴ Not Regular

54. $\{wwR x \mid w \in (a+b)^*, x \in (a+b)^+\}$ $x = (a+b)^+$. ∴ Regular

55. $\{wwR x \mid w \in (a+b)^+, x \in (a+b)^*\}$ Not-Regular

56. $\{wwR x \mid w \in (a+b)^*, x \in (a+b)\}$ Not-Regular

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

58. $\{wwR x \mid w, x \in a^*\}$ Regular

59. $\{wwR x \mid w, x \in a^+\}$ Regular

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

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

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

63. $\{wxwR \mid w \in (a+b)^+, x \in (a+b)^*\}$ axa + bxb (atleast two length)
∴ Regular

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

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

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

∴ $w=aa$ ∴ $aaxaa$

$w=ab$ $abxba$ ∴ $axa+bxb = a(a+b)^+a +$

$w=ba$ $baxab$ $b(a+b)^+b$

$w=bb$ $bbxbb$ ∴ Regular

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

70. $\{xwwR \mid w,x\in(a+b)^+\}$ xaa+xbb

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

72. $\{xwwR \mid w\in(a+b)^+,\ x\in(a+b)^*\}$ xaa+xbb NOT.

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

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

75. $\{xwwR \mid w,x\in a^*\}$ R

76. $\{xwwR \mid w,x\in a^+\}$ R

77. $\{xwwRy \mid w,x,y\in(a+b)^+\}$ $xaay+xbby \Rightarrow$ Regular

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

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

$wxwy$ $xwyw$
(a) (z)
$axay + bxby$ $xayb + xbyb$

80. $\{wxwy \mid w,x,y \in (a+b)^*\}$ $axay + bxby$ ∴ Regulas

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

82. $\{xwywz \mid w,x,y,z \in (a+b)^+\}$ ∴ $xayaz + xbybz$
∴ Regulas

83. $\{wxwRyw \mid w,x,y \in (a+b)^*\}$ $(a+b)^*$
∴ Regulas

84. $a^{n!}$ $\{a^1, a^2, a^6, a^{24}, \ldots\}$
∴ Not-Regulas

85. a^{Psime} $\{a^2, a^3, a^5, \ldots\}$ NR.

86. a^{2n} $\{a^0, a^2, a^4, \ldots\}$ AP → Regulas

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

88. $\{a^{mn} \mid m,n \geq 1\} \rightarrow \{a^1, a^2, a^3, \ldots\}$
Regulas

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

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

91. $(a^{n!})^*$ a^* → Regulas

92. $(a^{Psime})^* = \Sigma^* - \{a\}$
"Complement of a" $\{a^n \mid n \neq 1\}$ → Regulas

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

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

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

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

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

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

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

100. $g^m (a^{2^n})^* t^{2^{nk}} (e^{Psime})^* R$

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

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

If L is Regulas then L^* is also Regulas.

$(a^{Prime})^* = (a^2)^* \cup (a^3)^* \cup (a^5)^* \cup (a^7)^* \ldots\ldots$

$= \{\epsilon, a^2, a^3, a^4, a^5, a^6, \ldots\ldots\}$

$= \Sigma^* - \{a\}$

= complement of $L = \{a\}$ $= \overline{\{a\}}$

$= \{a^n \mid n \neq 1\}$

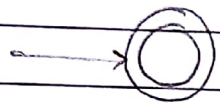
$= \{a^n \mid n = 0 \text{ or } n \geq 2\}$

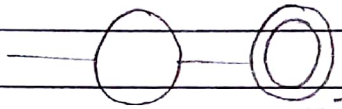
$= \epsilon + aaa^*$

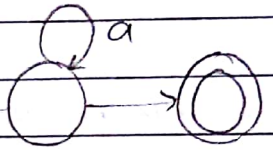
$= \epsilon + aa^+$

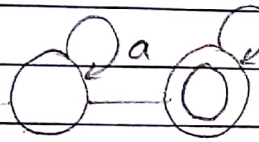
Understanding –

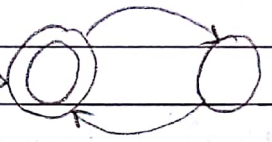
Non-Deterministic finite Automata with ϵ-moves :

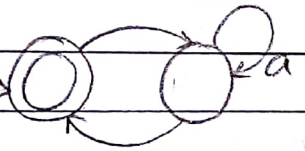
1. 

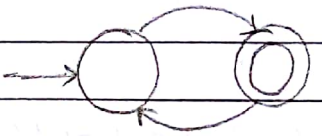
2. 

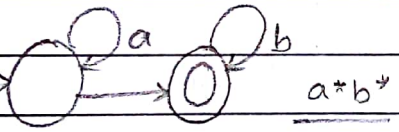
3. 

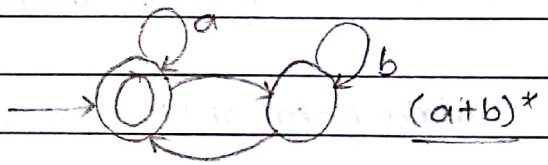
4. 

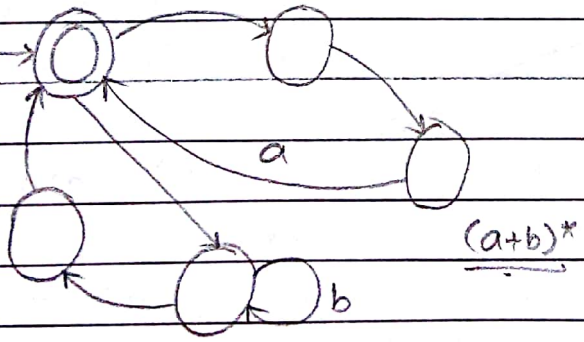
5. 

6. 

7. 

8. 

9. 

10. 

Application of ϵ-NFA:

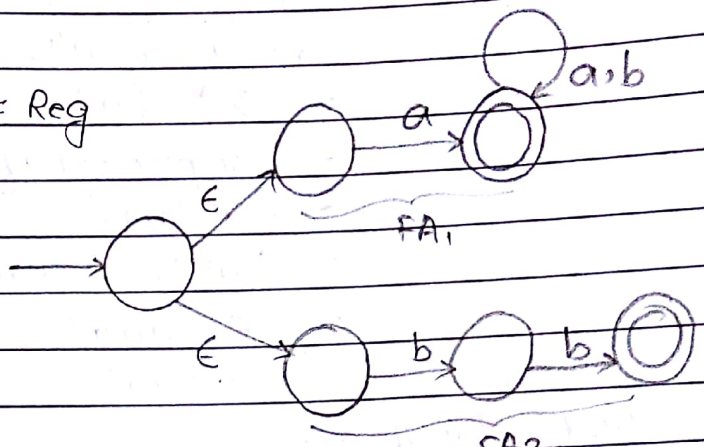
01. Union $\quad Reg_1 \cup Reg_2 = Reg$

$Reg_1 \rightarrow FA$

$L_1 = a\alpha$

$Reg_2 \rightarrow FA$

$L_2 = \chi bb$



02. Concatenation:

$Reg_1 . Reg_2 = Reg$

03. Kleene closure

$(Reg)^* = Regular$

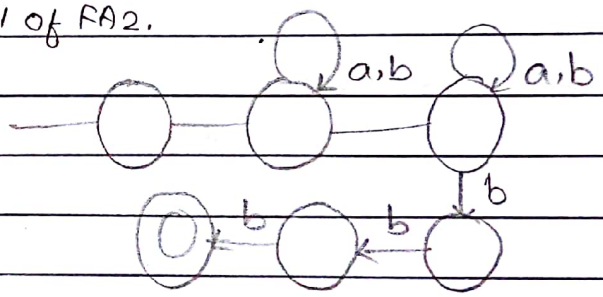
04. positive closure

$(Reg)^+ = Regular$

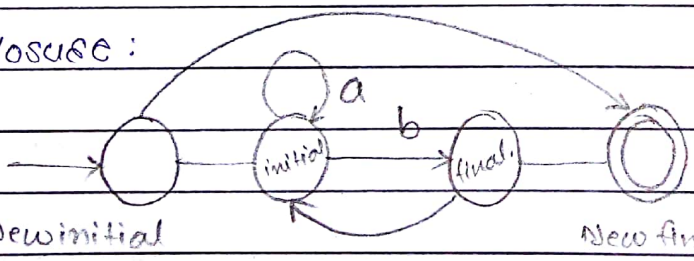
Concatenation:

i. Give ϵ-move from final of FA1 to initial of FA2.

ii. Make final of FA1 as non-final.



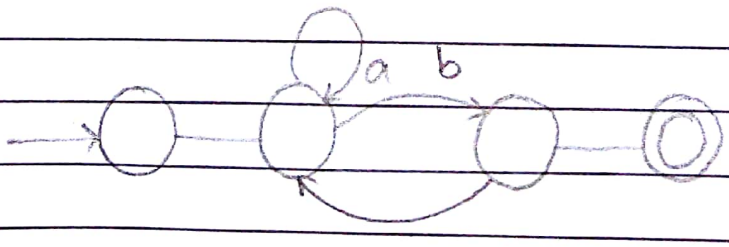
Kleene closure:



$L = a+b$

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

Positive closure:



Conversions

01. NFA ⟶ DFA (Subset construction)
02. ϵ-NFA ⟶ NFA
03. ϵ-NFA ⟶ DFA (Subset construction)

04. DFA ⟶ NFA
05. NFA ⟶ ϵ-NFA
06. ϵ-NFA ⟵ DFA

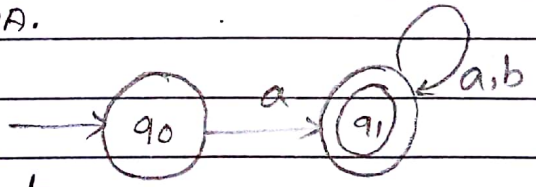
} "by definition"

DFA ≅ NFA ≅ ϵ-NFA

Conversion from NFA to DFA (Subset Construction)

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

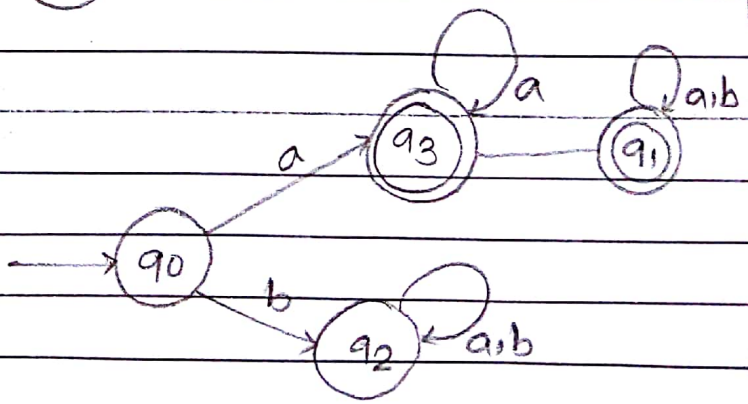
i. Initial (DFA) is initial (NFA).
ii. $\delta_D(\{P,Q\}, i) = \delta_N(P,i) \cup \delta_N(Q,i)$
iii. If any subset (state) contain final of NFA, make that subset as final in DFA.



δ_N	a	b
→ A q_0	{A,B}	∅
* B q_1	{B}	{B}

DFA: q_3

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 =
  $\log_2 n \leq$ No. of states in min NFA $\leq n$
- If min NFA has n states, No. of states in min. DFA =
  $1 \leq$ 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:

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

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

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

Convession from ϵ-NFA to NFA

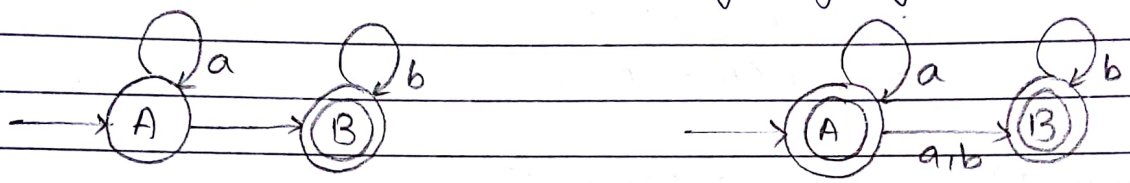
i. No. of states in NFA = No. of states in ϵ-NFA.

ii. Initial state (NFA) is same as initial (ϵ-NFA).

iii. $\delta_N(P, i) = \epsilon\text{-closure}\left(\delta_\epsilon(\epsilon\text{-closure}(P), i)\right)$

iv. If any ϵ-clo(q) contain final of ϵ-NFA then make 'q' as final in NFA.

ϵ-closure(q) : It is set of states whose each state is reachable from q without reading any symbol.



$\delta_\epsilon(A, a) = A$
$\delta_\epsilon(A, b) = \phi$
$\delta_\epsilon(B, a) = \phi$
$\delta_\epsilon(B, b) = B$

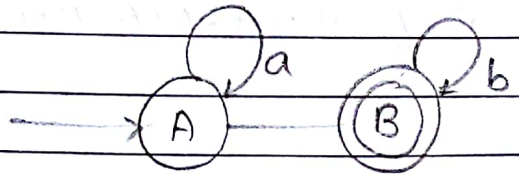
ϵ-closure(A) = {A, B}
ϵ-closure(B) = {B}

$\delta_N(A, a) = \epsilon\text{-clo}(\delta_q(\epsilon\text{-clo}(A, a))$
$= \epsilon\text{-clo}(\delta_\epsilon\{A, B\}, a))$
$= \epsilon\text{-clo}(\delta_\epsilon(A, a) \cup \delta_\epsilon(B, a))$
$= \epsilon\text{-clo}(\{A\} \cup \phi)$
$= \epsilon\text{-clo}(A)$
$= \{A, B\}$

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

Conversion from ϵ-NFA to DFA

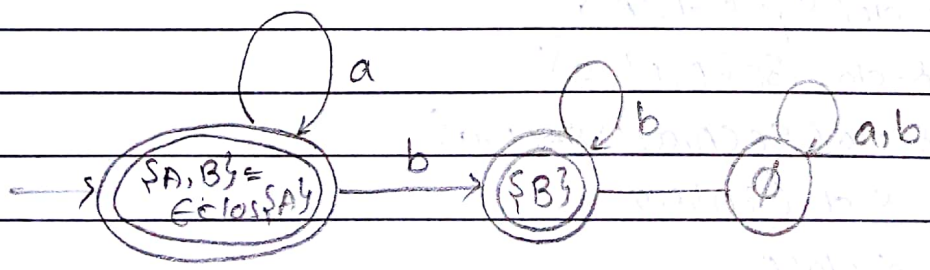
- Initial state (DFA) = ϵ-closure (Initial of ϵ-NFA)
- $\delta_D(P,i) = \epsilon\text{-clo}(\delta_\epsilon(P,i))$
- If any state contain final of ϵ-NFA, then make it as final in DFA.



$\delta_D(\{A,B\},a) = \epsilon\text{-clo}(\delta_\epsilon(\{A,B\},a))$
$= \epsilon\text{-clo}(\{A\} \cup \phi)$
$= \epsilon\text{-clo}\{A) = \{A,B\}$

$\delta_D(\{A,B\},b) = \epsilon\text{-clo}(\delta_\epsilon(\{A,B\},b))$
$= \epsilon\text{-clo}(\phi \cup \{B\})$
$= \epsilon\text{-clo}(B) = \{B\}$

$\delta_D(\{B\},a) = \epsilon\text{-clo}(\delta_\epsilon(\{B\},a))$ $\quad \delta(\{B\},b] = \{B\}$
$= \epsilon\text{-clo}(\phi) = \phi$



Regular Grammars

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

V → set of variables (non-terminals)

T → set of terminals

P → set of rules / productions

S → start symbol. $S \in V$

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

Eg. $S \to ABb$

$A \to aA / b$

$B \to bBa / ab$

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

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

$S = S$

- Regular Grammar → It is LLG or RLG.

Left Linear Grammar

$V \to VT^* / T^*$ → atmost one non-terminal (at left most)

↳ Exactly one non-terminal.

Eg. $S \to Aa / bcd$

$A \to \epsilon$

Right Linear Grammar

$V \to T^*V / T^*$

↳ atmost one non-terminal but at right most position.

Eg. $S \to abS / efg$.

1. $S \to abc$ LLG1
RLG1
Regulas Grammas.

2. $S \to A$ LLG.
$A \to ab$ RLG
∴ Regulas Grammas

3. $S \to Sa|Sbc|\epsilon$
only LLG1.

4. $S \to aS|bS|ab$
only RLG1.

5. $S \to aS|Sb|a$
Not RLG, not LLG.

Identify language generated by the following Regulas Grammas:

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

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

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

4. $S \to AB$ (Not-Reg.G1)
$A \to a, B \to b$
$L = \{ab\}$

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

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

- Regulas language always contain regulas grammas but non-regulas grammas can also generate regulas language.

- Every regulas-grammas generates regulas language.

Not. Grammar.

6. $S \rightarrow Sa|b$ — $L = \{b, ba, baa, \ldots\}$ — 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. $S \rightarrow aS|\varepsilon$ — a^*

10. $S \rightarrow Sa|\varepsilon$ — a^*

11. $S \rightarrow aS|a$ — a^+

12. $S \rightarrow aS|b$ — a^*b

13. $S \rightarrow Sa|a$ — a^+

14. $S \rightarrow Sa|b$ — ba^*

15. $S \rightarrow aaS|\epsilon$ — $(aa)^*$

16. $S \rightarrow Saa|\epsilon$ — $(aa)^*$

Equivalence

- Regulas Expression $\cong$ Finite Automata
- Finite Automata $\cong$ Regulas grammas.

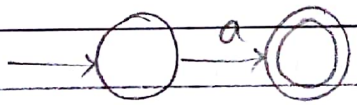
} R.E $\cong$ Reg Grammar

Convession: 1. RE $\rightarrow$ FA 2. FA $\rightarrow$ RE

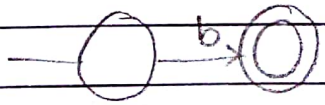
Convession from Regulas Expression to finite Automata.

(Induction Method)

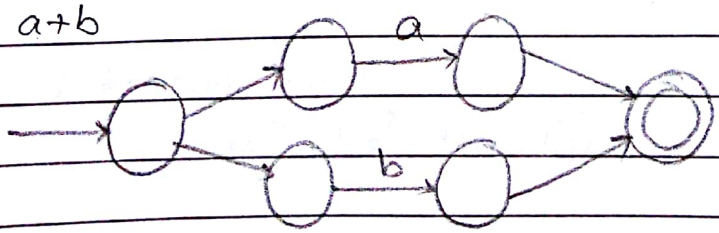
1. a



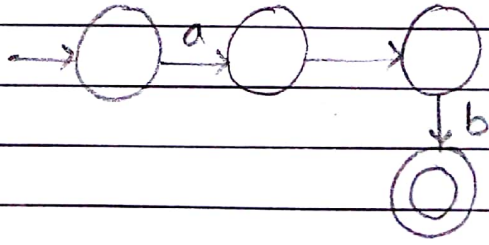
2. b



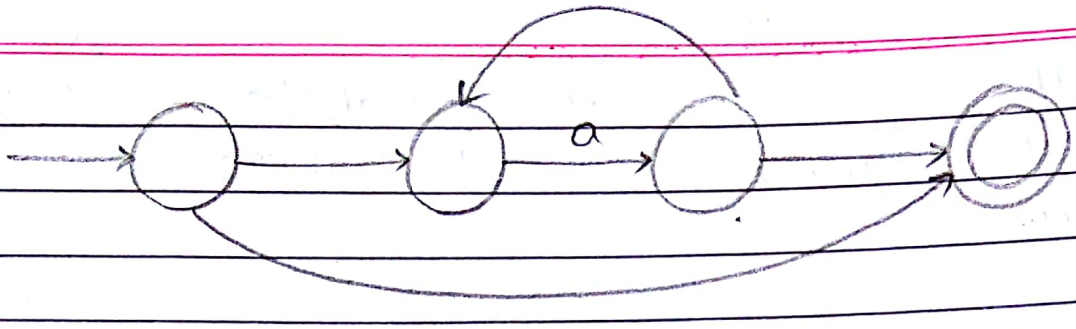
3. a+b



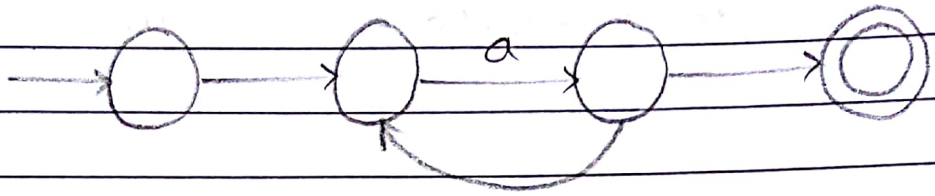
4. a.b.



5. a*

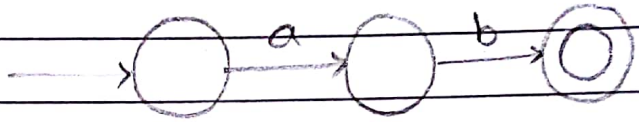


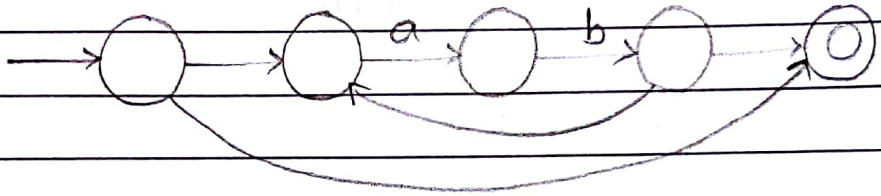
6. a+



Q.1. (ab)*

ab:





2. ((a*b)+.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.
- $O(n^3)$

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

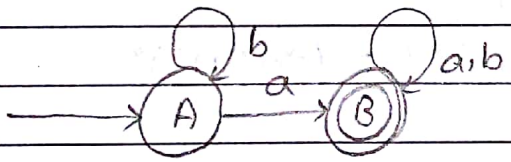
Arden's method:

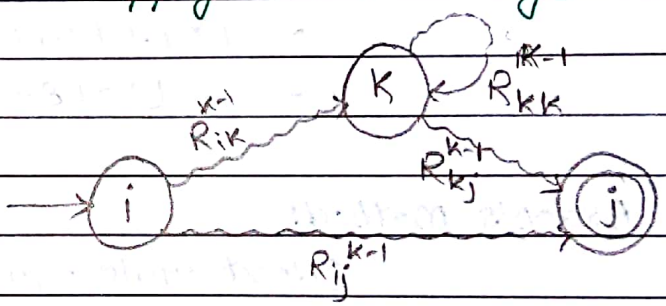
- state equation based
- Every, state, we compute equation.
- Final state will decide Regular Expression.

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

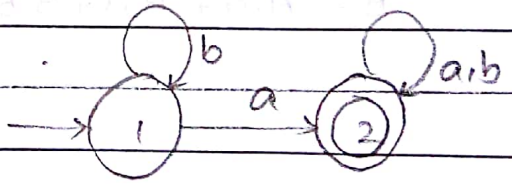
Note: If P contain ε then it may have infinite solutions

(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) ← no. of states

3. Take base cases:

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

$R_{11}^{0} = b + \varepsilon$ $\quad R_{21}^{0} = \phi$

$R_{12}^{0} = a$ $\quad R_{22}^{0} = a + b + \varepsilon$

$$R_{12}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$$

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

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

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

$$R_{12}^{1} = R_{12}^{0} + R_{11}^{0} (R_{11}^{0})^* R_{12}^{0}$$

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

$$= a + b^* a$$

$$= b^* a \quad \text{---(2)}$$

$$R_{22}^{1} = R_{22}^{0} + R_{21}^{0} (R_{11}^{0})^* R_{12}^{0}$$

$$= (a+b+\varepsilon) + \phi$$

$$= a+b+\varepsilon \quad \text{---(3)}$$

Substitute (2) & (3) in 1

$$R_{12}^{1} = R_{12}^{1} + R_{12}^{1} (R_{22}^{1})^* R_{22}^{1}$$

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

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

$$= b^* a [\varepsilon + (a+b)^*] = b^* a (a+b)^*$$

Arden's Method:

(Next state equation)

$A = \varepsilon + A.b \longrightarrow A = \varepsilon . b^* = b^*$

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

$R = Q + RP \quad P.S.$

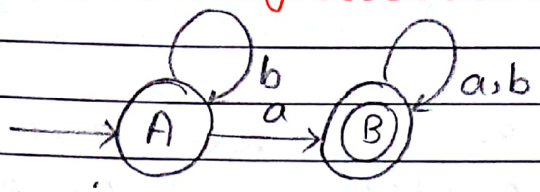
$B = (b^* a)(a+b)^*.$

Equivalence between Regular Grammar & Finite Automata.

1. FA $\Rightarrow$ RLG
2. RLG $\Rightarrow$ FA
3. FA $\Rightarrow$ LLG
4. LLG $\Rightarrow$ FA
5. RLG $\Rightarrow$ LLG
6. LLG $\Rightarrow$ RLG.

FA $\simeq$ LLG $\simeq$ RLG,

Finite Automata to Right Linear Grammar



P.S I/P N.S P.S I/P N.S

$A \rightarrow bA$ $A \rightarrow aB$

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

(∵ B is final state add null production)

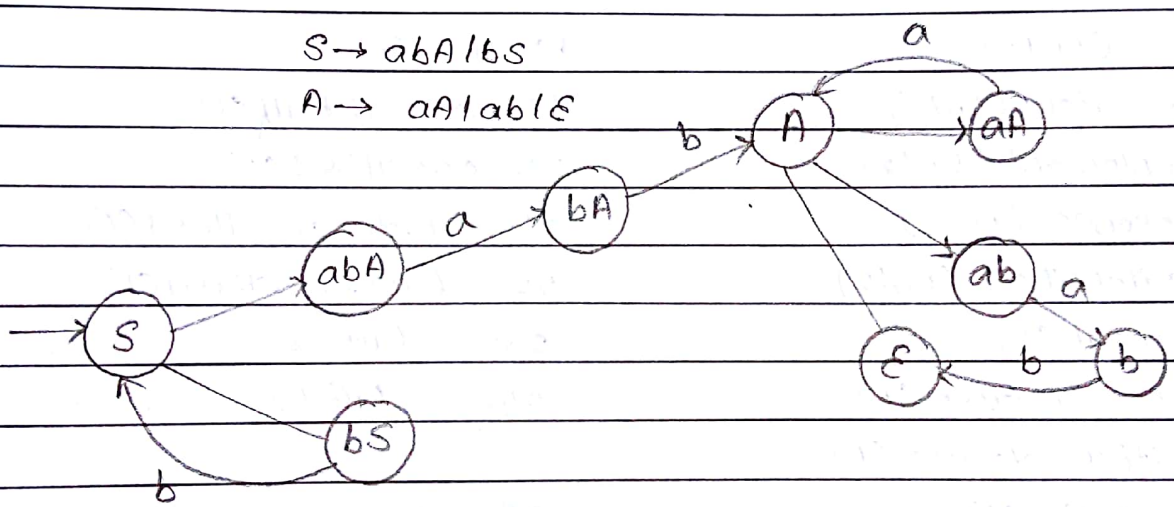
∴ $A = bA \mid aB$, $B = aB \mid bB \mid \epsilon$ } with null production

$A = bA \mid aB \mid a$, $B = aB \mid bB \mid a \mid b$ } without null production

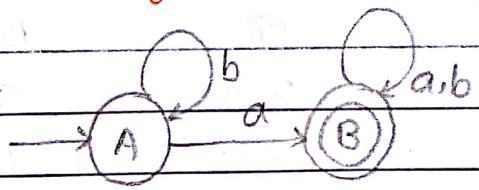
Right Linear Grammar to Finite Automata

$S \rightarrow abA \mid bS$

$A \rightarrow aA \mid ab \mid \epsilon$



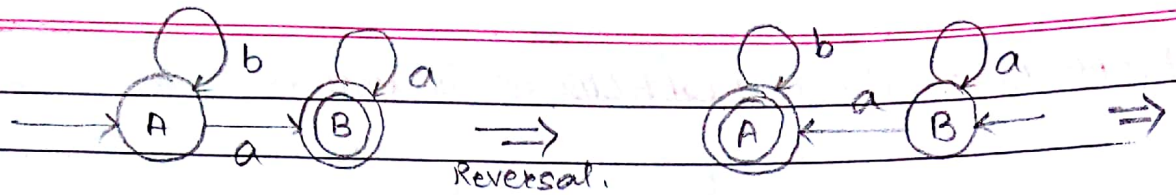
Finite Automata to Left Linear Grammar.



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

Reversal :- Final = Initial

Initial = Final. Reverse every edge.

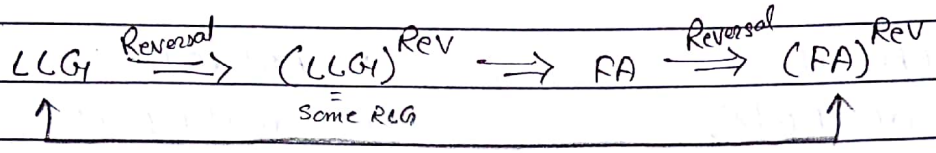


$B \to aB \mid aA \xrightarrow{\text{Reversal}} B \to Ba \mid Aa$

$A \to bA \mid \varepsilon \xrightarrow{\text{Reversal}} A \to Ab \mid \varepsilon.$

(Every RHS)

Left Linear Grammar to Finite Automata :



Closure Properties of Regular languages:

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

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

} not closed.

Union

Regular ∪ Regular = Regular

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

} Proofs.

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

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

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

$Reg_1 \Rightarrow FA_1$
$Reg_2 \Rightarrow FA_2$
} $FA_1 \times FA_2 \Rightarrow Reg_1 \cup Reg_2$

• $L_1 = \emptyset$, $L_2 =$ any
$L_1 \cup L_2 = L_2$

• $L_1 = \Sigma^*$, $L_2 =$ Any
$L_1 \cup L_2 = \Sigma^*, L_1$.

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

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

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

L_1 = Non-Regular L_2 = Non-Regular
$L_1 \cup L_2$ = may be or may not be regular

Intersection

Regular ∩ Regular = Regular

- Use compound FA

- Regulas languages closed undes intessection.
- Intessection is closed fos segulas languages.
- (set of all segulass, ∩) is closed.
- ({L|L is segulas} , ∩) is closed.

Eg. $L_1 = a^*$ $L_2 = b^*$ $L_1 \cap L_2 = \varepsilon$

$L_1 = a^+, L_2 = b^+$ $L_1 \cap L_2 = \phi$

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

$L_1 = \phi$ L_2 = any $L_1 \cap L_2 = L_1$

$L_1 = \Sigma^*$, L_2 = Any $L_1 \cap L_2 = L_2$

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

L_1 = Regulas L_2 = Not-Regulas
$L_1 \cap L_2$ = may os may not be segulas

Complement

Regulas = Regulas

Psoof: Regulas ⇒ DFA ⟶ modified DFA ⇒ Regulas
(Inteschange finals & non-final)

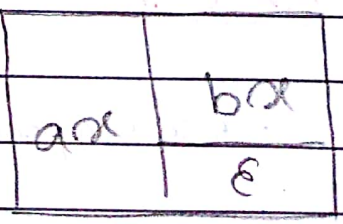
Eg.
L = Set of all stsings whese each stsing not containing 'aaa' as substsing.

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

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

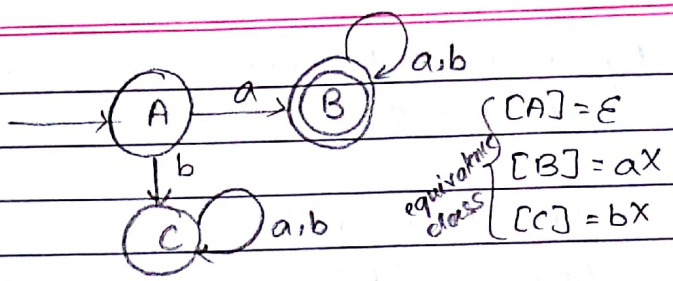
- $L = \phi \Rightarrow \bar{L} = \Sigma^*$
- $L = \Sigma^* \Rightarrow \bar{L} = \phi$
- $L = a(a+b)^* \Rightarrow \bar{L} = b(a+b)^* + \varepsilon$

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



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

  $\overline{L} = \varepsilon + (a+b)^*a$



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. $L_1 - L_2 = L_1 \cap \overline{L_2}$
3. $L_1 - L_2 = \overline{(L_1 \cap \overline{L_2})} = \overline{L_1} \cup L_2$

$Reg_1 - Reg_2 = Reg_1 \cap \overline{Reg_2}$

$= Reg_1 \cap Reg = Regular$

Eg.

- $L_1 = a^*, L_2 = b^*$

  $L_1 - L_2 = a^+$

  $L_2 - L_1 = b^+$

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

  $L_1 - L_2 = \phi$

  $L_2 - L_1 = \Sigma^*$

- $L_1 = \phi, L_2 = any$

  $L_1 - L_2 = \phi$

  $L_2 - L_1 = any$

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

  $L_1 - L_2 = \phi$

  $L_2 - L_1 = \overline{L_1}$ $(\Sigma^* - L_2 = \overline{L_2})$

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

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

  $L_2 - L_1$ = ——— " ———

- $\overline{\text{Non-regular}} = \text{Non-regular}$
- $\overline{\text{Regular}} = \text{Regular}$

Concatenation

$$Reg_1 . Reg_2 = Regular$$

Proof: 1. Use ε-NFA

2. Use RE.

Eg. • $L_1 = \phi$, L_2 = any
$L_1 \cdot L_2 = \phi$ $L_2 \cdot L_1 = \phi$

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

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

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

Reversal

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

Proof:
- Reg $\Rightarrow$ finite automata $\Rightarrow$ (finite automata)^{Rev} $\Rightarrow$ (Reg)^{Rev}
- Reverse each RHS, in regular grammar
$S \to Sa|b \Rightarrow ba^*$

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

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

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

Kleene Closure

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

Proof:
- use ε-NFA
- use Regular expression

Eg. • $L = (a+b)^*$ $L^* = L$
• $L = a^*$ $L^* = L$
• $L = \phi$ $L^* = \{\varepsilon\}$
• $L = a^*b^*$ $L^* = (a+b)^*$

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

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

Positive Closure

$(Regulas)^+ = Regulas$

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

• $L = a^*$ $L^+ = L$

• $L = \phi$, $L^+ = \phi$

• $L = a^*b^*$ $L^+ = (a+b)^*$

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

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

• $L = a+b$ $L^+ = (a+b)^{*+}$

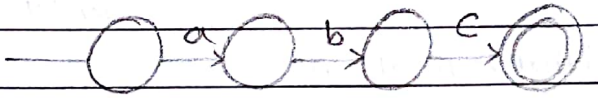
• $L = \{\varepsilon\}$ $L^+ = L$

Prefix:

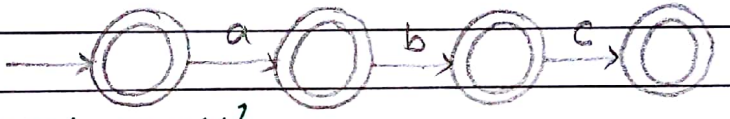
$w = abc$

prefix(w) = ε, a, ab, abc (sequence of beginning symbols)

$L = \{abc\}$



from initial to final path make all states as final.



$w = abc$

$= uv$

prefix = ε abc

= a bc

= ab c

= abc ε

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

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

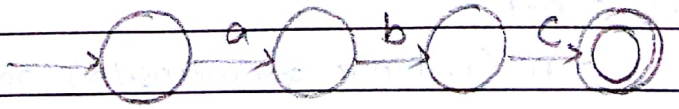
no. of proper prefixes $= n$

Suffix

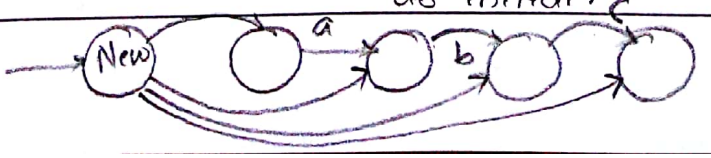
$w = abc$

suffix(w) = ε, c, bc, abc (sequence of ending symbols)

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



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



$w = abc$

$= uv$ (suffix)

$= \varepsilon abc$

$= a\,bc$

$= ab\,c$

$= abc\,\varepsilon$

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

$|w|$ = no. of suffixes $= n+1$

no. of proper suffixes $= n$

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

Substring

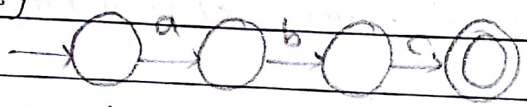
string $(w) = aaa$

substring $\Rightarrow$ ε, a, aa, aaa

string $(w) = abc$

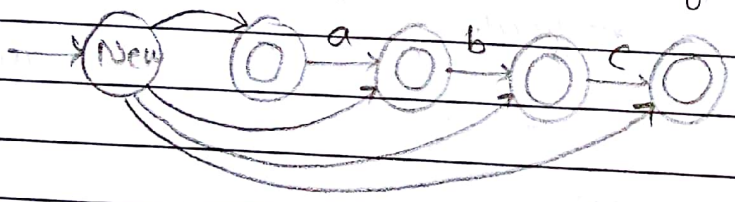
substring $\Rightarrow$ $\varepsilon, a, b, c, ab, bc, abc$

$L = \{abc\}$



from initial to final path

every state is initial & final.



string $(w) = abc = xyz$

substrings of 'abc' ε :-

a	ε a bc
b	a bc
c	ab c ε
ab	ε abc
bc	a bc ε
abc	ε abc ε

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

If $|w| = n$

$n+1 \leq$ no. of substrings $\leq \Sigma n+1$

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

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

$= n$ (exclude zero length) of non-empty substring of different length.

$w = abc$

substring(w) = ε, (a, b, c), (ab, bc), abc

4 different lengths

Subsequence

$w = aaa$

subsequence(w) = ε, a, aa, aaa

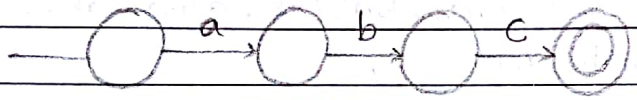
$w = abc$

subsequence(w) = $\varepsilon, a, b, c, ab, bc, ac, abc$

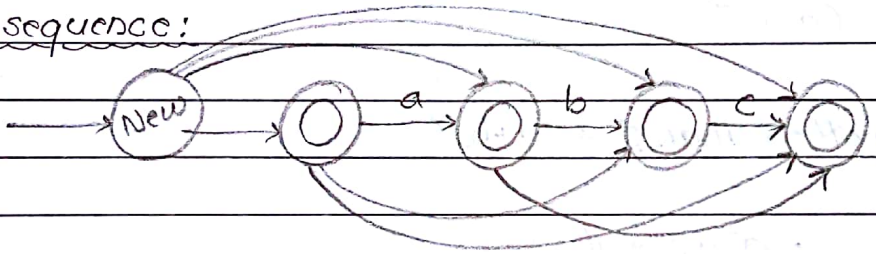
If $|w| = n$,

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

$L = \{abc\}$



Subsequence:



Quotient

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

$L_1 = \{a, ab, aac\}$ $\quad L_2 = \{\varepsilon, a, b, c\}$

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

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

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

$Any/\varepsilon = Any$ $\quad ab/\varepsilon = ab$

$a/\varepsilon = a$

$aa/\varepsilon = aa$

- $a^*/a = \{\varepsilon/a, a/a, aa/a, aaa/a, \ldots\}$
  $= \{\varepsilon, a, a^2, a^3, \ldots\}$
  $= a^*$

- $a^*a/a = \frac{a^*a}{a} = a^*$

- $a^+/a = a^+$

- a^*b/ab^* (HW)

- $(a+b)^*/a^* = (a+b)^*/\varepsilon$
  $= (a+b)^*$

- $(a+b)^*/ab = \{\varepsilon.ab/ab, a.ab/ab, bab/ab, aa.ab/ab, \ldots\}$
  $= \{\varepsilon, a, b, aa, \ldots\}$
  $= (a+b)^*$

Note:

$(a+b)^*/$ any other than $\phi = (a+b)^*$

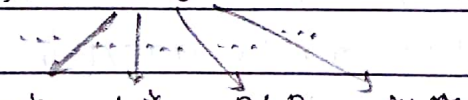
- $L/\phi = \phi$
- $a^+/b^+ = \phi$

Subset

Subset (Regular) = need not be Regular

∴ subset property is not closed.

Given: a^*b^*



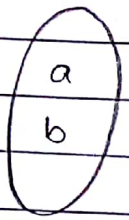
$a^*\ldots b^*$ a^nb^n a^*b^*

Note:

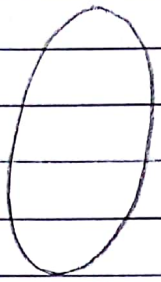
- Subset operation is not closed for all types of formal languages.
- Subset of finite language is finite. (so regular)

Substitution for Regulas languages (Regulas Substitution)

$\Sigma = \{a, b\}$ $\quad \Sigma^* = \{\epsilon, a, b, ab, ba, \ldots\}$ $\quad 2^{\Sigma^*} = P(\Sigma^*)$

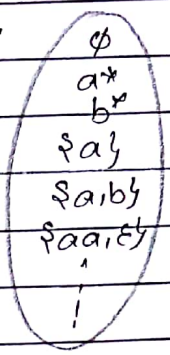


set of symbols



set of strings

$= \{$



set of languages

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

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

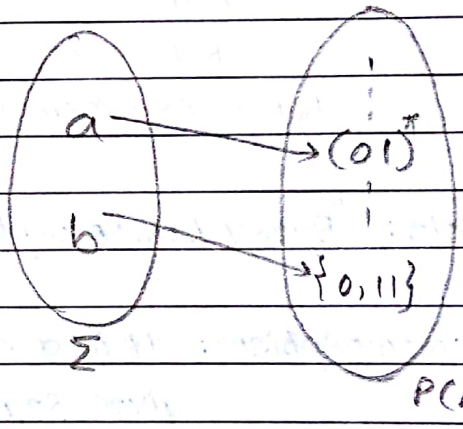
Given: $L = a^*b^*b$ (Regulas)

$f(a) = (01)^*$ Regulas $\quad \Sigma = \{a, b\}$

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

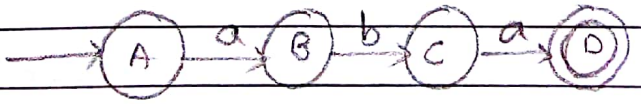
$f(L) = (f(a))^* (f(b))^* f(b)$

$= (01)^*)^* (0+11)^* (0+11)$

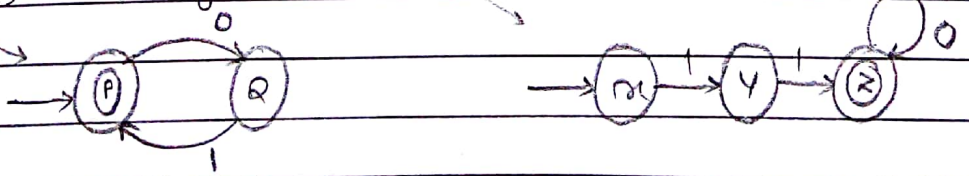


Substitution : It is a function where each input symbol of given regulas language over Σ is substituted with some regulas over Δ.

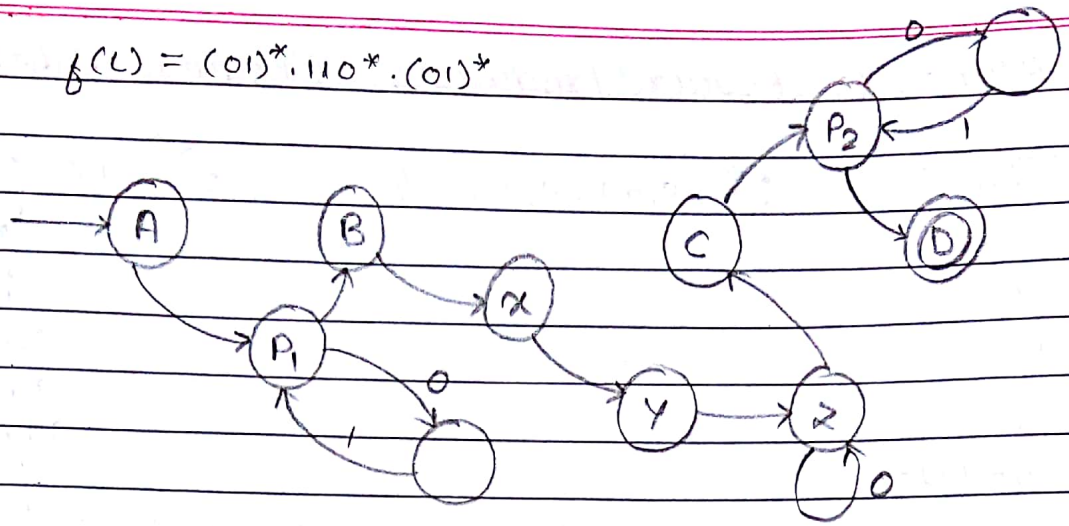
$L = \{aba\}$



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



$f(L) = (01)^*.110^*.(01)^*$



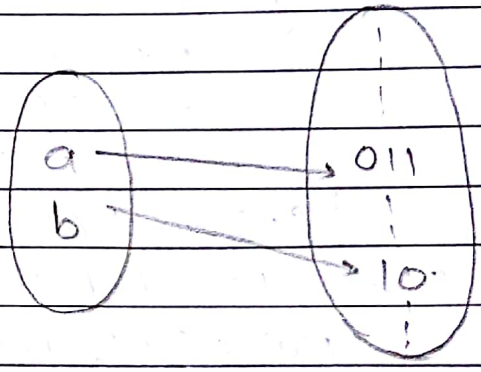
Homomosphism for Regulas languages (string substitution)

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

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

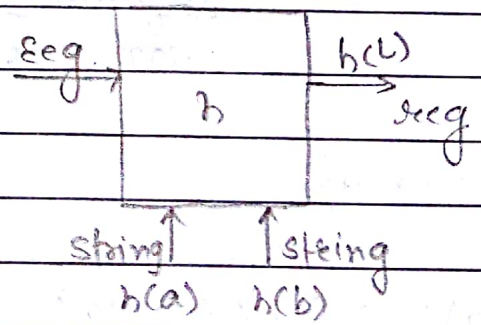
Substitutions with strings: $h(a) = 011$, $h(b) = 10$, $\Delta = \{0,1\}$

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



Note: "Every homomosphism is substitution."

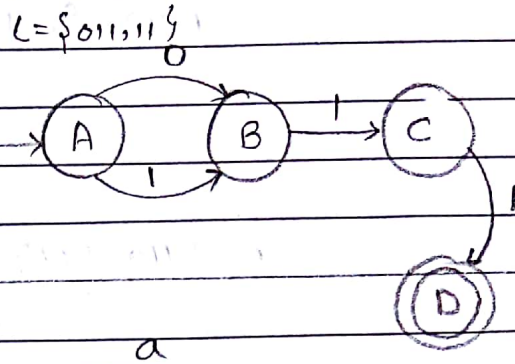
Homomosphism: It is a substitution where each input symbol of given regulas language over Σ is substituted with some string over Δ.



Inverse Homomorphism

$L = \{011, 11\}$

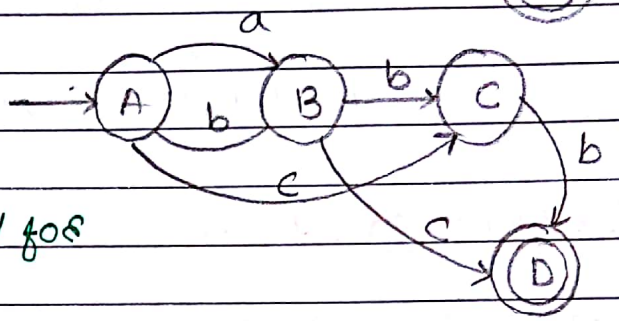
Given: $h(a) = 0,\ h(b) = 1,\ h(c) = 11$



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

Each state → 0, 1, 11.

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



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

Half(L)

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

$L = \{\ \varepsilon,\ a,\ ab,\ bb,\ aba,\ abaa\ \}$

ε (ε) ε | a — x | ab — $|a| = |b|$ | bb — (b) b | aba — x | $abaa$ — ab aa

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

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

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

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

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

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

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

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

second half(L) = $(a+b)^+$

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

- **one-third(L):**

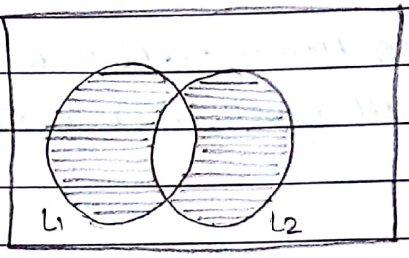
$$\frac{1}{3}(L) = \{x \mid xyz \in L,\ |x|=|y|=|z|\}$$

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

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

Exclusive OR ($L_1 \oplus L_2$)

$$L_1 \Delta L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)$$
$$= (L_1 - L_2) \cup (L_2 - L_1)$$



Symmetric Difference.

Exclusive NOR ($L_1 \odot L_2$)

$$\overline{L_1} \cdot \overline{L_2} \cup L_1 \cdot L_2$$

$$Reg_1 \odot Reg_2 = \overline{Reg_1} \cdot \overline{Reg_2} \cup Reg_1 \cdot Reg_2$$
$$= \downarrow Reg. \downarrow Reg \cup Reg.$$
$$= \text{Regular.}$$

Finite union

$R_1 \cup R_2 \cup R_3 \cup \ldots \cup R_k$, finite

$\therefore$ Regular

Finite intersection

$R_1 \cap R_2 \cap R_3 \cap \ldots \cap R_k$

$\therefore$ Regular

Finite Difference

$R_1 - R_2 - R_3 - \ldots - R_k$ $\therefore$ Regular

Finite Concatenation

$R_1 . R_2 . R_3 . \ldots . R_k$ = Regular

Finite Subset

finit 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 → finite language → Regular.

Infinite Union

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

Infinite Intersection

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

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

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

ii. $\{\varepsilon\} \cup \{ab\} \cup \{a^2b^2\} \cup \ldots = a^n b^n$

Infinite Difference

$R_1 - R_2 - R_3 \sim \ldots = $ Need not be regular.

Infinite Concatenation

Need not be regular.

i. $\phi . L_2 . L_3 \ldots = \phi$

ii. $\{a\} . \{a\} . \{a\} = \{a^\infty\}$

↳ one string infinite

Infinite subset, Infinite substitution over regular is Need not be regular.

i. $\phi - R_1 - R_2 - R_3 - \ldots\ldots = \phi$

ii. $\Sigma^* - \{\varepsilon\} - \{ab\} - \{a^2b^2\} - \ldots\ldots = \Sigma^* - a^nb^n$

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. $I_1 \cap I_2$ = may be or may not finite.
9. $I_1 - I_2 =$ ——"——
10. $F \cap Any = F$
11. $F - any = F$
12. $I \cup any = I$
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. $F \cap R =$
23. $F \cap I =$
24. $R \cap I =$
25. $\phi - any = \phi$
26. $Any - \Sigma^* =$
27. $\phi . Any =$ any.
28. $\phi \cap Any = \phi$

Context Free Grammar (CFG)

Type 0

REG(unrestricted grammar)

$\alpha \rightarrow \gamma$ (phrase structured grammar)

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 grammar

$\alpha \rightarrow \gamma$ $|\alpha| \leq |\gamma|$ and avoid using null productions

$(V+T)^*V(V+T)^* \rightarrow (V+T)^*$

Type 2

CFG)

$V \rightarrow$ any $V \rightarrow (V+T)^*$

Eg. $S \rightarrow AaBb \mid ab$

$A \rightarrow a$

$B \rightarrow bA$

Type 3

(Regular Grammar)

RLG and LLG.

$V \rightarrow T^*V \mid T^*$ $V \rightarrow VT^* \mid T^*$

1. derivation of string — LMD, RMD, Parse Tree
2. CFGs & CFLs
3. Types of CFG — Ambiguous, Unambiguous.

Desivation of Steing

Left most desivation (LMD)

Given:

$S \to AB$

$A \to aC|ef$

$B \to b|dB$

$C \to d$

adb = steing

(a b A b B a) — left most symbol; left most non-terminal; Right most non-terminal

$\xrightarrow{SF} S$

$\Downarrow \quad S \to AB$

$\xrightarrow{SF} AB$

$\Downarrow \quad A \to aC$

$\xrightarrow{SF} aCB$

$\Downarrow \quad C \to d$

$\xrightarrow{SF} adB$

$\Downarrow \quad B \to b.$

$\xrightarrow{SF} adb$ (steing)

only terminals

5 desivative steps (5 sentential forms)

desivation length of desivation = 4 (No. of substitutions)

Right Most Desivation

S

$\Downarrow \quad S \to AB$

AB

$\Downarrow \quad B \to b$

Ab

$\Downarrow \quad A \to aC$

aCb

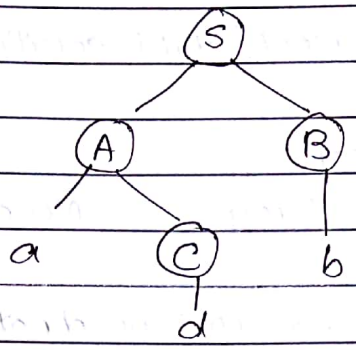
$\Downarrow \quad C \to d$

adb

length of desivation = 4

5 desivative steps

Passe Tree:



string: adb.

Derivation:

Derive string using start symbol of grammar.

LMD

In each sentential form, left most ~~deri~~ non-terminal is substituted to derive the string.

RMD

In each sentential form, right most non-terminal is substituted to derive string.

Passe 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.

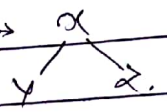
In LMD, RMD, sentential form with only terminal.

Substitution.

Non-terminal is replaced with RHS of its production.

Production Tree

Every production creates parent/child relation. $x \to yz \to$



For given CFG and given string,

Q.1. Is LMD and RMD string → Need not be same.

Q.2. Will LMD unique → 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.

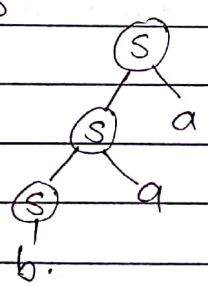
Unambiguous Context free grammar

∀ string,

No. of parse tree = 1.

$S \to SaSb$

baa



Note:

- Is given CFG ambiguous?
- Is given CFG unambiguous?

} undecidable

- Is given amb. CFG has any logic?

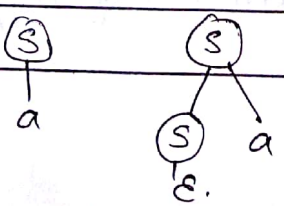
decidable.

∃ string,

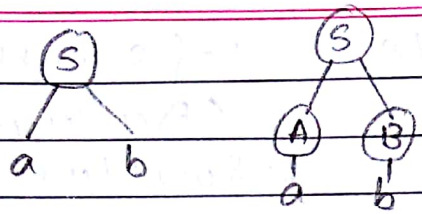
No. of parse trees > 1.

$S \to SaSaS|\varepsilon$

a:



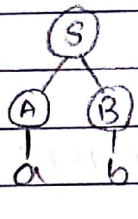
1. $S \to AB/ab$
$A \to aA/a$
$B \to bB/b$



∴ Ambiguous

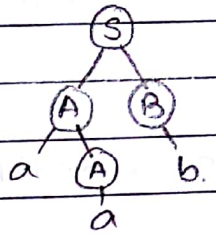
2. $S \to AB$
$A \to aA/a$
$B \to bB/b$

ab



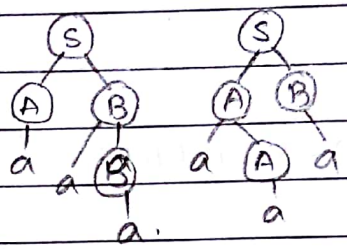
unambiguous

aab.



3. $S \to AB$
$A \to aA/a$
$B \to aB/a$

aaa.



∴ Ambiguous.

CFGs and CFLs

1. $S \to a/b$ $\quad L = \{a, b\} = a+b$

2. $S \to AB/a$ $\quad L = \{a\}$
$A \to a$, $B \to \varepsilon$

3. $S \to asb/\varepsilon$ $\quad L = a^n \varepsilon b^n = a^n b^n$

4. $S \to aas/\varepsilon$ $\quad L = \{\varepsilon, aa, aaaa, \ldots\} = (aa)^*$

5. $S \to saa/\varepsilon$ $\quad$ ——— " ———

6. $S \to asa/\varepsilon$ $\quad$ ——— " ———

7. $S \to asa/\varepsilon$ $\quad L = a^*$

8. $S \to sa/sb/\varepsilon$ $\quad (a^* + b^*) \Rightarrow \varepsilon . (a+b)^* \Rightarrow (a+b)^*$

9. $S \to sa/a/b$ $\quad L = (a+b)a^*$

10. $S \to sa/sb/a/b$ $\quad L = (a+b)^+$

11. $S \to as/bs/\varepsilon$ $\quad L = (a+b)^*$

12. $S \to as/a/b$ $\quad L = a^*(a+b)$

13. $S \to as/bs/a/b$ $\quad L = (a+b)^* \text{ or } \varepsilon (a+b)^*(a+b)$

14. $S \to as/bs$ $\quad L = \emptyset$

15. $S \to as$ $\quad L = \emptyset$

14. $S \rightarrow aSa|bSb|\varepsilon$ $L = \{\varepsilon, aa, bb, abba, \ldots\}$

(Even length palindrome.)

$L = \{wwR / w \in (a+b)^*\}$

15. $S \rightarrow aSa|bSb|a|b$ L = set of odd length strings

$= \{wxwR / w \in (a+b)^*, x \in (a+b)\}$

16. $S \rightarrow aSa|bSb|a|b|\varepsilon$ L = set of pallindrome (odd & even)

$= \{w(a+b+\varepsilon)wR / w \in (a+b)^*\}$

$= \{wxwR / w \in (a+b)^*, x \in (a+b+\varepsilon)\}$

17. $S \rightarrow aSbb|\varepsilon$ $L = \{\varepsilon, abb, aabbbb, \ldots\}$

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

18. $S \rightarrow aaSb|\varepsilon$ $L = \{\varepsilon, a^2b, a^4b^2, \ldots\}$

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

19. $S \rightarrow aSbb|a$ $L = \{a, aabb, \ldots\}$

$= a^n a b^{2n} = a^{n+1} b^{2n} \; / n \geq 0$

20. $S \rightarrow aSbb|b$ $L = \{b, ab^3, \ldots\}$

$= a^n b b^{2n} = a^n b^{2n+1} \; / n \geq 0$

21. $S \rightarrow aaSbb|\varepsilon$ $L = \{\varepsilon, aabb, \ldots\}$

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

22. $S \rightarrow AB$ $L = \{\varepsilon, ab, \ldots\}$

$A \rightarrow aA|\varepsilon \rightarrow a^*$ $= a^*.b^*$

$B \rightarrow bB|\varepsilon \rightarrow b^*$

23. $S \rightarrow AB$ $L(S) = L = L(A).L(B)$

$A \rightarrow aAb|\varepsilon \rightarrow a^n b^n$ $= a^n b^n . c^*$

$B \rightarrow cB|\varepsilon \rightarrow c^*$ $= \{a^m b^n c^k / m = n, m, k, c \geq 0\}$

24. $S \to AB$
$A \to aA/\varepsilon$ $\quad a^*$
$B \to bBc/\varepsilon$ $\quad \to b^n c^n$

$L(S) = L(A).L(B)$
$= a^* b^n c^n$
$= \{a^m b^n c^k / n=k, m,n,k \geq 0\}$

25. $S \to aSb/A/\varepsilon$
$A \to bA/\varepsilon$ $\quad \to b^*$

$L(S) = a^n b^* c^n$
$= \{a^m b^n c^k / m=k, m,n,k \geq 0\}$

26. $S \to AB$
$A \to aA/\varepsilon$ $\quad a^*.$
$B \to aBb/\varepsilon$ $\quad \to a^n b^n$

$a^* b^n L(S) = a^*.a^n b^n$
$= a^m.b^n / m \geq n$

27. $S \to AB$
$A \to aAb/\varepsilon$ $\quad a^n b^n$
$B \to bB/\varepsilon$ $\quad \to b^*$

$a^n b^* L(S) = a^n b^n . b^*$
$= a^n b^m / n \leq m$

28. $S \to SA/\varepsilon$ $\quad (ab)^*$
$A \to ab$

$L(S) = (ab)^*$

Q

29. $S \to AS/\varepsilon$ $\quad (ab)^*$
$A \to ab$

$L(S) = (ab)^*$

30. $S \to A/B$
$A \to aAbb/\varepsilon$ $\quad a^n b^{2n}$
$B \to aaBb/\varepsilon$ $\quad a^{2m} b^m$

$a^n b^{2n} + a^{2m} b^m \{a^n b^{2n}\} \cup \{a^{2m} b^m\}$

31. $S \to AB$
$A \to aAbb/\varepsilon$
$B \to aaBb/\varepsilon$

$L(S) = \{a^n b^{2n} . a^{2m} b^m\}$
$= \{a^i b^j a^k b^l / j=2i, k=2l\}$

L_1 = Regular Language (language of strings)

L_2 = set of all regular language (Language of Languages)

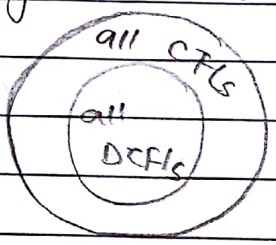
$\overline{L_1}$ = Regular language

$\overline{L_2}$ = 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 ≅ PDA ≅ CFL
- DPDA < PDA
- Languages accepted by all DPDAs ⊂ languages accepted by all PDAs.



(id1.type == id2.type)
context sensitive

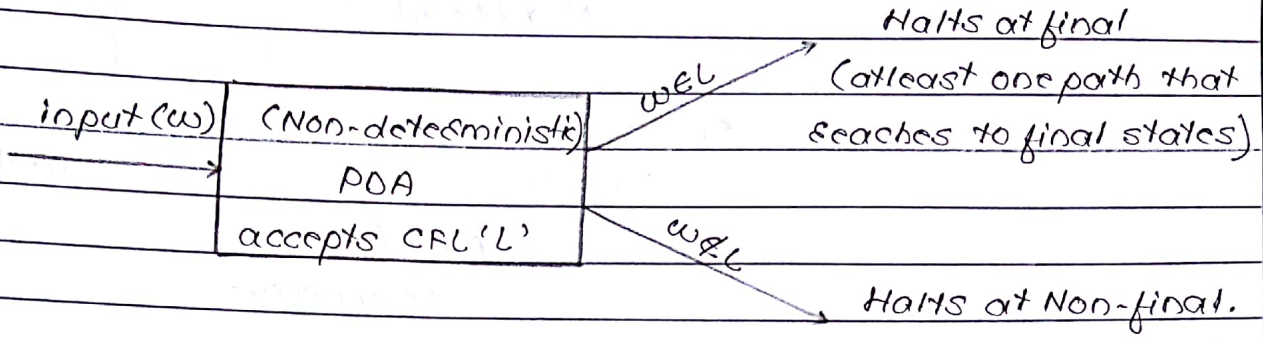
- CFL is also called as **Non-deterministic CFL.**
- Every DCFL is CFL.
- CFL need not be DCFL.
- These are CFLs but not DCFLs.

Applications of PDA

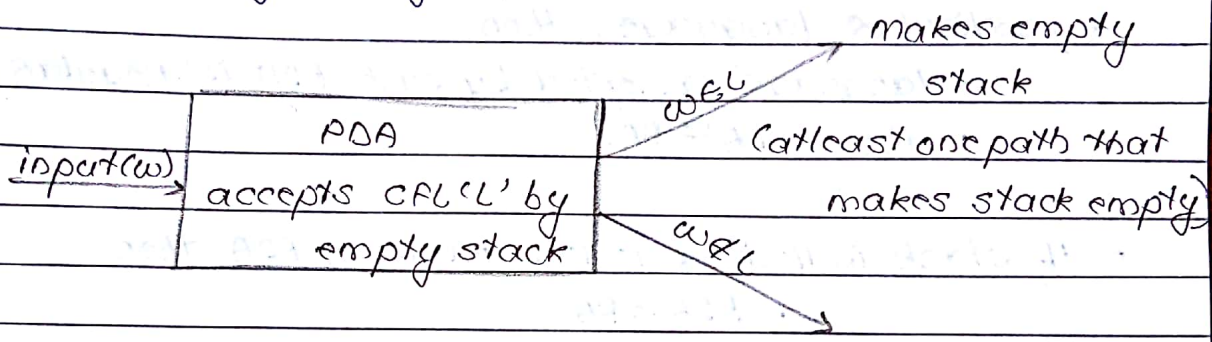
- Used in syntax ~~any~~ analysis of compiles.
- Any application uses single stack. (DFS, Recursion, Expressions)

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.

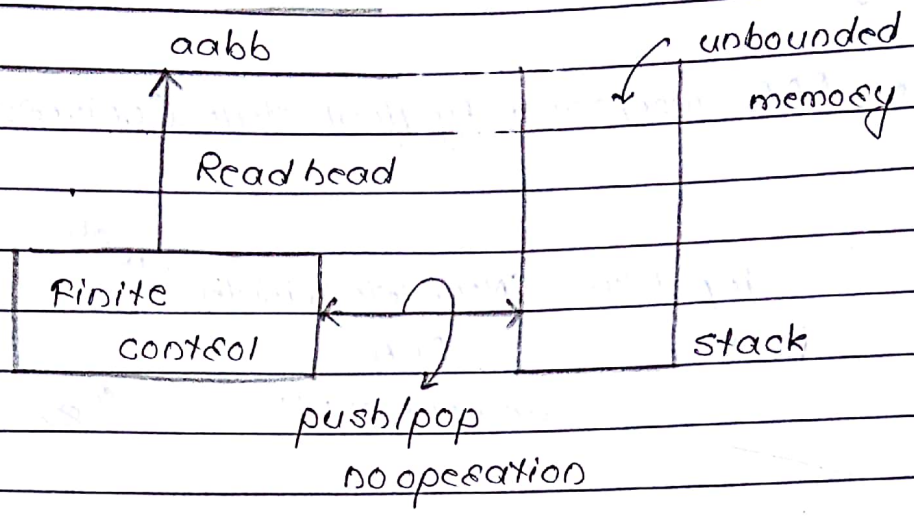
L_4 = set of all CFLs.

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:



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) = 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.)

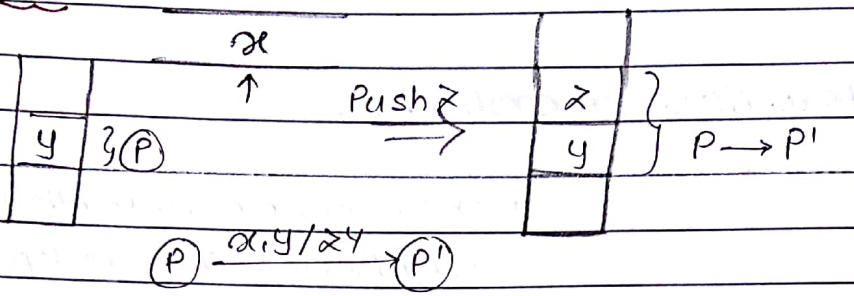
(PDA) $\delta : Q \times \Sigma_\varepsilon \times \Gamma \longrightarrow 2^{Q \times \Gamma^*}$

(DPDA) $\delta : Q \times \Sigma \times \Gamma \longrightarrow Q \times \Gamma^*$

PS i/p top of stack NS operation

Operations on the stack

Push:



Push 'z' (Read x)

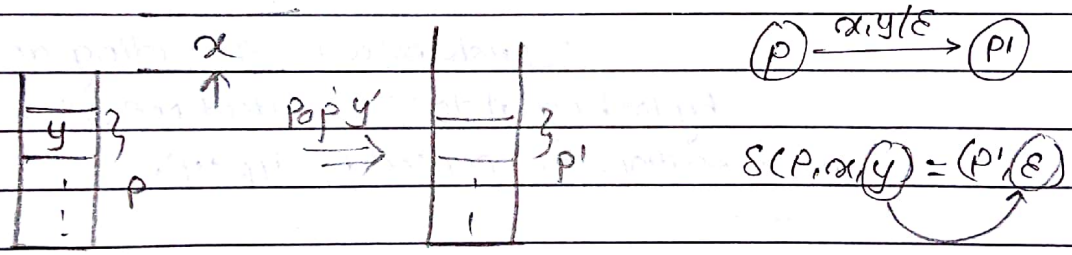
$\delta(P, x, y) = (P', zy)$ — I: when top of ~~state~~ stack is y

$\delta(P, x, \varepsilon) = (P', z)$ — II: without top of stack

$\delta(P, x, s) = (P', zs)$ — III: with entire stack

POP

pop 'y'

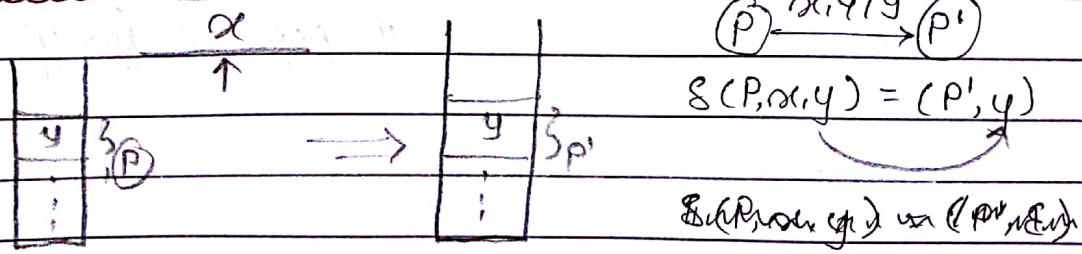


pop "y"

$\delta(P, x, y) = (P', \varepsilon)$

$\delta(P, x, ys) = (P', s)$

No operation



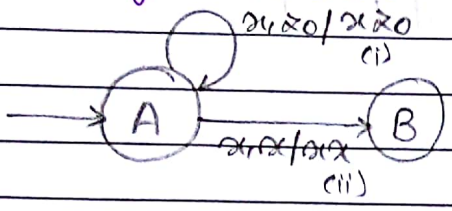
$\delta(P, x, y) = (P', \varepsilon)$

$\delta(P, x, y) = (P', y)$

$\delta(P, x, \varepsilon) = (P', \varepsilon)$

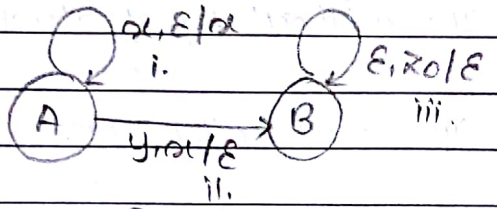
$\delta(P, x, s) = (P', s)$

Understanding transitions at each state

1. 

i. From state A, by reading input x, by looking at z0 as top of stack, it pushes x' onto stack and goes to state A.

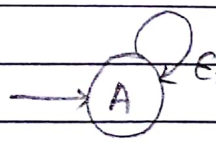
ii. From state A, by reading i/p x, by looking at x as tos it pushes x' onto stack and goes to state B.

2. 

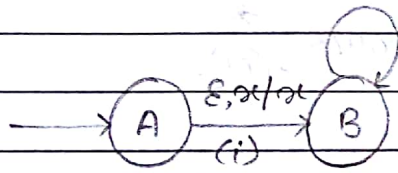
i. push 'x' for every i/p 'x'.

ii. pop 'x' for i/p 'y'.

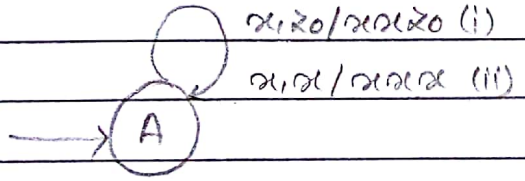
iii. pop z0 without seeing an i/p.

3. 

→ pop 'z0' without reading i/p (by reading ε)

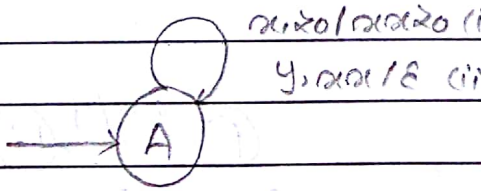
4. 

i. push 'x' without reading an i/p, by looking at tos 'x', → don't perform any operation. ii. pop 'x' for i/p 'y'.

5. 

i. push 'xx' for 'x' (first)

ii. push 'xx' for 'x' (every)

6. 

i. push 'xx' for 'x' (first)

ii. pop 'xx' for 'y'.

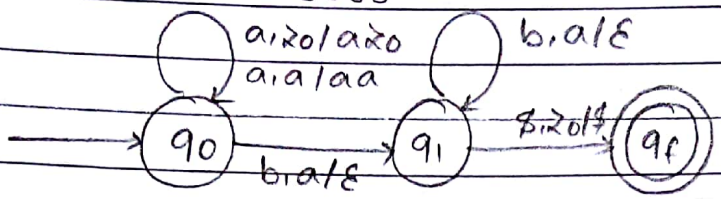
PDA acceptance
- by FS
- by ES
- Both.

- using $ & z0
- using $ but no z0
- using z0 but no $
- without using $, z0

Designing (PDA) for the following CFLs.

1. $\{a^n b^n \mid n \geq 1\}$

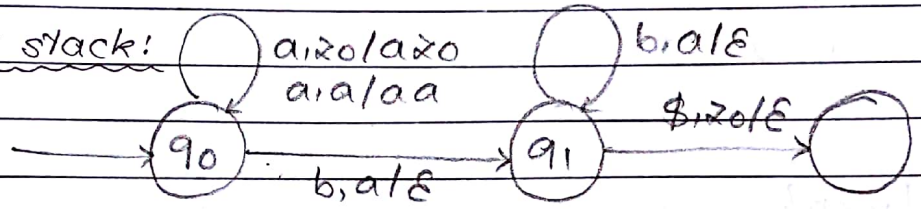
aaaabbbb



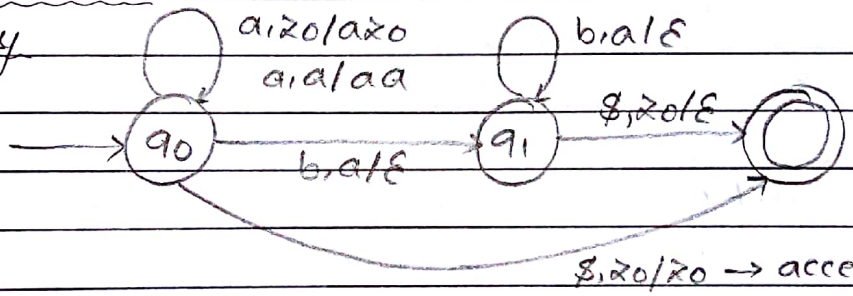
q0 : push a's for i/p a's
q1 : pop a's for i/p b's

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \varepsilon)$
$\delta(q_1, b, a) = \delta(q_1, \varepsilon)$
$\delta(q_1, \$, z_0) = \delta(q_f, \$)$

By empty stack:



By both (final state and empty stack)



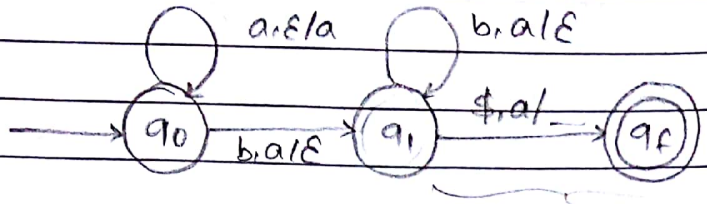
$\$, z_0/z_0$ → accepts empty string

$a^m b^n$
- $m = n$ → acceptance : $\$, z_0$
- $m > n$ → ——"—— ; $\$a$
- $m < n$ → b's are more

$m \leq n \Rightarrow m < n$ or $m = n$
$m \geq n \Rightarrow m > n$ or $m = n$
$m \neq n \Rightarrow m < n$ or $m > n$.

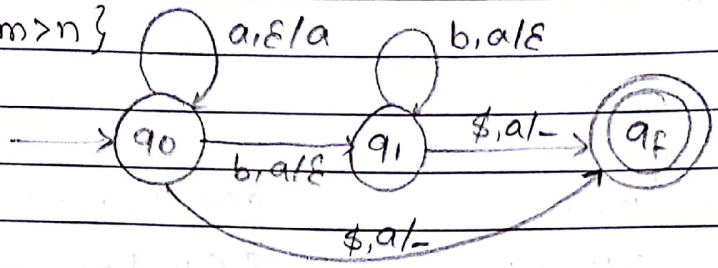
(1) $\{a^m b^n \mid m \geq n, m, n \geq 1\}$
(2) $\{a^m b^n \mid m \geq n\}$ ← 4
(3) $\{a^m b^n \mid m > n, m, n \geq 1\}$
(4) $\{a^m b^n \mid m > n\}$
(5) $\{a^m b^n \mid m < n, m, n \geq 1\}$
(6) $\{a^m b^n \mid m < n\}$
(7) $\{a^m b^n \mid m \neq n, m, n \geq 1\}$
(8) $\{a^m b^n \mid m \neq n\}$ ← a+ b+ (6)
(9) $\{a^m b^n \mid m \leq n, m, n \geq 1\}$ ← 5
(10) $\{a^m b^n \mid m \leq n\}$ ← 6, 2

3. $\{a^m b^n \mid m > n, m,n \geq 1\}$

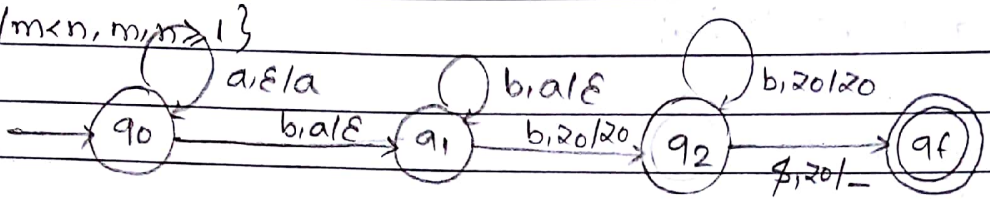


a's are more

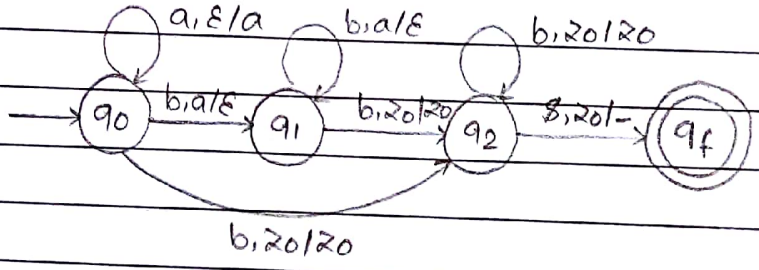
4. $\{a^m b^n \mid m > n\}$



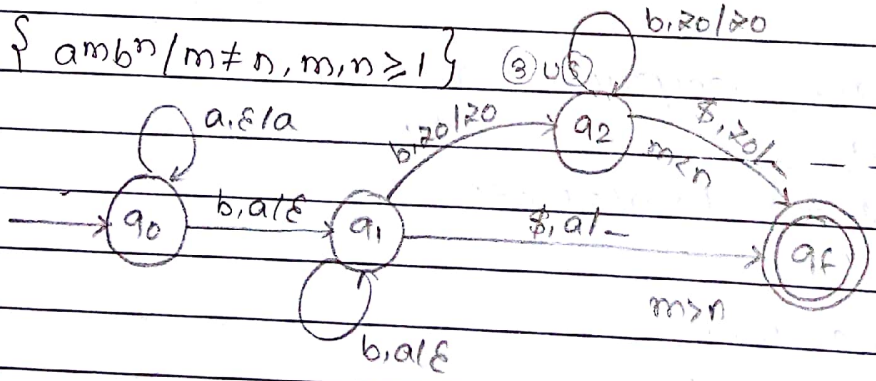
5. $\{a^m b^n / m < n, m,n \geq 1\}$



6. $\{a^m b^n \mid m < n\}$ $m=0 \rightarrow b^+$



7. $\{a^m b^n / m \neq n, m,n \geq 1\}$ ③ ∪ ⑤



15. $\{w \mid w \in (a+b)^+, n_a(w) = n_b(w)\}$

16. $\{w \mid w \in (a+b)^*, n_a(w) < n_b(w)\}$

17. $\{w \mid w \in (a+b)^+, n_a(w) < n_b(w)\}$

18. $\in (a+b)^*, n_a(w) > n_b(w)\}$

19. $\in (a+b)^+, n_a(w) > n_b(w)\}$

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

21. $\in (a+b)^+, n_a(w) \geq n_b(w)\}$

22. $\in (a+b)^*, n_a(w) \leq n_b(w)$

$\in (a+b)^+, n_a(w) < n_b(w)$

$\in (a+b)^*, n_a(w) \neq n_b(w)\}$

$\in (a+b)^+, n_a(w) \neq n_b(w)\}$

$a^n b^n$.

$\{w / w \in a^* b^*, n_a(w) = n_b(w)\}$

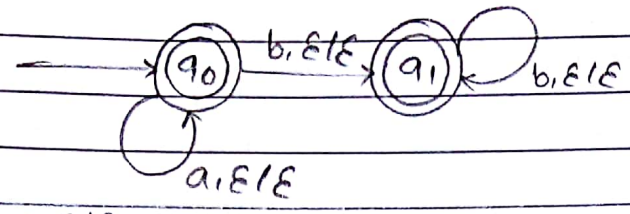
$a^m b^n / m = n$

$a^m b^n / n_0.o_f\ a's = n_0.o_f\ b's$

$a^n b^n / n \geq 0$

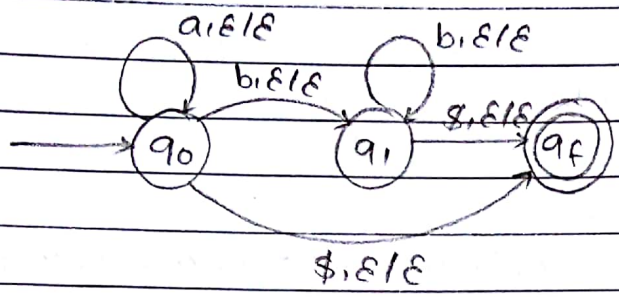
13.

without $



Note:

Any finite automata can be converted to PDA.



14. $\{w / w \in (a+b)^*,\ n_a(w) = n_b(w)\}$

case 01: a, z0/z0 (push a)

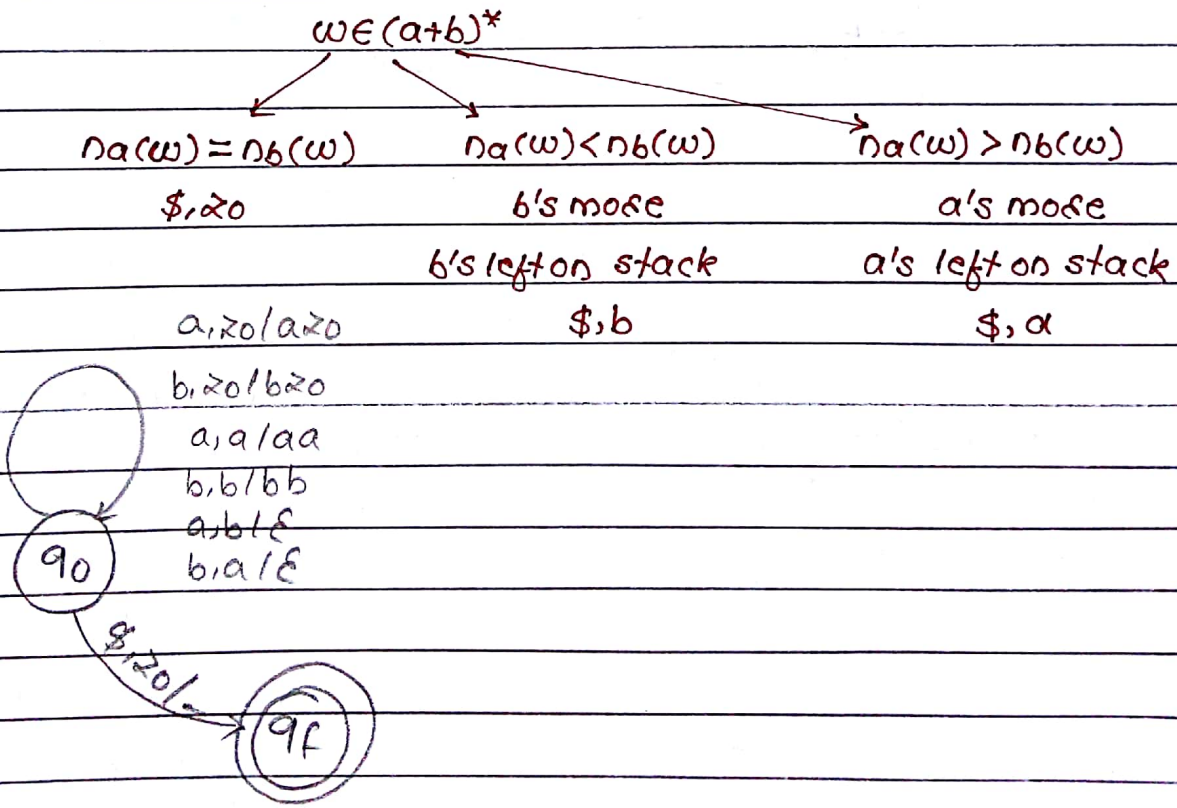
case 02: b, z0/z0 (push b)

case 03: a, a/aa (push a)

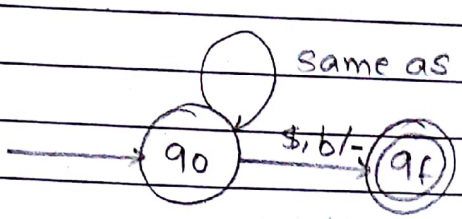
case 04: b, b/bb (push b)

case 05: a, b/ε (pop b)

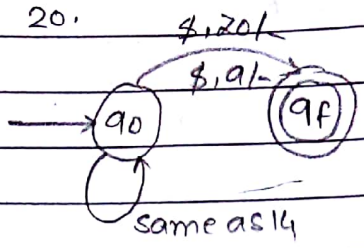
case 06: b, a/ε (pop a)



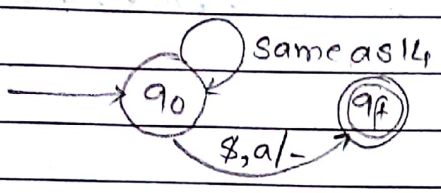
16.



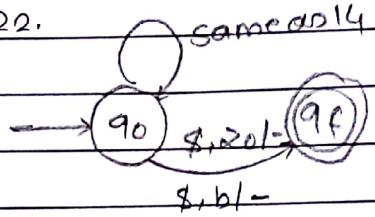
20.



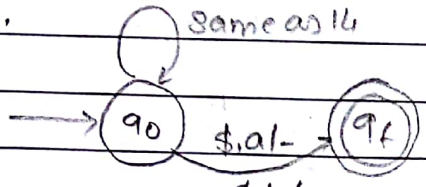
18.



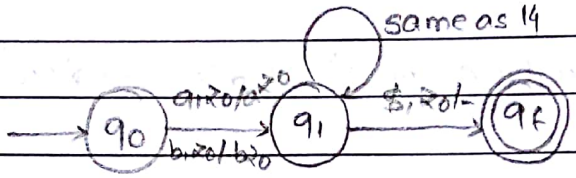
22.



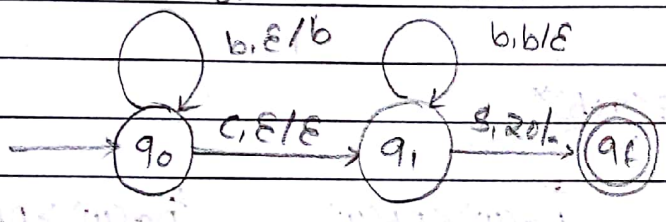
24.



15. $\{w \mid w \in (a+b)^+, n_a(w) = n_b(w)\}$



26. $\{wcw^R / w \in (a+b)^*\}$



a,z0/az0
b,z0/bz0
a,a/aa
b,b/bb
a,b/ab
b,a/ba
} a,ε/a
b,ε/b.

ε,z0/z0
c,a/a
c,b/b
} c,ε/ε

27. $\{wcw^R / w \in (a+b)^+\}$

28. $\{a^n b^{2n} / n \geq 1\}$

29. $\{a^n b^{2n}\}$

30. $\{a^{2n} b^n / n \geq 1\}$

31. $\{a^{2n} b^n\}$

~~32.~~ $\{ww^R / w \in (a+b)^*\}$

Every student is clever

$\forall x (S(x) \rightarrow C(x))$

Some student is present

$\exists x (S(x) \wedge P(x))$

$S \rightarrow C \quad \exists S \vee C$

logically same

Student is clever.

every student is clever.

If student exist, must be clever.

no student exist or student clever.

$a^n b^{2n}$ (#b's = 2.#a's)

I. $a^n b^{2n}$

Each 'a' push 2x's

Each 'b' pop 1x.

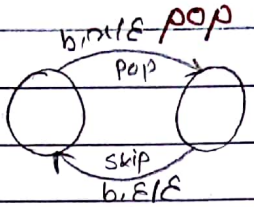
a, ε / xx

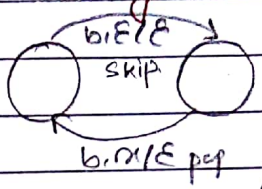
a, ε / aa

II. $a^n b^{2n}$

push 1x for each 'a'

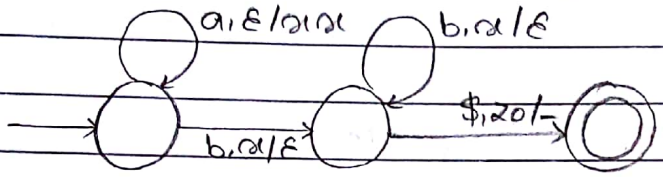
pop 1x for every 2b's



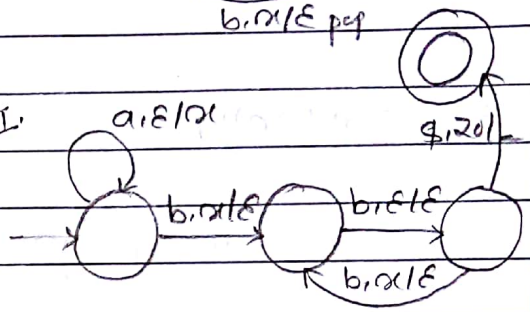


28. $a^n b^{2n} / n \geq 1$

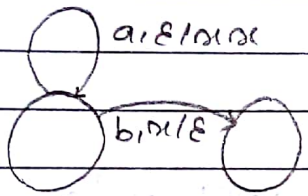
I.



II.



29.



$a^{2n} b^n$

I. $a^{2n} b^n$

each 2a's push 1a

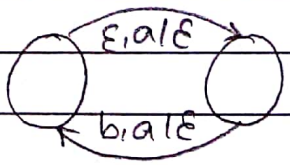
each 'b' pop '1a'

II. $a^{2n} b^n$

push 'a' for each 'a'

pop 2a's for each 'b'

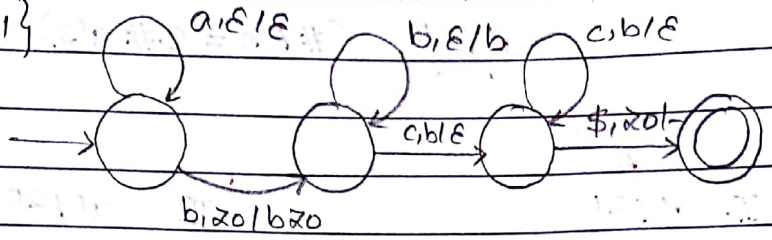
b, aa / ε



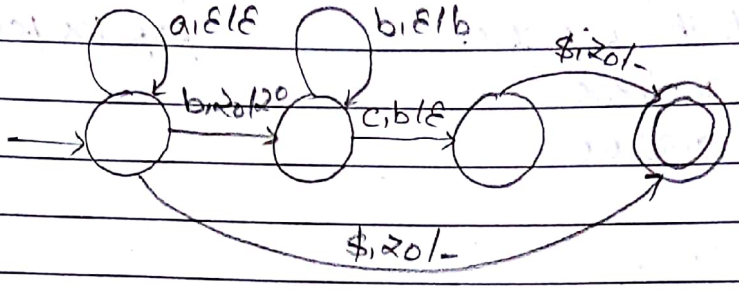
33. $\{a^*b^nc^n\}$ 34. $\{a^nb^*c^n\}$ 35. $\{a^nb^nc^*\}$

32. $\{a^*b^nc^n \mid n \geq 1\}$

$a^*b^nc^n$ b^nc^n



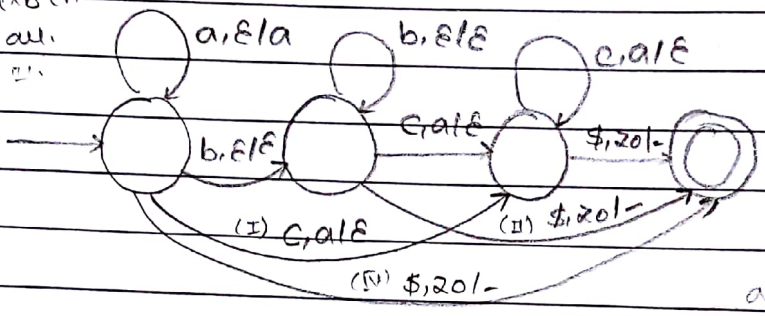
33. $\{a^*b^nc^n\}$



34. $\{a^nb^*c^n\}$

a^nc^n b^* $a^*b^nc^n$ ε (no a, b, c)

no b's | no a's & c's | all



36. $\{a^mb^nc^k \mid \text{if } (m = \text{even}) \text{ then } (n = k)\} \Rightarrow \{a^mb^nc^k \mid (m = \text{odd}) \oplus (n = k)\}$

$(aa)^*b^nc^n$ $a(aa)^*b^nc^n$

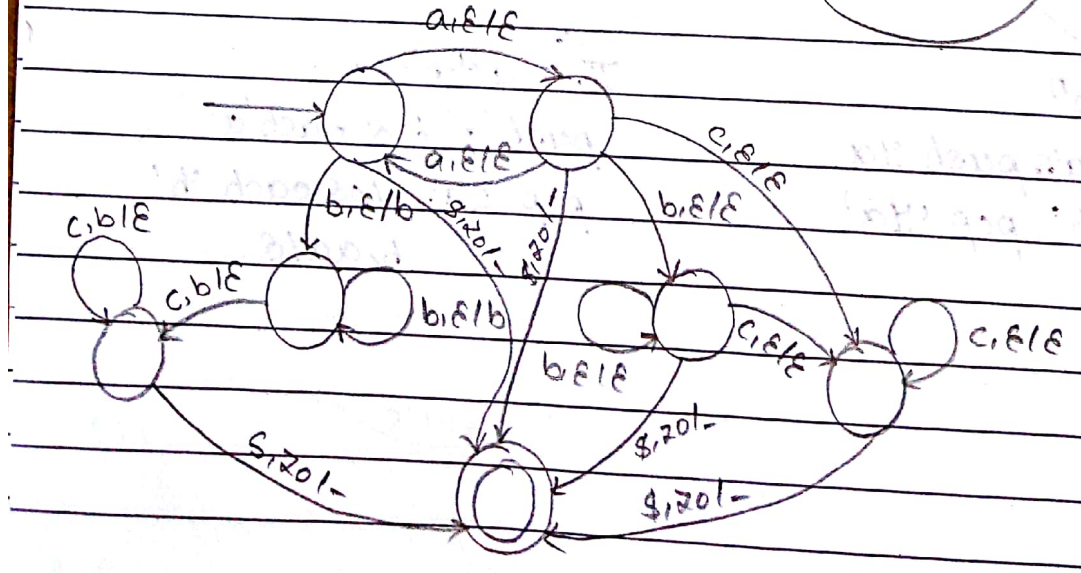
$m \neq$ even

$m =$ even

$(aa)^*b^kc^k$ $a(aa)^*b^*c^*$

$m =$ odd

$a(aa)^*b^*c^*$ $a^*b^mc^m$



37. $\{a^m b^n c^k$ / if $(n = \text{even})$ then $(m = k)\}$

38. $\{a^m b^n c^k$ / if $(k = \text{even})$ then $(m = n)\}$

39. $\{a^m b^n c^k$ / if $(m = n)$ then $(k = \text{even})\}$

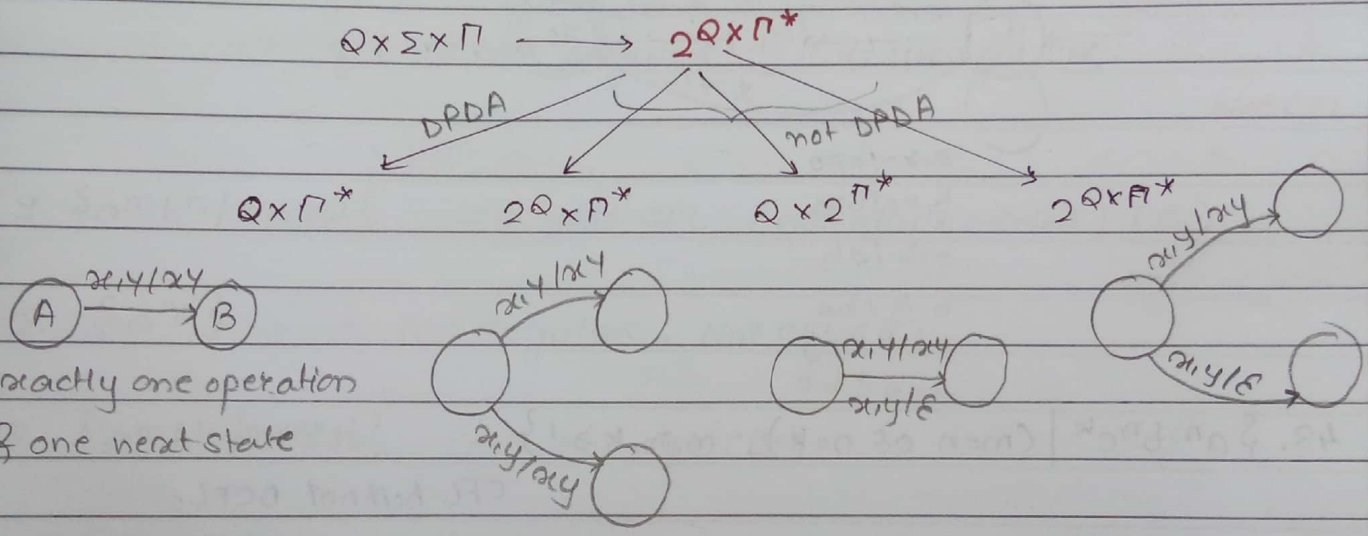
40. $\{a^m b^n c^k$ / if $(n = k)$ then $(m = \text{even})\}$

41. $\{a^m b^n c^k$ / if $(m = k)$ 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. $\{ww^R / w \in (a+b)^*\}$ 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 = tos) $\Rightarrow$ 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. $\{a^m b^n c^k \mid \text{if } (m = n) \text{ then } (k = \text{even})\}$

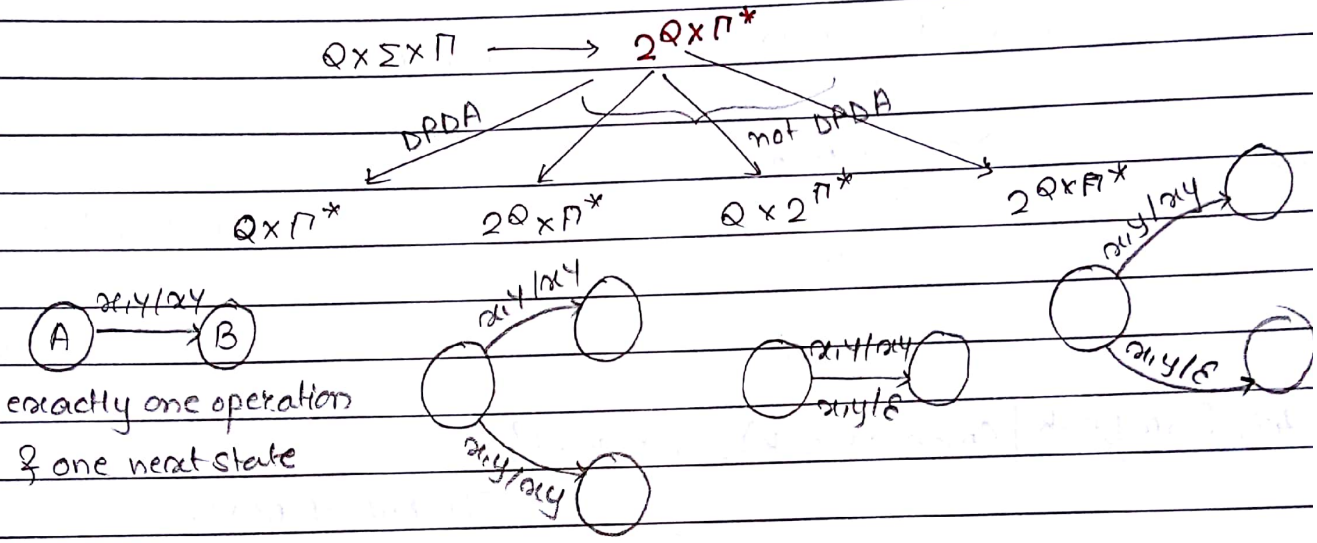
40. $\{a^m b^n c^k \mid \text{if } (n = k) \text{ then } (m = \text{even})\}$

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. $\{ww^R \mid w \in (a+b)^*\}$ CFL but not DCFL.

$w = aa$

$w = ab$

$w = abba$

Q.1. when w^R begins ?

present symbol same as previous i/p.

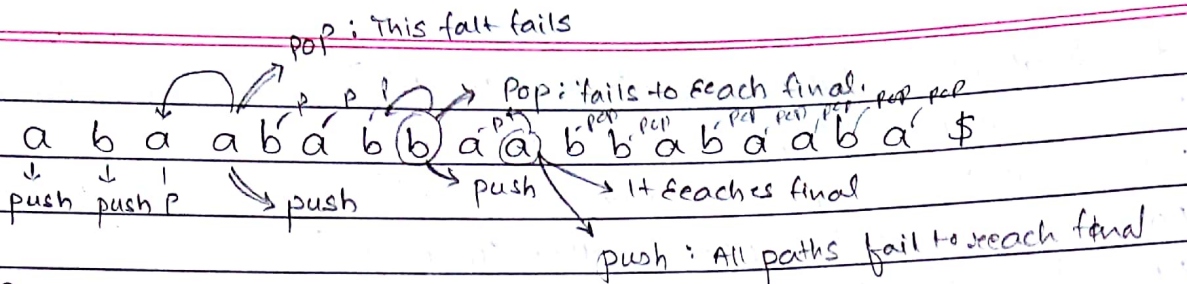
If present i/p same as previous i/p

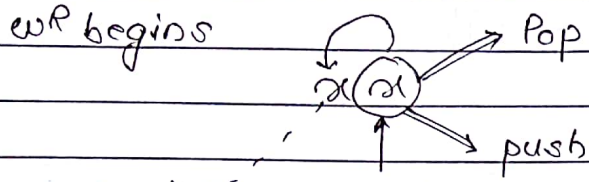
$\Downarrow$

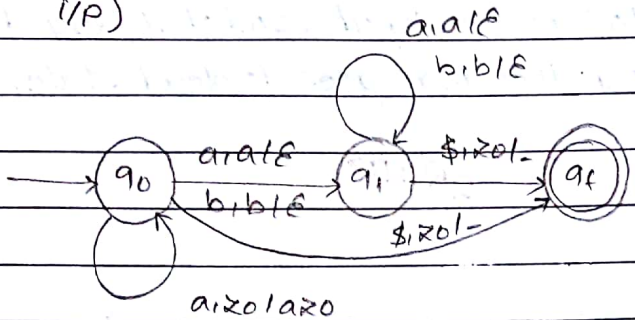
w^R may begin or w may continue.

Q.2. If begins, how do you understand ?

If (present i/p = tos) $\Rightarrow$ we may reverse w^R.





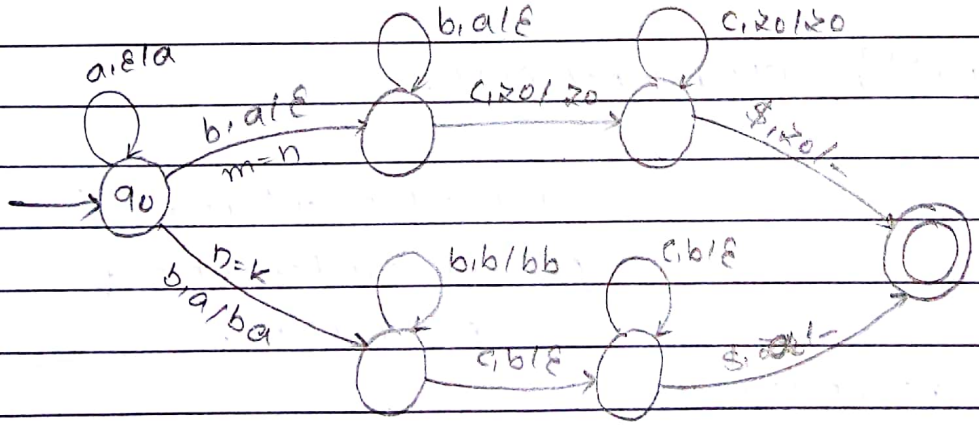


a,z/a z0
b,z/b z0
a,b/ab
b,a/ba
a,a/aa
b,b/bb

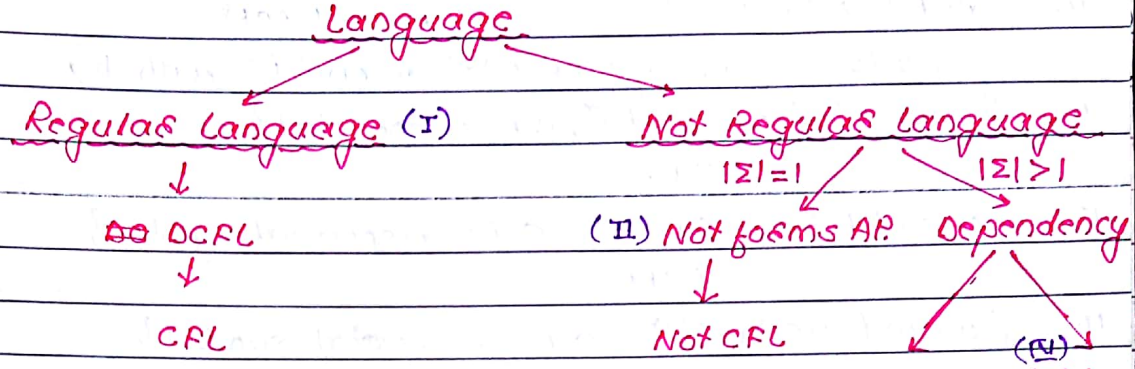
43. $\{a^m b^n c^k \mid (m=n \text{ or } n=k),\ m,n,k \geq 1\}$ $a^n b^n c^n$

CFL but not DCFL.

PDA exist but not DPDA.



Identifying CFLs and DCFLs.



Language over one symbol ⇒ PDA ≅ FA

- If L is regular ⇒ L is CFL.
- If L is not regular ⇒ L is also not CFL.

1. $\{a^m b^n / m = n^2\}$ ⇒ Not CFL (not stack order) (IV)
2. $\{a^{n^2}\}$ ⟶ Not regular, Not CFL (II)
3. $\{a^m b^n / (m \neq n)\}$ ⟶ Not CFL (IV)
4. $\{a^m b^n / gcd(m,n) = 1\}$ ⟶ Not CFL.
5. $\{a^m b^n c^k / m = n = k\}$ ⟶ $a^n b^n c^n$ ∴ Not CFL
6. $\{a^m b^{2m} c^{3m}\}$ ⟶ Not CFL. (IV)
7. $\{aw / n_a(w) = n_b(w), w \in (a+b)^*\}$ ⟶ DCFL
8. $\{wb / n_a(w) = n_b(w), w \in (a+b)^*\}$ ⟶ DCFL
9. $\{a^m b^n / m = n \text{ or } m < n\}$ ⟶ DCFL

10. $\{a^m b^n \mid m=n \text{ or } m=2n\}$ ⟶ CFL but not DCFL.

min: No. of a's = no. of b's max: no. of a's = 2 × no. of b's.

11. $\{a^m b^n \mid n \le m \le 2n\}$ CFL but not DCFL.

~~Twice~~ $\{w \mid n_a(w) = n_b(w)+1,\ w \text{ endits with } b\}$

12. $\{awcw^R \mid w \in (a+b)^*,\ c \text{ is a speacial}\}$

DCFL.

13. $\{wcw^R b \mid w \in (a+b)^*,\ c \text{ is a speacial symbol}\}$

DCFL

14. $\{wcw \mid w \in (a+b)^*,\ c \text{ is a special symbol}\}$

~~DCFL~~. not stack or dec ∴ not CFL.

15. $\{wcw \mid w \in (a+b)^*,\ c \in (a+b)\}$

not ~~dcf~~ CFL.

16. $\{wcw^R \mid w \in (a+b)^*, c \in (a+b)\}$ CFL but not DCFL.

17. $\{a^m b^n \mid n = 100m\} = a^m b^{100m} \Rightarrow$ DCFL

18. $\{a^m b^n \mid m = n^n\} \Rightarrow$ not CFL

19. $\{a^m b^n \mid m = 2^n\} \Rightarrow$ not CFL.

20. $\{a^{m+n} b^n c^m\}$ DCFL

21. $\{a^m b^{m+n} c^n\}$ DCFL

22. $\{a^m b^n c^{m+n}\}$ ~~not cft~~ DCFL.

23. $\{a^{m+n} b^{m+n} c^{m+n}\}$ not CFL

24. $\{a^{m+n} b^m c^{m+n}\} c^m$ not LCFL.

25. $\{a^{m+n} b^{n+k}\}$ a^*b^* (Regular)

26. $\{a^m b^n c^k \mid \text{if } (m=n) \text{ then } k=\text{even}\}$ ⟷ DCFL

27. $\{a^m b^n c^k \mid \text{if } (m=n) \text{ then } (n=k)\} = \{a^m b^n c^k \mid (m!=n) \text{ or } (n=k)\}$

$m=n=k$: $a^p b^p c^p$

$m \neq k$: $a^q b^r c^* \mid q \neq r$; $a^q b^p c^p \mid q \neq p$

$m \neq n$: $a^q b^r c^* \mid q \neq r$

$a^* b^t c^t$

∴ CFL but not DCFL.

$\{a^i b^j c^k \mid i=j \text{ or } i \neq j\}$

28. $\{xww^R / w, x \in (a+b)^+\}$ — CFL but not DCFL

29. $\{ww^Rx / w, x \in (a+b)^+\}$ — ——"——

30. $\{wxw^R / w, x \in (a+b)^+\}$ — Regular

31. $\{ww / w \in (a+b)^+\}$ — Not CFL

32. $\{\bar{w}w / w \in (a+b)^+\}$ — CFL but not DCFL.

33. $\{wxw / w, x \in (a+b)^+\}$ — Not regular, not CFL.

34. $\{wwx / w, x \in (a+b)^+\}$ — Not CFL

35. $\{xwx / w, x \in (a+b)^+\}$ — Not CFL.

36. $\{ca^nb^{2n}\} \cup \{da^{2n}b^n\}$ — DCFL

37. $\{a^nb^{2n}c\} \cup \{a^{2n}b^nd\}$ — CFL but not DCFL.

38. $\{a^mb^nc^kd^l / \text{if } (m+n=k) \text{ then } (d=\text{even})\}$ — DCFL

39. $\{a^{2^{n2}}\}$ — not regular → not CFL.

40. $\{a^mb^nc^kd^l / m+n=k+l\}$ — DCFL

41.

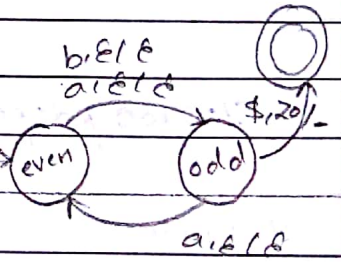
Note: Any CFL can be constructed with only one state (CFG ⇒ PDA algorithm)

$L = \{ww / w \in (a+b)^*\}$ — No odd length string; Even length in form of ww

$\bar{L} = \{\bar{w}w / w \in (a+b)^*\}$ — even length not in ww; Every odd length

$= \Sigma^* - L$

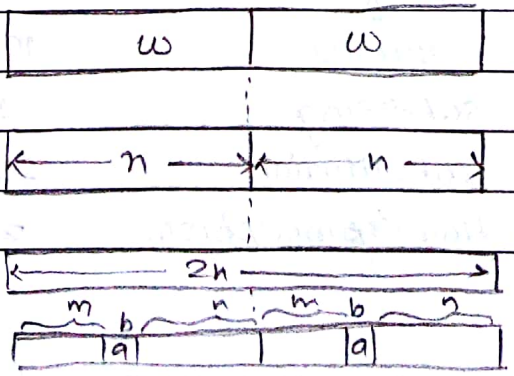
$= \Sigma^* - \{ww / w \in (a+b)^*\}$



$\bar{w}w =$ — All odd length strings.

Even length not in form of ww

ww logic



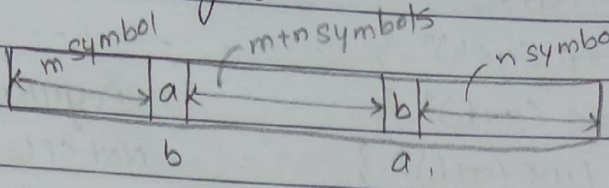
$|n+m|$ Between same positions of 2w's length $= m+n = n+m$

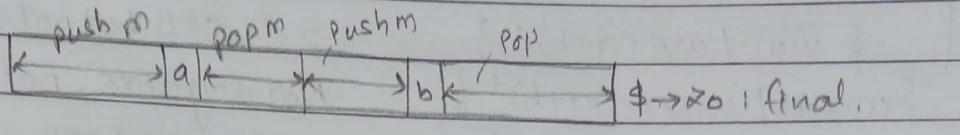
To show not in ww, we need different symbol somewhere in the even length string.

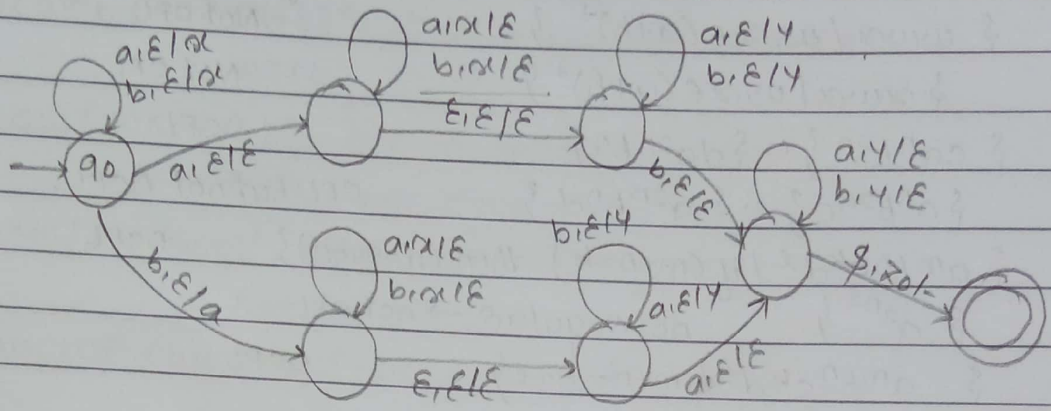
$\exists x | uv=x, |u|=|v|$ same position in u & v are diff.

$\exists x | uv=x, |u|=|v|$ same position

not in ww (even length)



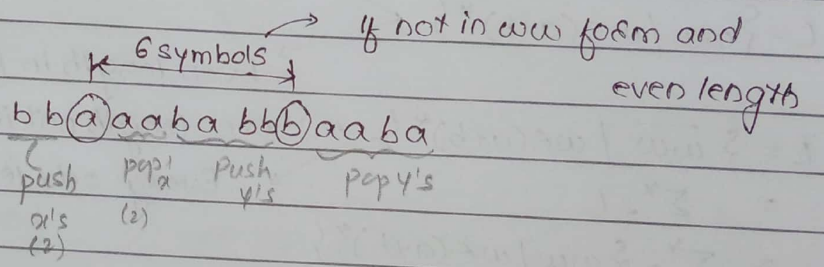




Note:

Complement of CFL → need not be CFL.

complement of DCFL → DCFL.



Closure properties of CFLs and DCFLs

1. Union
2. Intersection
3. Complement
4. Difference
5. concatenation
6. Reversal
7. Kleene closure
8. positive closure
9. subset
10. prefix
11. suffix
12. substring
13. substitution
14. Homomorphism
15. $h^{-1}(L)$
16. Finite union
17. Finite intersection
18. Finite difference
19. finite concatenation
20. finite subset
21. finite substitution

22. Infinite union
23. Infinite Intersection
24. Infinite Difference
25. Infinite concatenation
26. Infitite subset
27. Infinite substitution
28. Regular closure

DCFLs (closed) → Complement, prefix
CFLs (not closed) → Intersection, complement, difference
(finite n) (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.

$CFL_1 = a^n b^n \rightarrow S_1 \rightarrow aS_1b \mid \varepsilon$

$CFL_2 = a^{2n} b^n \rightarrow S_2 \rightarrow aaS_2b \mid \varepsilon$

$CFL_1 \cup CFL_2$: $S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow aS_1b \mid \varepsilon$
$S_2 \rightarrow aaS_2b \mid \varepsilon$

Eg. (for not DCFL)

$DCFL_1 = a^n b^n$

$DCFL_2 = a^{2n} b^n$

$DCFL_1 \cup DCFL_2 = \{a^m b^n \mid m=n \text{ or } m=2n\}$
= not DCFL
= CFL

CFL but not DCFL
ww^R, $\overline{ww}$
not CFL → ww, $a^nb^nc^n$

$\phi; \Sigma^*$ reg
a^nb^n → 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)

$CFL_1 \rightarrow a^nb^nc^*$ $CFL_2 \rightarrow a^*b^nc^n$ $\therefore CFL_1 \cap CFL_2 = a^nb^nc^n$

$DCFL_1$ $DCFL_2$ 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

$\overline{L} = \{ww \mid w \in (a+b)^*\}$ not CFL

Eg. $a^nb^n \rightarrow$ DCFL

$\overline{a^nb^n} \rightarrow$ DCFL $(\Sigma^* - a^nb^n)$

$L = \overline{a^nb^nc^n}_{(CFL)}$ $\overline{L} = a^nb^nc^n$ not CFL

$\{a^mb^nc^k \mid m \neq n$ or $n \neq k$ 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.

∪, L1.L2, L*, L+

PDA, CFL, CFGL

Concatenation

CFLs (closed)

$L_1 = a^n b^n \rightarrow S_1 \rightarrow aS_1b \mid \varepsilon$ $\quad L_2 = a^{2n}b^n \Rightarrow S_2 \rightarrow aaS_2b \mid \varepsilon$

$S \Rightarrow S_1 S_2$

$S_1 \rightarrow aS_1b \mid \varepsilon, \quad S_2 \rightarrow aaS_2b \mid \varepsilon$

DCFLs (not closed)

$L_1 = a^n b^n$ (DCFL) $\quad L_2 = a^{2n}b^n$ (DCFL)

$L_1.L_2 = a^n b^m . a^{2m}b^m \rightarrow$ CFL but not DCFL.

DCFL1. DCFL2 = need not be DCFL

= always CFL.

Reversal

$(DCFL)^{Rev}$ = Need not be DCFL. $\quad (CFL)^{Rev}$ = CFL.

$L = (\{a^n b^n\} \cup \{da^{2n}b^n\})$ = DCFL. $\quad$ Eg. $L = a^n b^n$ (CFL)

$L^{Rev} = (\{b^n a^n c\} \cup \{b^n 2^n d\})$ $\quad S \rightarrow aSb \mid \varepsilon$

not DCFL but CFL. $\quad$ // reverse every RHS

$S \rightarrow bSa \mid \varepsilon$

$L^{Rev} = b^n a^n$

$(DCFL)^{Rev}$ = always CFL.

Kleene closure

$(DCFL)^*$ = Need not be DCFL. $\quad (CFL)^*$ = CFL

$DCFL = \{ca^n b^n\} \cup \{a^{2n}b^n\} = L$ $\quad L = a^n b^n \quad S \rightarrow aSb \mid \varepsilon$

//

$L^* = (\{ca^n b^n\} \cup \{a^{2m}b^m\})^*$ $\quad L^* : \alpha \rightarrow S\alpha \mid \varepsilon$

not DCFL (always CFL) $\quad S \rightarrow aSb \mid \varepsilon$

also L^+ = " "

Positive closure

$(DCFL)^+$ = Need not be DCFL $\quad (CFL)^+ = CFL$

always CFL. $\quad L = a^n b^n$

$S \rightarrow aSb \mid \varepsilon$

$L^+ : \alpha \rightarrow S\alpha \mid S$

$S \rightarrow aSb \mid \varepsilon$

Subsets

Subset (DCFL) = Need not be DCFL.

Subset (CFL) = Need not be CFL.

$a^nb^nc^*$ (DCFL)(CFL)

$\{abc\}$ c^* a^nb^n $a^nb^nc^n$ → not DCFL, not CFL

Substitution

DCFL substitution (not closed)

$L = \{ca^nb^n\} \cup \{da^{2n}b^n\}$

$f(c) = \{\varepsilon\}$ $f(d) = \{\varepsilon\}$

$f(a) = \{a\}$ $f(b) = \{b\}$ } DCFL

$f(L) = \{a^nb^n\} \cup \{a^{2n}b^n\}$

not DCFL

CFL substitution (closed)

$L = a^nb^n$ Given CFL

$f(a) = 0^m1^{2m}$ } CFLs

$f(b) = 0^{2k}1^k$ }

$f(L) = (f(a))^n (f(b))^n$

$= (0^m1^{2m})^n (0^{2k}1^k)^n$

$S \to xSY \mid \varepsilon$

$x \to 0x11 \mid \varepsilon$, $Y \to 00Y1 \mid \varepsilon$

Homomosphism

closed for CFLs, not closed for DCFLs.

Invese Homomosphism

closed for CFLs and DCFLs.

Prefix :

closed for both CFLs & DCFLs

$L = a^nb^n \Rightarrow S \to aSb \mid \varepsilon$

$prefix(L) = \{ a^mb^m \mid m \geq n \}$

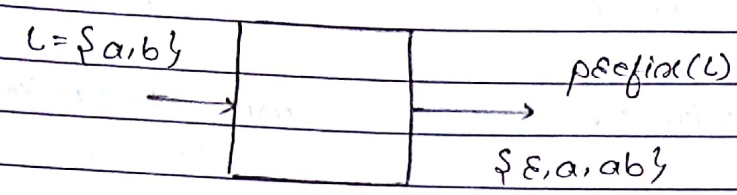
$S \to \varepsilon \mid aAS \mid aSb$

aabb ——— aabb

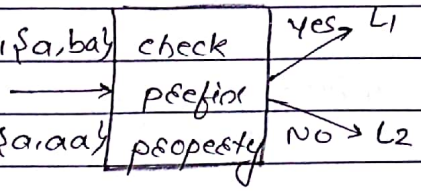
ε a aa aab

Prefix property is different from prefix operation.

prefix operation



prefix property



Suffix

DCFLs ⇒ not closed

DCFL = {$a^n b^n$} ∪ {$da^{2n}b^n$}

suffix(L) = not DCFL

= CFL.

CFLs ⇒ closed

$L = a^n b^n$

$S \rightarrow aSb \mid \varepsilon$

⇓

suffix(L) $S \rightarrow \varepsilon \mid b \mid Sb \mid aSb$

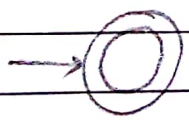
$\{a^m b^n \mid n \geq m\}$

Substring

CFLs ⇒ closed DCFLs ⇒ not closed

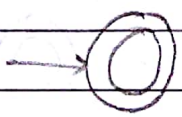
DCFLs	Union	Intersection	difference	Concatenation	⊆	substitution
Finite:	×	×	×	×	✓	×
Infinite:	×	×	×	×	×	×
CFLs						
Finite:	✓	×	×	✓	✓	✓
Infinite:	×	×	×	×	×	×

Understanding PDA:

1. 

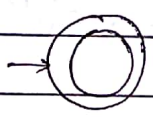
with z_0 & $

L = ϕ

2. 

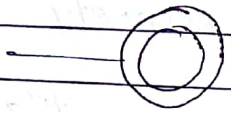
z_0 is present and no $

$L = \{\varepsilon\}$

3. 

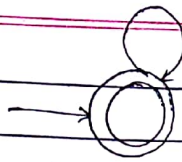
z_0 is not present but $ exist.

L = ϕ

4. 

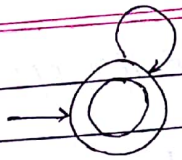
without Z_0 & $\$$

$L = \{\varepsilon\}$

5. 

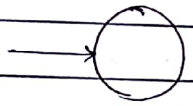
with $\$$ & Z_0

$L = \{\varepsilon\}$

6. 

with $\$$ but no Z_0

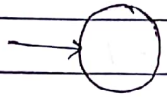
$L = \{\varepsilon\}$

7. 

with Z_0 & $\$$

PDA by empty stack

$L = \phi$

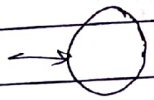
8. 

Z_0 is present

& no $\$$.

∴ PDA by Empty stack.

$L = \phi$

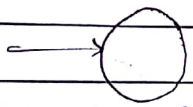
9. 

Z_0 is not present

but $\$$ exist

PDA by ES.

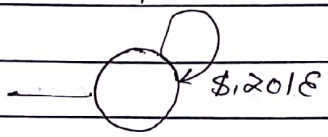
$L = \emptyset$

10. 

without Z_0 & $\$$

PDA by ES.

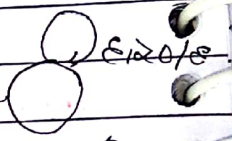
$L = \{\varepsilon\}$

11. 

with $\$$ & Z_0

PDA by ES.

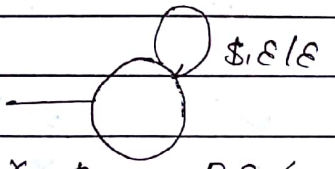
$L = \{\varepsilon\}$

12. 

with Z_0 & no $\$$

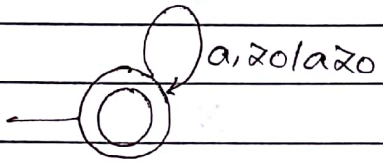
PDA by ES.

$L = \{\varepsilon\}$.

13. 

Z_0 ✗, $\$$ ✓ E.S ✓

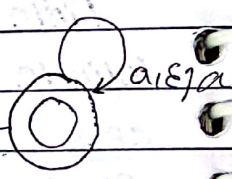
$L = \{\varepsilon\}$

14. 

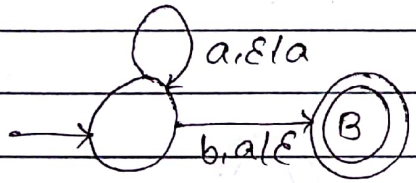
final state mechanism

no $\$$ but Z_0.

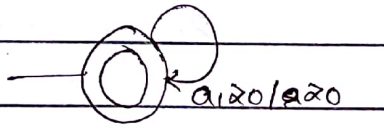
$\{\varepsilon, a\}$

15. 

$L = a^*$.

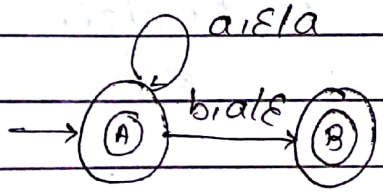
16. 

$L = a^+ b$

17. 

$L = \{\varepsilon, a, aa, \ldots\}$

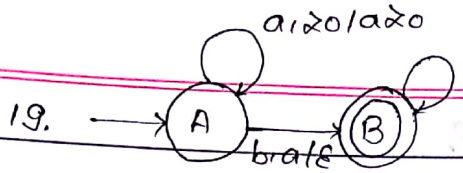
$= a^*$

18. 

$A \rightarrow a^*$, $B \rightarrow a^+ b$

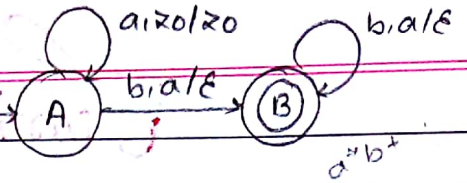
$\therefore a^* + a^+ b$

19.



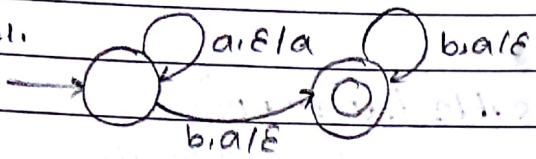
$a^* + a^* b \cdot b^*$ $\quad a^* b + b^*$

20.



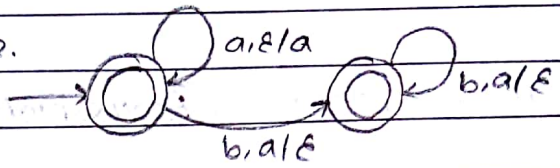
$a^* b^+$

21.

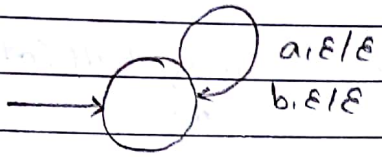


a^*

22.

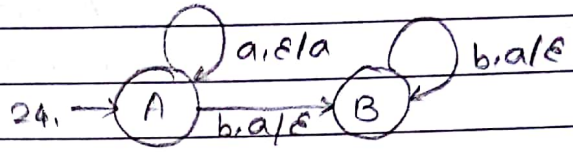


23.



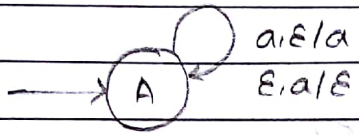
ES.

24.



ES

25.



Eg.

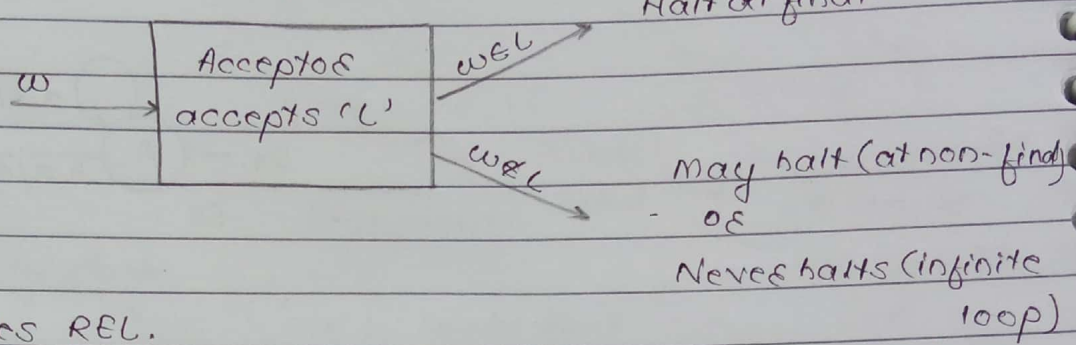
$a^n b^m c^m d^n \rightarrow$ CFL

$a^n b^n c^m d^m \rightarrow$ CFL

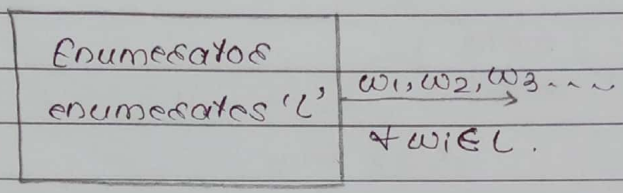
$a^n b^m c^n d^m \rightarrow$ not CFL

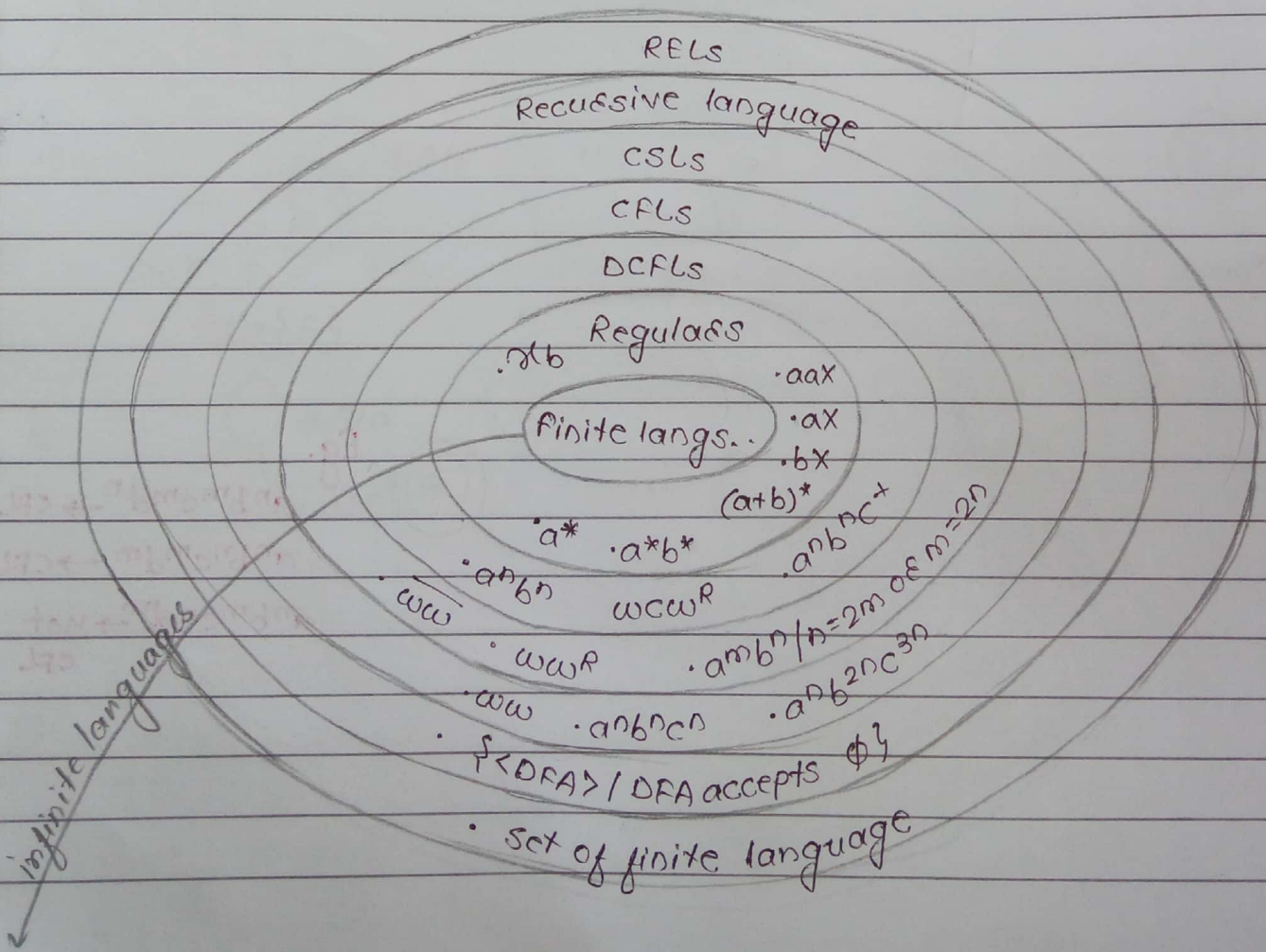
Turing Machine

- It accepts REL. (Recursively Enumerable Language)



- It enumerates REL.

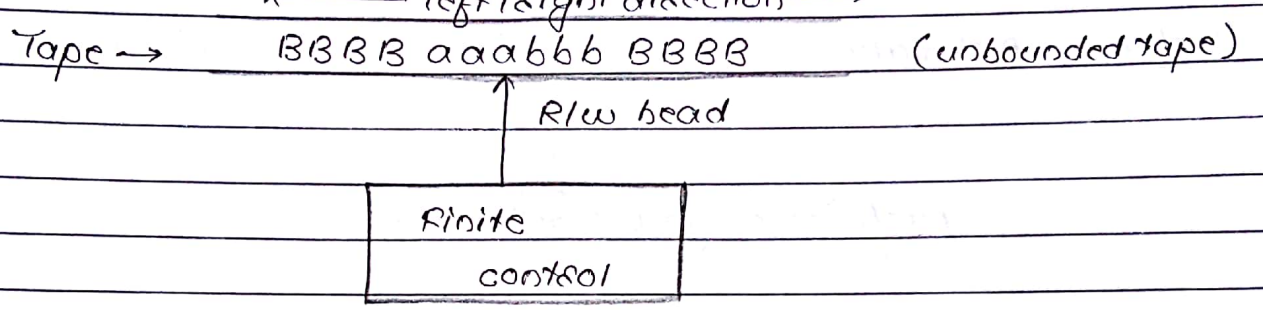




Configuration :

PDA + n-stack $\cong$ PDA + 1 stack $\cong$ TM $\cong$ FA + 2 stacks $\cong$ FA + n (≥ 2) stacks,
(≥ 1) (2 stack PDA).

FA + R/W head + L/R direction $\cong$ Turing Machine



Linear Bounded Automata

It is an Turing machine but tape is restricted (bounded) to length of input.

If $|w| = n$, then size of tape = $O(n)$.

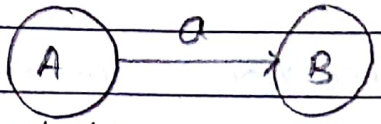
A Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$ where Q, Σ, Γ are all finite sets and

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. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ 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

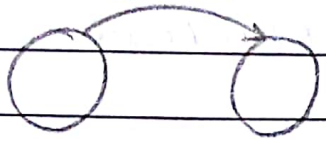
$$\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$$

Finite Automata



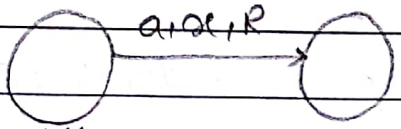
change state

Push Down Automata



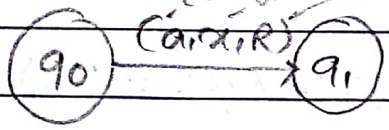
push some symbol onto stack

Turing Machine



write with some other symbol.

I. Replace 'a' by 'α'.

i. 

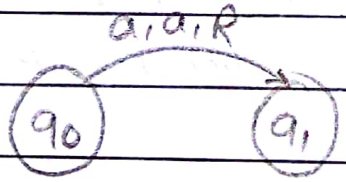
ii. $\delta(q_0, a) = (q_1, \alpha, R)$

$Q \times \Gamma = Q \times \Gamma \times \{L, R\}$

iii.

δ	a
q_0	(q_1, α, 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 → replaced by α $\quad\quad$ $\alpha : \alpha$

b → replaced by γ $\quad\quad$ $\not{b} : \gamma$

III. q_0 : $\alpha\ \alpha$ $\quad\quad$ q_3 : skip 'Y' to reach B.

q_1 : $\not{b}\ \gamma$

q_2 : reverse scan

IV. ..... B a a a a b b b b B .....

α a a a γ b b b

$\alpha\ \alpha$ a $\gamma\ \gamma$ b b

$\alpha\ \alpha\ \alpha$ a $\gamma\ \gamma\ \not{b}$ b

$\alpha\ \alpha\ \alpha\ \alpha\ \gamma\ \gamma\ \gamma$

V. .. B. $\not{\alpha}$ a a a a b b b b B .....

$q_0 \rightarrow q_1$ skip a $\rightarrow q_1$ $\quad$ I st scan

q_2

skip a

$q_2 \rightarrow q_0$

α $\quad\quad$ γ

... B α $\not{a}$ a a γ $\not{b}$ b b B

q_0 q_1 skip a, γ

q_1

skip γ, a q_2

$q_2 \rightarrow q_0$

γ

B α α α α γ γ γ $\not{b}$ B $\quad$ Last seen

$q_2 \leftarrow q_2$ q_1

$\rightarrow q_0$

q_0:

i. $\not\alpha\, x$ -and move to q_1

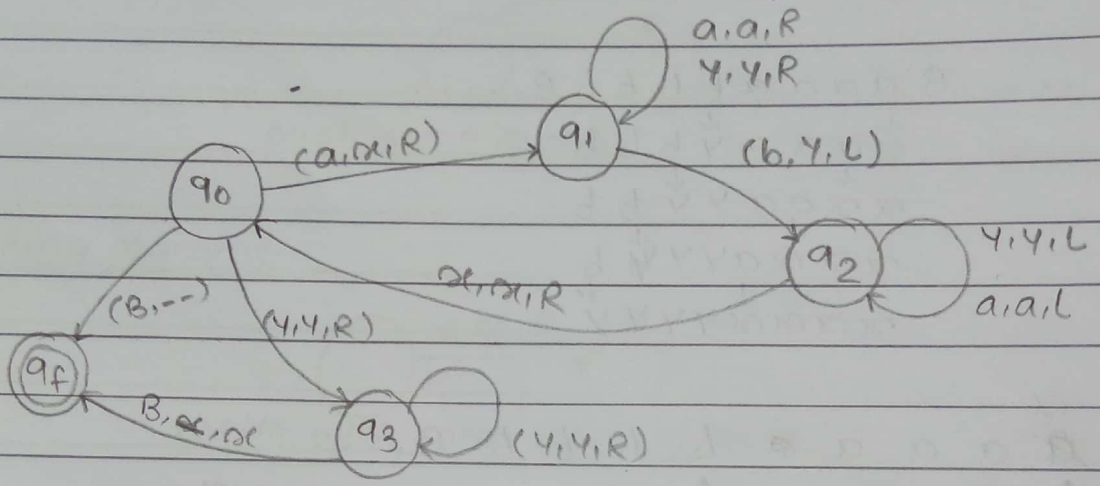
ii. when it sees 'Y' move to q_3.

q_1:

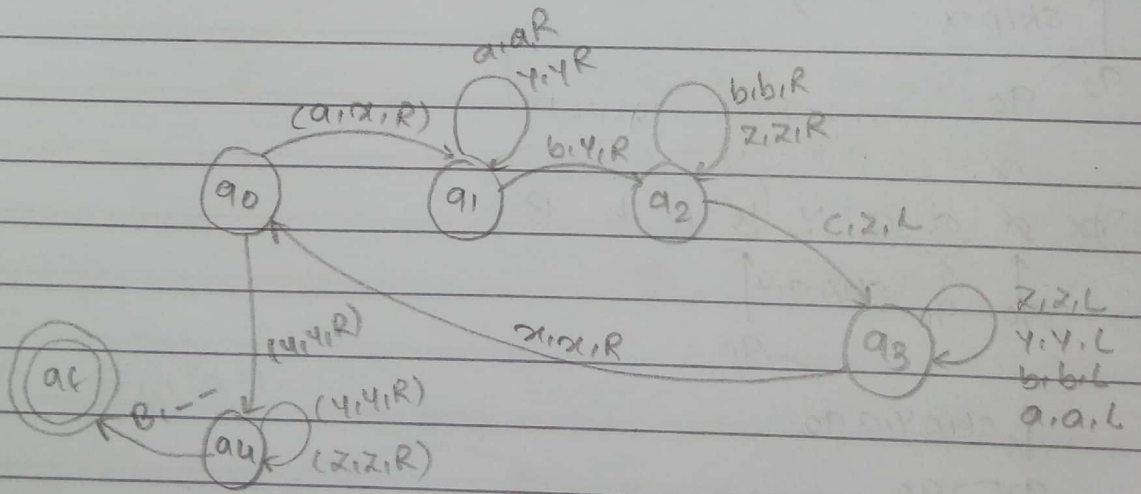
i. skip a, Y

ii. Replace 'b' with 'Y' and move to q_2 in left direction.

q_2: skip Y, a in left direction to find first 'x'.



2. $\{a^n b^n c^n \mid n \geq 1\}$



q_0: $\not\alpha\, x$

q_1: $b \not\alpha\, Y$

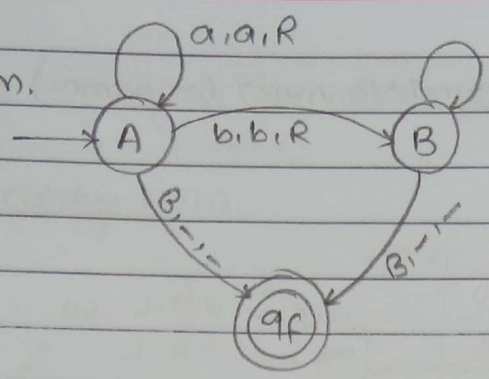
q_2: $\not\alpha\, Y$

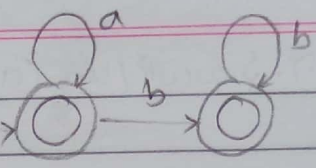
q_3: reverse direct^n

q_4: skip Y & Z to reach B.

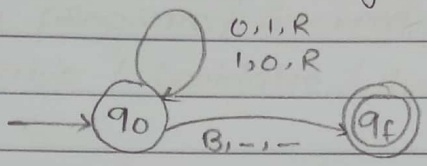
3. a^*b^*

T.M.





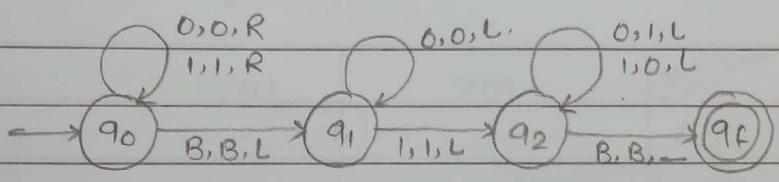
4. 1's complement of a binary number.



... B 1 0 1 0 1 1 B ...

0 1 0 1 1

5. 2's complement of a binary number.



Given: 1 0 1 0 0

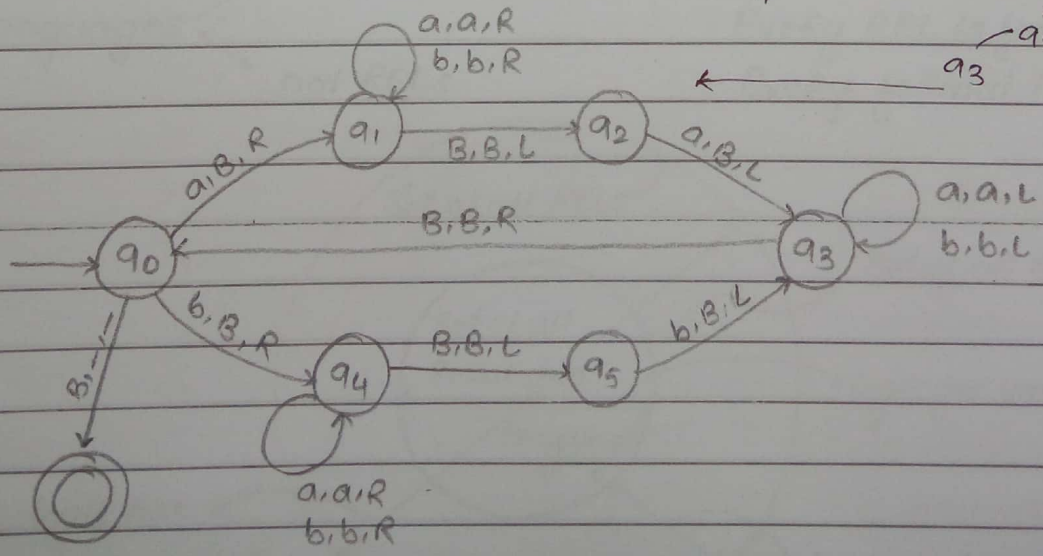
0 1 0 1 1 → 1's C

0 1 1 0 0 → 2's C.

1's C Same

6. $\{ ww^R \mid w \in (a+b)^* \}$

... B B a b a b b a b a B B ....



B B B b b B B

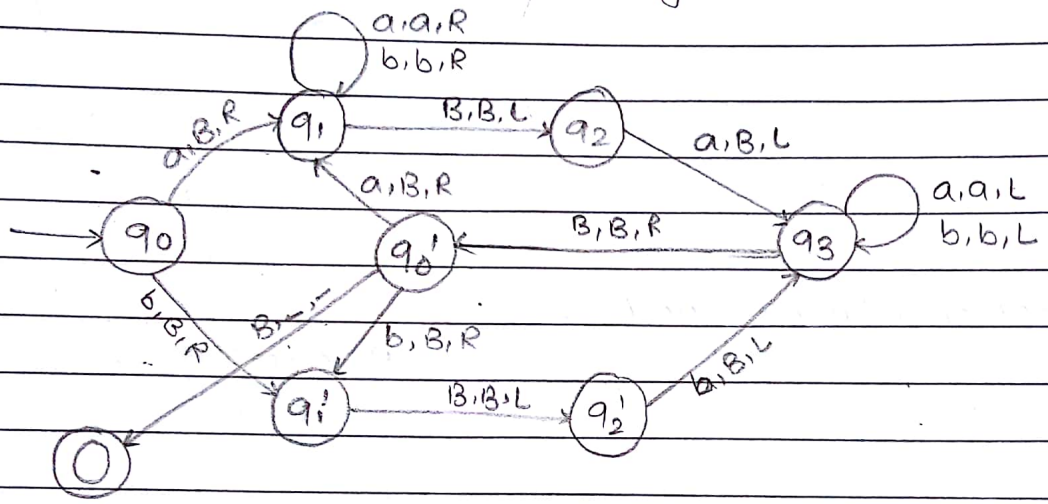
q0 - q1

q1

q2

7. $\{ww^R / w \in (a+b)^+\}$

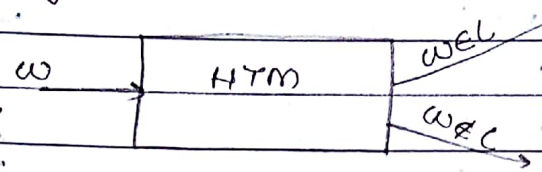
(1st and last symbols must be same)



8. $\{wcw^R / w \in (a+b)^*\}$ / $w \in (a+b)^+$

Recursive language : (Decidable language)

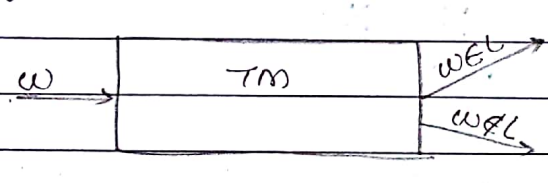
→ Accepted by HTM.



→ Halting program / Generated by algorithm / lexicographically enumerable / Decidable / Turing decidable / TM decidable.

Recursively Enumerable Language (REL)

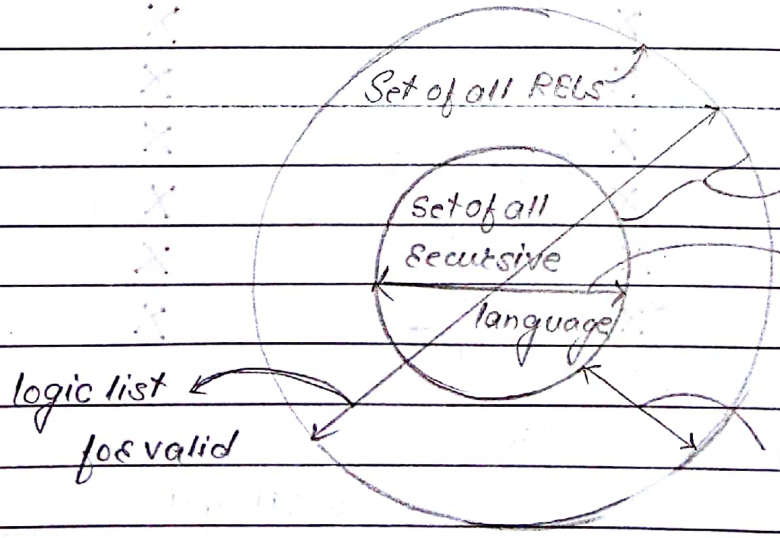
→ Accepted by TM



→ Program / Algorithm / Enumerable / Acceptable / Recognitable / computable / countable.

Languages → RELs
Languages → not RELs

Note:
- Every REL is formal language.
- Every formal language need not be REL.



Closure properties of Recursive language and REL.

	Recursive language	Recursively Enumerable language.
1. Union	✓	✓
2. Intersection	✓	✓
3. Complement	✓	✗
4. Difference	✓	✗
5. Kleene closure	✓	✓
6. positive closure	✓	✓
7. Concatenation	✓	✓
8. Reversal	✓	✓
9. Subset	✗ ✗	✗
10. Substitution	✗ ✗	✓
11. Homomorphism	✗ ✗	✓
12. ε-free Homomorphism	✓	✓
13. Finite Union	✓	✓
14. Finite Intersection	✓	✓
15. Finite Difference	✓	✓
16. Finite Concatenation	✓	✓
11! Inverse Homomorphism	✓ ✓	✓
17. Finite subset.	✓ ✓	✓
18. Finite Substitution	✓	✓
19. Infinite Union	✗	✗
20. Infinite Intersection	✗	✗
21. Infinite Difference	✗	✗
22. Infinite Concatenation	✗	✗
23. Infinite Subset	✗	✗
24. Infinite Substitution	✗	✗
25. Regular Closure		

Recursive language (Not closed)
Substitution, Homomorphism

Recursively Enumerable language (Not closed)
Complement, Difference

Note:

1. $\subseteq$ for all not closed.
2. Infinite ($\cup, \cap, \ldots$) for all not closed.
3. Inverse Homomorphism for all closed.
4. $\subseteq$ finite for all closed.

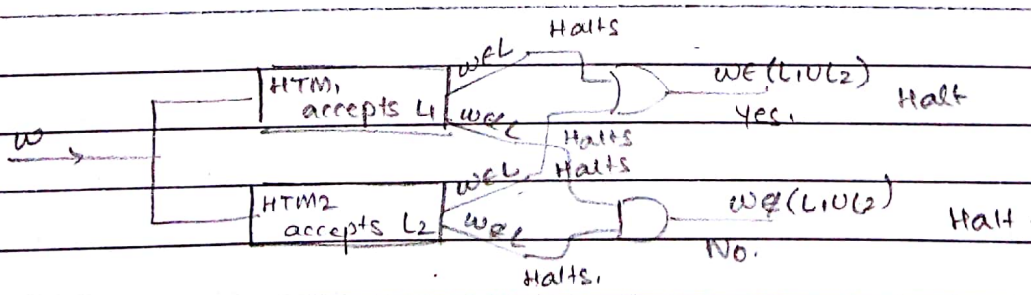
Note:

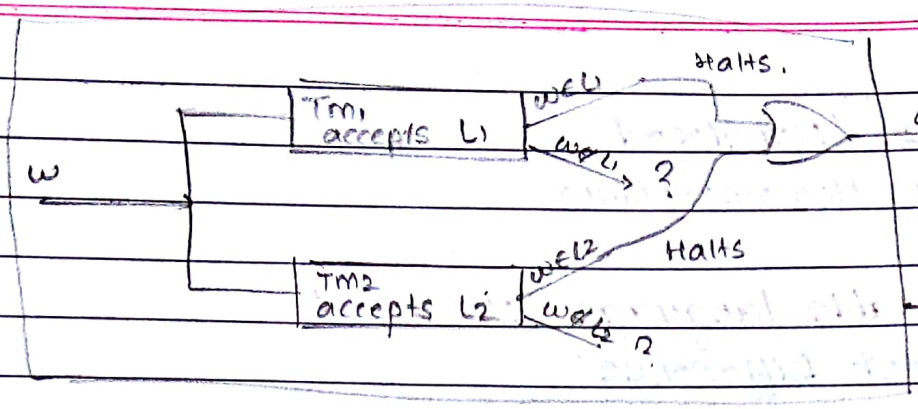
$\subseteq$	$\infty(\cup, \cap, \ldots)$	h^{-1}	$\subseteq$ fin
x	x	✓	✓

✓D: $\bar{L}$, prefix(L)
x C: $\cap, \bar{L} \rightarrow L_1 - L_2$, fin $\cap$, fin diff
x Rec: $f(L) \rightarrow h(L)$
x REL: $\bar{L} \rightarrow L_1 - L_2$

Union

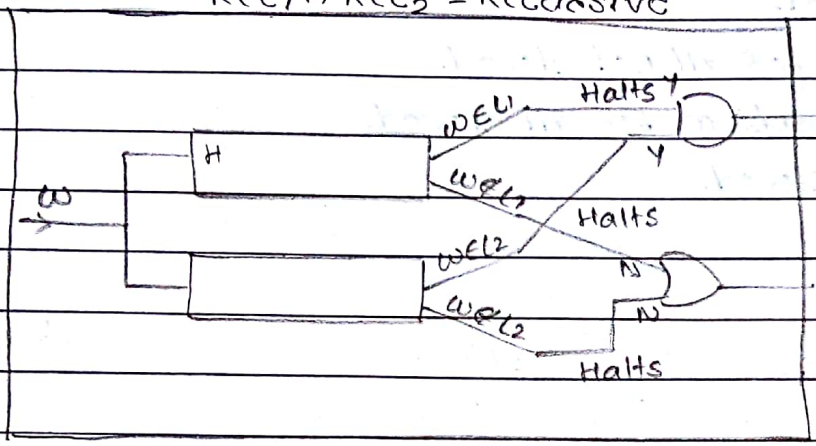
$Rec_1 \cup Rec_2 =$ Recursive $REL_1 \cup REL_2 = REL$.

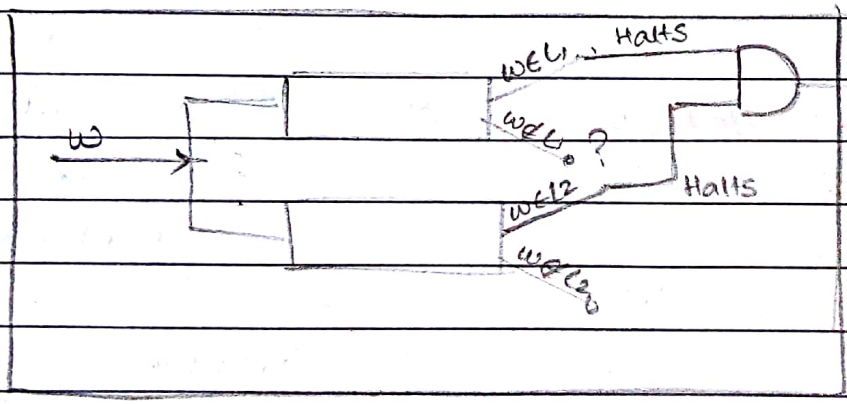




Intersection

$Rec_1 \cap Rec_2 = Recursive$

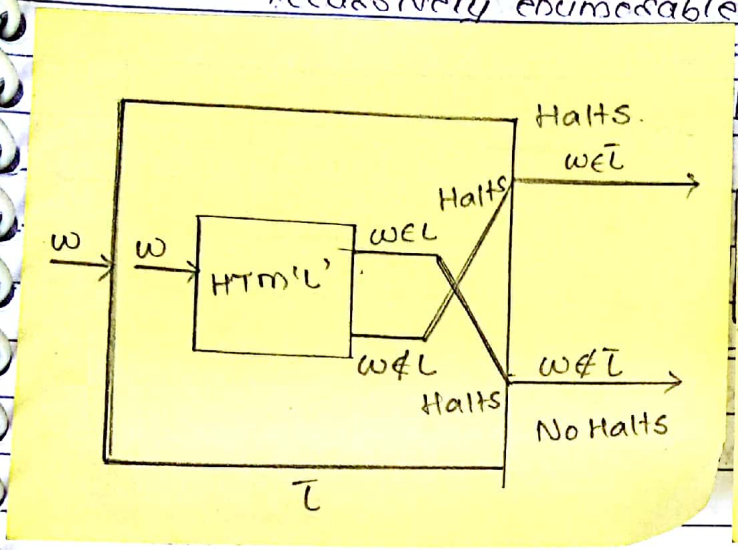


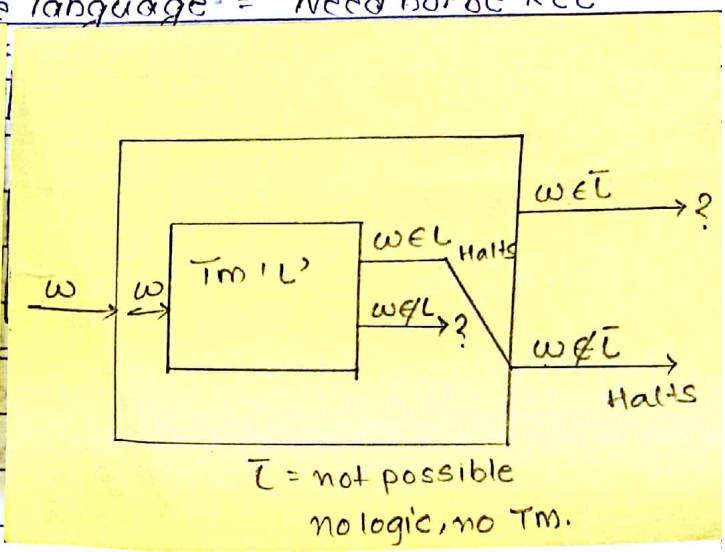


Complement :

Recursive language = Recursive

Recursively enumerable language = Need not be REL



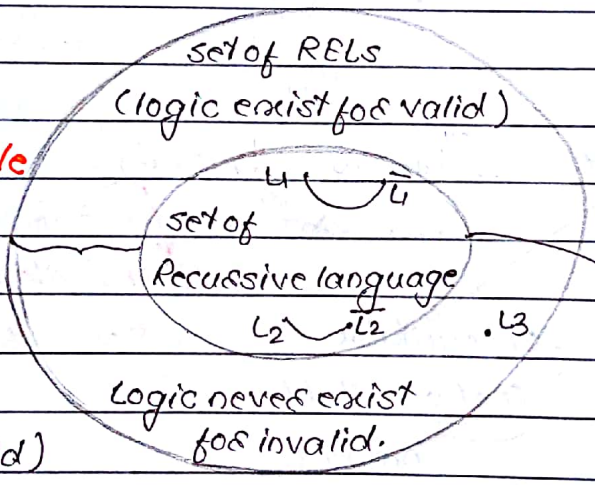


DOT RELs

D → decidable

SD → semidecidable

Set of RELs but not recursive (semidecidable) (logic exist for valid)



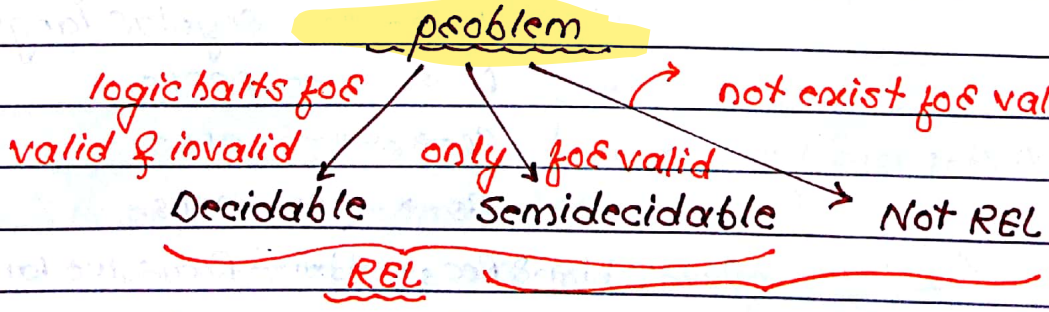
REL̄ = Need not be REL

(D or SD) = D or not REL

REL̄ = never be semidecidable

logic exist for valid and invalid.

L̄3 (logic never exist valid of L̄3)



Every not REL → undecidable

Undecidable → not or need not be SD, REL.

Types of Turing Machine :

Turing Machine ≅ single tape turing machine

≅ One way tape turing machine (unbounded)

≅ Multi-tape turing machine

≅ Multi-head turing machine

≅ Multi-tape & Multi-head turning machine

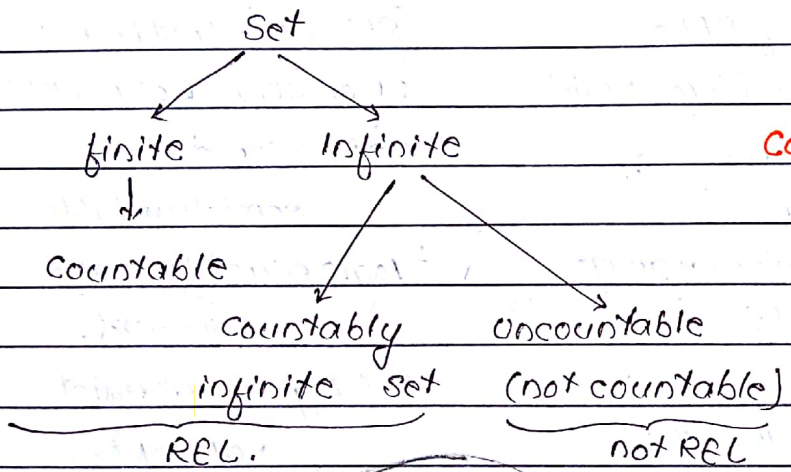
≅ Multi-dimensional tape turing machine

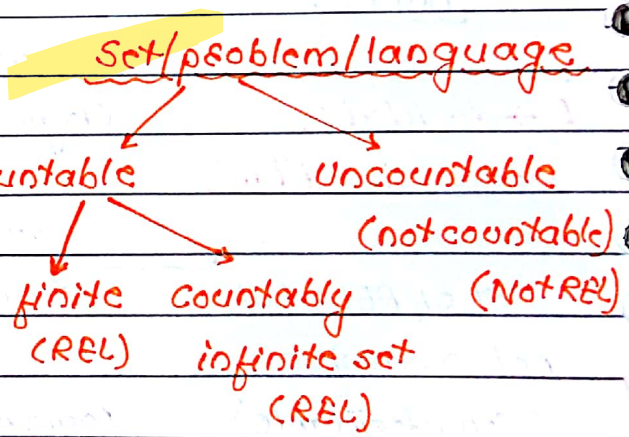
≅ two stack PDA

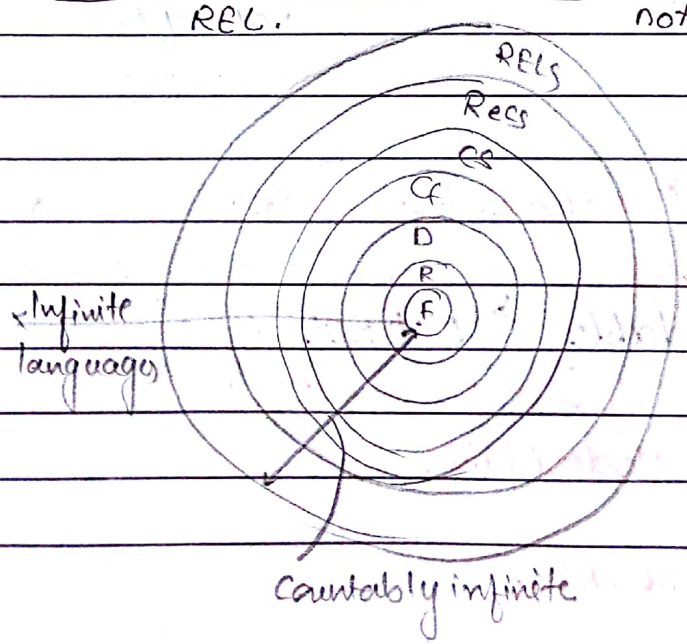
≅ Counter machine

≅ Universal turing machine (programmable)

Countable and Uncountable







F → set of finite language

R → —"— regular language

D → —"— DCFLs.

Cf → —"— CFLs.

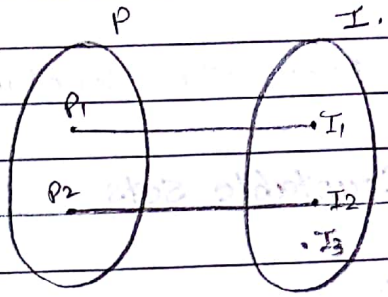
Cs → —"— CSLs.

Recs → —"— Recursive language

RELs → —"— RELs.

Countably Infinite Set:

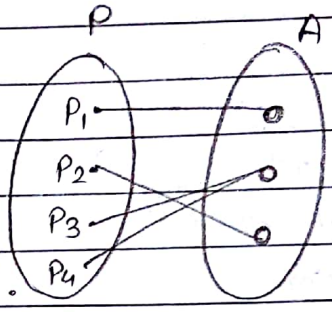
Injective function



Every two elements of P(a,b), should have different images in I.

$\forall a, b \in P$, $a \neq b$ $\quad f(a) = I_1$, $f(b) = I_2$ $\quad \Big\} \; I_1 \neq I_2$

Surjective function



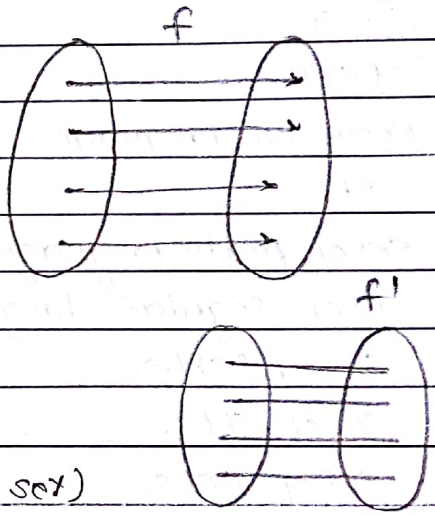
$\forall a \in A, \exists p \in P$

$f(p) = a$ $\quad$ pre-image

Every element of codomain has atleast one pre-image.

Bijective function (one to one correspondence) (invertible)

$f \rightarrow A$ $\quad f: A \rightarrow B$

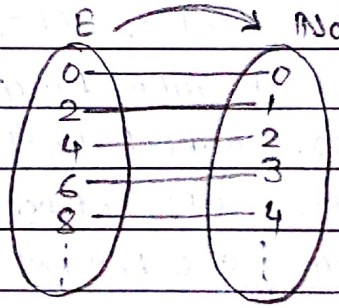


- f is injective and surjective.
- f and f^{-1} are functions.
- f is a function, every codomain element has exactly one pre-image.

Set of Natural numbers (countably infinite set)

$I = \{0, 1, 2, \ldots\} \rightarrow No$

$= \{1, 2, \ldots\} \rightarrow N$



X is countably infinite set iff

$f: X \rightarrow N$ is bijective

where N can be any known countably infinite set.

$f(x) = x/2$

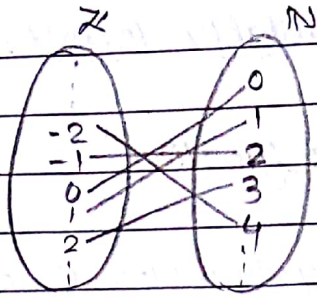
$x \in E$, $f'(x) = 2x$

$x \in No$

$x \in \mathbb{Z}$ $f(x) = 0 ; x = 0$

$= 2x - 1 ; x = +ve$

$= -2x ; x = -ve$



"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 (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. Cartesion 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. {L | L! DCFL but not regular}

Equivalence

- **Recursive language** ≅ Decidable ≅ Lexicographically Enumerable
  ≅ Total function
  ≅ REL but not undecidable
  ≅ REL but not semidecidable
  ≅ Countable but no semidecidable
  ≅ Computable but not semidecidable

- **REL but not recursive** ≅ semidecidable
  ≅ partial decidable ≅ REL but not decidable
  ≅ partial function but no total
  ≅ countable but not decidable
  ≅ undecidable but REL
  ≅ undecidable but partial decidable.

- **Not Recursively Enumerable language** ≅ Uncountable
  ≅ undecidable but not semidecidable
  ≅ undecidable but not REL.
  ≅ no. function ≅ not computable
  ≅ not Enumerable
  ≅ not acceptable

- **Recursively Enumerable language** ≅ countable
  ≅ computable
  ≅ partial function
  ≅ enumerable
  ≅ Turing recognitable
  ≅ either decidable or semidecidable.

- Undecidable ≅ not decidable ≅ either semidecidable or not REL
  ≅ not recursive. ≅ not total function

Recursive language:

1. L → 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.

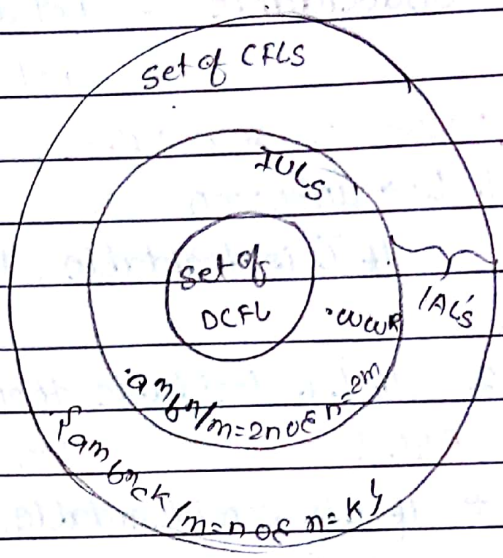
undecidable < semidecidable
undecidable < not REL

complement undecidable < complement semidecidable → not REL
complement undecidable < complement (not REL) < semidecidable
complement undecidable < complement (not REL) < not REL

- If $\bar{L}$ is undecidable then L is undecidable.

Inherently Unambiguous Language

If there exist unambiguous CFG for any language, then it is Inherently Unambiguous language (IUL).



Find language generated by CFG?

$S \to A/a$ $\quad L=\{a\} \to$ IUL

$A \to a$ $\quad$ L produced by given CFG has some unambiguous CFG, so, L is IUL.

Eg. $\{a^m b^n / m=2n$ or $n=2m\}$ → (CFL but not DCFL)

$a^{2n}b^n$ $\quad$ $a^n b^{2n}$

$A \to aaAb/\varepsilon$ $\quad$ $B \to aBbb/\varepsilon$

$S \to A/B$ $\quad$ ∴ unambiguous CFG.

$A \to aaAb/\varepsilon$

$B \to aBbb/\varepsilon$

- WW^R, $S \to aSa/bSb/\varepsilon$. → 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

$\{a^m b^n c^k / m=n$ or $n=k\}$

$a^n b^n c^*$ $\quad$ $a^* b^n c^n$

$S \to AB/CD$

$A \to aAb/\varepsilon$

$B \to cB/\varepsilon$

$C \to aC/\varepsilon$

$D \to bDc/\varepsilon$

Note: Every CFG is Ambiguous

Regulas closuse	Regs(L)	DCFLs(L)	CFLs(L)	CSLs(L)	Recs(L)	RELs(L)
L ∪ Reg	Reg	DCFL	CFL	CSL	Rec	REL
L ∩ Reg	Reg	DCFL	CFL	CSL	Rec	REL
L − Reg	Reg	DCFL	CFL	CSL	Rec	REL
Reg ∪ L	Reg	DCFL	CFL	CSL	Rec	REL
Reg ∩ L	Reg	DCFL	CFL	CSL	Rec	REL
Reg − L	Reg	DCFL	?	CSL	Rec	?

$$\text{Reg} - \text{DCFL} = \text{Reg} \cap \overline{\text{DCFL}}$$
$$= \text{Reg} \cap \text{DCFL}$$
$$= \text{DCFL}.$$

$$\mathcal{X} \rightarrow \begin{matrix} \cup \\ \cap \\ - \end{matrix} \text{Reg} = \mathcal{X}.$$

Simplified CFG:

If CFG has i. No unit ii. No Null iii. No useless productions then it is simplified CFG.

I. Reduced CFG:

If CFG has no useless productions then it is reduced.

II. Simplication of CFG: (CFG → simplified CFG)

Step 01: Delete Null production

Step 02: Delete unit production

Step 03: Delete useless production.

III. Deleting Null production $a(\mathcal{X} \rightarrow \varepsilon)$

1. $S \rightarrow a|\varepsilon$ "If many NULL productions, delete one by one."

⇓ delete $S \rightarrow \varepsilon$

$S \rightarrow a$ but empty string not possible in result, so that it should be noted.

2. $S \to AaIAb$ $A \to bI\varepsilon$ ⇒ delete ε $S \to AaIAbIaIb$ $A \to b$

"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: ($x \to Y$ is unit production)

1. $S \to AIAb$ $A \to aIsb$ ⇒ delete $S \to A$ $S \to AbIaIsb$ $A \to aIsb$

place all A productions in RHS side of $S \to A$.

"If many UNIT productions, delete one by one."

V. Deleting Useless productions

Step 1: Delete the productions which cannot derive any string. ($C \to Cd$)

Step 2: Delete the productions which are not reachable from start symbol.

Given:

$S \to aSIAB$
$A \to aCIaEbIg$
$B \to b$
$C \to dC$
$F \to e$

Step 1: Find productions which cannot derive any string.
'C' cannot derive & 'E' also.

So delete all productions where C & E are present.

$S \to aSIAB$, $A \to g$, $B \to b$, $F \to e$

Step 2: Delete from productions which are not reachable.

Reachable set = { S, A, B }

So, F is not reachable.

$S \to aS/AB$, $A \to g$, $B \to b$

Null productions : $\alpha \to \varepsilon$ $\alpha() \{ \}$

UNIT productions : $\alpha \to Y$ $\alpha() \{ Y(); \}$

USELESS productions : $\alpha \to a\alpha$ $Y \to ab$

α never ends Y is not reachable from S.

Examples:

01. $S \to aA \Rightarrow$ simplified CFG = { } $L = \phi$

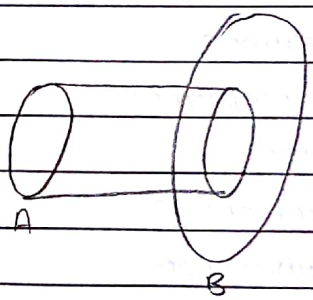
02. $S \to aA/bB \Rightarrow$ simplified CFG = $S \to aA$, $A \to a$

$A \to a$ $\therefore L = \{aa\}$

Reducibility

$A \leq B$

A is reducible to B.



B is atleast 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/reg/DCFL/CFL/CSL/rec/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) → 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) → 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 the A is also regular (may finite/infinite).

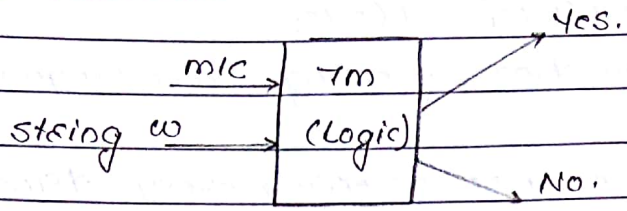
12. If $\overline{B}$ is decidable (B is decidable) then A is decidable.
13. If $\overline{A}$ is undecidable (A is undecidable) then B is undecidable.

Decidable and Undecidable problem

- Membership problem
  Emptyness
  Non-emptyness
  Finiteness
  Infiniteness
  Equivalence
  Totality
  Containment
  Ambiguity
  Halting
  disjoint.

"
Finite automata (reg lang) (reg gra) (reg. exp)
DPDA (DCFL)
PDA (CFL) (NCFL) (NPDA) (CFG)
HTM (recursion) (decidable lang)
TM (REL) "

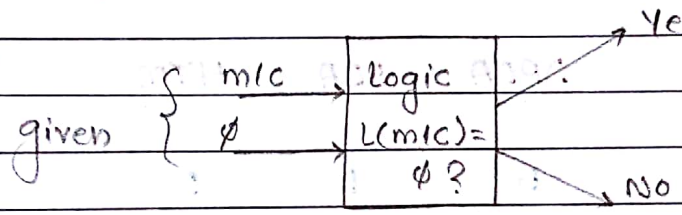
Membership problem



Membership for machine
Is w accepted by machine?
Is $w \in L(mlc)$?
Is $w \in L(grammar)$?
Is $w \in L$?
membership for corresponding grammar
language

} Same

Emptiness problem



Is given machine accepts empty language ?
Is $L(mlc) = \phi$?
Is $L(grammar) = \phi$?
Is given machine rejecting every string ?

Non-Emptiness problem

Is $L(mlc) \neq \phi$?
Is given machine accepting atleast one string ?

Finiteness problem:

Is L(mlc) = finite ?

Infinite(ness) problem:

Is (mlc) = Infinite ?

Equivalence problem

Is $L(m/c_1) = L(m/c_2)$

Is given two machines accepting same language?

Totality problem

Is given machine accepting every string?

Containment problem (subset checking)

Is $L(m/c_1) \subseteq L(m/c_2)$?

$L_1 \subseteq L_2$ iff

1. $L_1 \cap \overline{L_2} = \phi$ 2. $L_1 - L_2 = \phi$

Ambiguity problem

Is grammar ambiguous?

Halting problem

Is given machine halts?

Disjoint problem

Is $L(m/c_1) \cap L(m/c_2) = \phi$

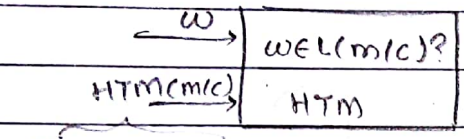
Problems	FA	DPDA	PDA	HTM	TM
• Membership		D	D	D	
• Emptiness		D	D		
• Non-Emptiness		D	D		
• Finiteness		D	D		
• Infiniteness		D	D		
• Equivalence		D	≠		
• Totality		D			
• Containment					
• Ambiguity					
• Halting		D	D	D	
• Disjoint					

Decidable (HTM exist)

undecidable (HTM not exist)

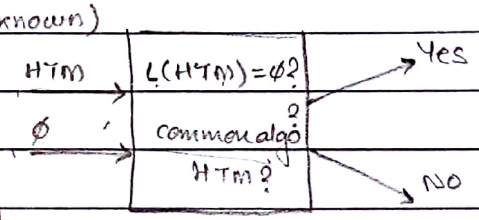
Membership ⇒ CYK algorithm, (FA, DPDA, PDA)
passing, Bottom-up passing, O(n3), dynamic programming

Algo.1 ⇒ FA, PDA, DPDA ⇒ CFG ⇒ Chomsky Normal Form (CNF) CFG ⇒ Dynamic programming

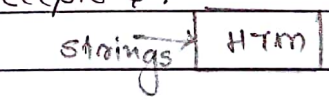


Is prog1 ≅ prog2 ?
Is given two programs logically equivalent?
Equivalence problem of TM.

Emptiness problem
↳ Can I verify any given HTM that accepts ϕ?



Can I design HTM that accepts ϕ?



No HTM.

Emptiness / Non-emptiness
FA, DPDA, PDA ⇒ CFG ⇒ simplify
- Atleast 1 product L≠ϕ
- No product L=ϕ

Finiteness / Infiniteness (dependency graph)
FA, DPDA, PDA ⇒ CFG ⇒ dependency graph ⇒ check loops

Equivalence

Is $FA_1 \cong FA_2$?

$FA_1 \Rightarrow$ min DFA$_1$ } Isomorphic (same)

$FA_2 \Rightarrow$ min DFA$_2$ }

Containment

Is $R_1 \subseteq R_2$?

$R_1 \cap \overline{R_2} = \phi$

DPDA / PDA

Is $D_1 \subseteq D_2$

$D_1 \cap D_2 = \phi$?

Is $HTM = \phi$?

HTM

Is $Rec_1 \subseteq Rec_2$

$Rec_1 \cap Rec_2 = \phi$?

Is $(HTM) = \phi$?

Finite automata $\Rightarrow$ all decidable

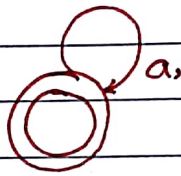
Turing machine $\Rightarrow$ all decidable

FA, PDA, DPDA (decidable)

{ mem, ϕ, not ϕ, fin, Inf

FA, DPDA (decidable)

{ totality, equivalence



FA, DPDA, PDA, HTM (decidable)

mem, halting

Decidable, Semidecidable & REL(NOT REL)

Decidable, Semidecidable, Not Recursively Enumerable Languages

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 regulars
↓ no logic
L is not REL.

Undecidable → Semidecidable, NOT REL.

01. Finite language $-L_1$ Decidable ($L_1 \to$ finite $\to$ TM, $\bar{L} \to$ TM, $L_1 \to$ HTM)
02. Language over $\Sigma = \{a,b\}$ (finite alphabet) $- L_2$
$L \subseteq \Sigma^*$ over $\Sigma = \{a,b\}$ Decidable
03. Language over ($\Sigma = \{a,b,c,\ldots\}$) finite alphabet $- L_3$ Not REL
$\Sigma^* = \{\varepsilon, a, b, aa, \ldots\}$
04. Σ^* over $\Sigma = \{a,b\} - L_4$-Reg Decidable
$\bar{L}_4$-Reg. $\exists$ HTM
05. Σ^* over $\Sigma = \{a,b,c,\ldots\} - L_5$ Not REL

06. Regular language — L_6. — Decidable
07. CFL — L_7 — Decidable
08. CSL — L_8 — Decidable
09. Recursive language — L_9 — Decidable

} Trivial.

10. REL — L_{10} — $L_{10} \rightarrow$ TM — Semidecidable
$\overline{L_{10}} \rightarrow$ no TM.
$\overline{REL}$ = 0 or not REL.
nologic nologic

11. Set of all regular languages. — L_{11}. → Semidecidable
$L_{11} \rightarrow$ logic, $\overline{L_{11}} \rightarrow$ no logic
12. Set of all CFLs. — L_{12} → Semidecidable

13. Set of all CSLs. — L_{13} → Semidecidable

14. Set of all DCFLs. — L_{14} → Semidecidable

15. Set of all recursive languages. — L_{15} → Semidecidable

16. Set of all RELs. — L_{16} $\overline{L_{16}} \rightarrow$ no logic → Semidecidable
17. Set of ~~all~~ DCFLs but not regulars. — L_{17} → Semidecidable
$\overline{L_{17}} \rightarrow$ all regulars and all not DCFLs
no logic

18. Set of CFLs but not DCFLs. — L_{18} → Semidecidable

19. $\{ \langle DFA \rangle \mid L(DFA) = \phi \}$ → decidable

20. $\{\langle NFA\rangle / L(NFA) = \phi\}$ ⟶ decidable

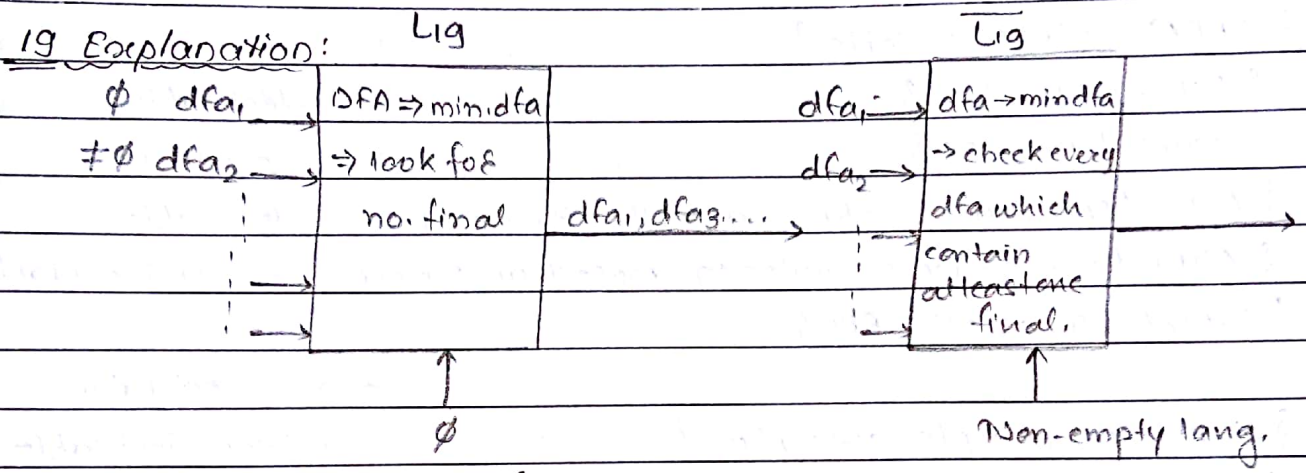
21. $\{\langle$ Regular Expression $\rangle / L($Regular Expression$) = \phi\}$ ⟶ decidable

22. $\{\langle LLG_1\rangle / L(LLG_1) = \phi\}$ ⟶ decidable

23. $\{\langle RLG_1\rangle / L(RLG_1) = \phi\}$ ⟶ decidable

24. $\{\langle$ Regular Grammar $\rangle / L($Regular Grammar$) = \phi\}$ ⟶ decidable

19 Explanation:



25. $\{\langle DFA\rangle / L(DFA) \neq \phi\}$ ⟶ decidable

26. $\{\langle DFA_1, DFA_2\rangle / L(DFA_1) = L(DFA_2)\}$ ⟶ decidable
Equivalence

27. $\{\langle DFA\rangle / \langle DFA\rangle =$ finite$\}$ finitness ⟶ decidable

28. $\{\langle DFA\rangle / L(DFA) = \Sigma^*\}$ totality ⟶ decidable

29. $\{\langle DFA_1, DFA_2\rangle / L(DFA_1) \subseteq L(DFA_2)\}$ ⟶ decidable
containment

30. $\{\langle DFA_1, DFA_2\rangle / L(DFA_1) \cap L(DFA_2) = \phi\}$ disjoint ⟶ decidable

31. $\{\langle PDA\rangle / L(PDA) = \phi\}$ ⟶ decidable

32. $\{\langle DFA\rangle /$ DFA accepts string 'w'$\}$ ⟶ decidable
$w \in L(DFA)$

33. $\{\langle PDA\rangle /$ PDA accepts string 'w'$\}$ ⟶ decidable

34. $\{\langle CFG_1\rangle / CFG_1 \text{ is ambiguous}\}$ → Semidecidable

35. $\{\langle CFG_1\rangle / CFG_1 \text{ is not ambiguous}\}$ → Not REL

36. $\{\langle PDA_1, PDA_2\rangle / L(PDA_1) = L(PDA_2)\}$ → Not REL

$L(PDA_1) \neq L(PDA_2)$ → Not REL

37. $\{\langle DFA\rangle / DFA \text{ accepts regulas}\}$ $\bar{L} = \phi$ → decidable

38. $\{\langle PDA\rangle / PDA \text{ accepts CFL}\}$ → decidable

39. $\{\langle DFA\rangle / L(DFA) = a^*\}$ → decidable

40. $\{\langle PDA\rangle / L(PDA) = a^*\}$ $\bar{L}$ → not REL → Not REL

41. $\{\langle TM\rangle / \text{size of TM} = 3\}$ → decidable

42. ~~$\{\langle TM\rangle /$~~ no. of states

42. $\{\langle PDA\rangle / L(PDA) = \text{finite}\}$ → decidable

43. $\{\langle TM\rangle / |TM| \geq 3\}$ → decidable

44. $\{\langle TM\rangle / |TM| \leq 3\}$ → decidable

45. $\{\langle TM\rangle / TM \text{ reaches states 'q' within 3 moves}\}$ (infinite computation exist) → decidable

46. $\{\langle TM\rangle / TM \text{ reaches states 'q' more than 3 moves}\}$ → Semidecidable

47. $\{\langle TM\rangle / TM \text{ accepts REL}\}$ $\bar{L} = \phi$

→ decidable

48. $\{\langle TM\rangle / TM \text{ accepts string 'w'}\}$ $w \in L(TM)$ $\bar{L}$ = no. L → semidecidable

49. $\{\langle TM\rangle / TM \text{ accepts string '01'}\}$ $01 \in L(TM)$ → semidecidable

50. $\{\langle TM\rangle / TM \text{ accepts only string '01'}\}$ $L(TM) = \{01\}$ → NOT REL ($\bar{L}$ also no logic)

51. $\{\langle TM\rangle / TM \text{ accepts some string}\}$ $L(TM) \neq \phi$ → Semidecidable

52. $\{\langle TM\rangle / TM \text{ accepts ~~some~~ every string}\}$ $L(TM) = \Sigma^*$ → Not REL

53. $\{\langle TM\rangle / L(TM) \leq \{00, 111\}\}$ → NOT REL.

54. $\{\langle TM\rangle / L(TM) \nsubseteq \{00, 111\}\}$ other than 00, 111. → Semidecidable

55. $\{\langle TM\rangle / TM \text{ accepts some string of length 2014}\}$ ($\bar{L}$ = NE logic) → Semidecidable.

56. $\{\langle TM\rangle / L(TM) = \text{Regulas}\}$ → No logic → NOT REL

57. $\{\langle TM\rangle / L(TM) = a^*\}$ → NOT REL

58. $\{\langle TM\rangle / L(TM) = CFL\}$ → NOT REL

59. $\{\langle TM\rangle / L(TM) = DCFL\}$ → NOT REL

60. $\{\langle TM\rangle / L(TM) = CSL\}$ → NOT REL

61. ~~$\{\langle TM\rangle /$~~

$\{\langle TM_1, TM_2\rangle / L(TM_1) = L(TM_2)\}$ → NOT REL.

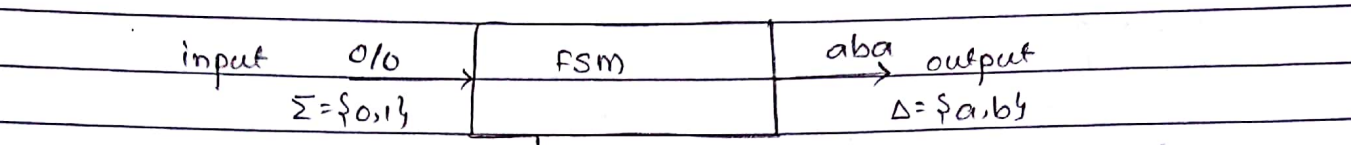
62. (51). $\{\langle TM\rangle / L(TM) = \phi\}$ → NOT REL

52. $\bar{L} = \{\langle TM \rangle / L(TM) \neq \Sigma^*\}$ no logic ⟶ NOT REL.

53. $L(TM) = \{\ \}, \{00\}, \{111\}, \{00, 111\}$ ⟶ no logic.

$\{\langle TM \rangle / TM$ halts on $\omega\}$ ⟶ semidecidable. $\bar{L}$ → No. logic → Not REL

Finite State machine without Output:

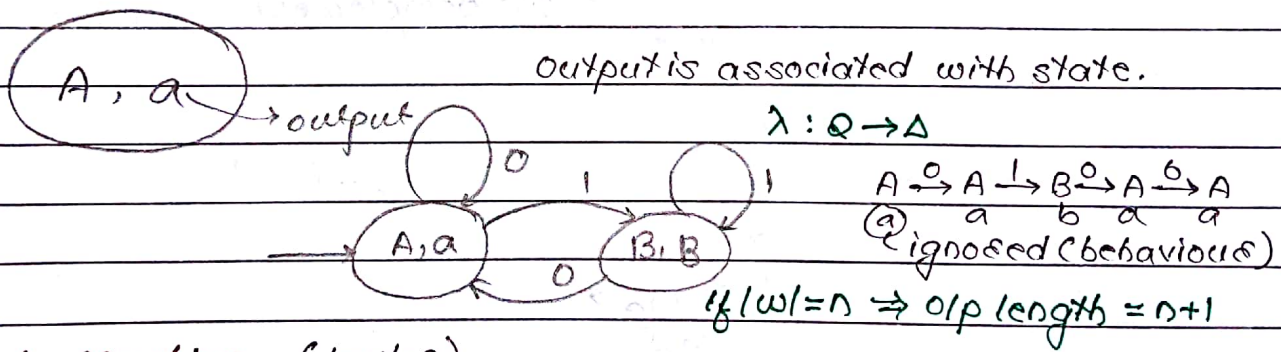


same for both
- ⟶ deterministic Finite automata
- ⟶ No. final state
- ⟶ Moore and Mealy machine
- ⟶ $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$ (output, O/P alphabet)

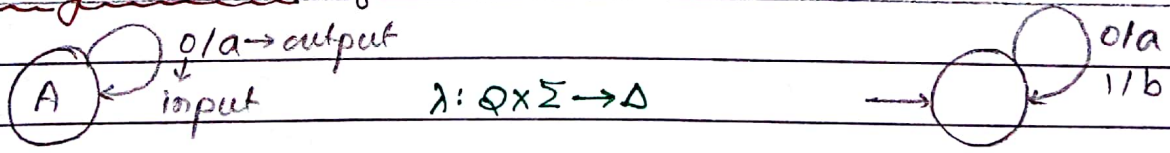
$\delta: Q \times \Sigma \to Q$ deterministic (transition function)

⟶ Both machine are deterministic.

Moore Machine



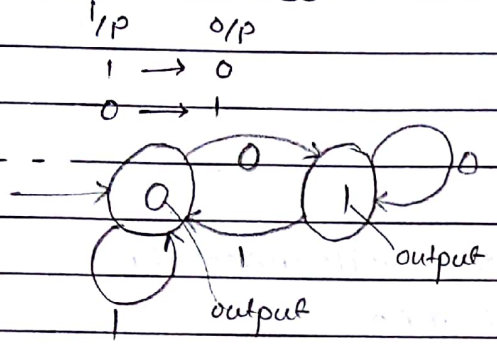
Mealy Machine (faster)



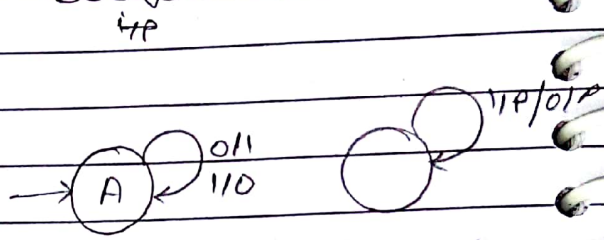
Construction of FSM with output:

01. 1's complement of binary number.

Moore machine

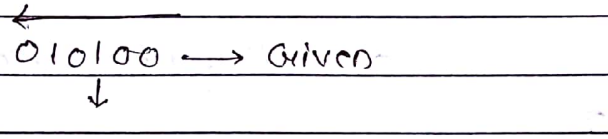


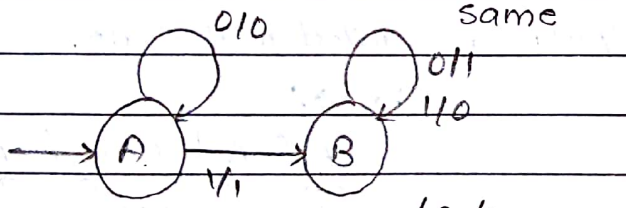
Mealy machine



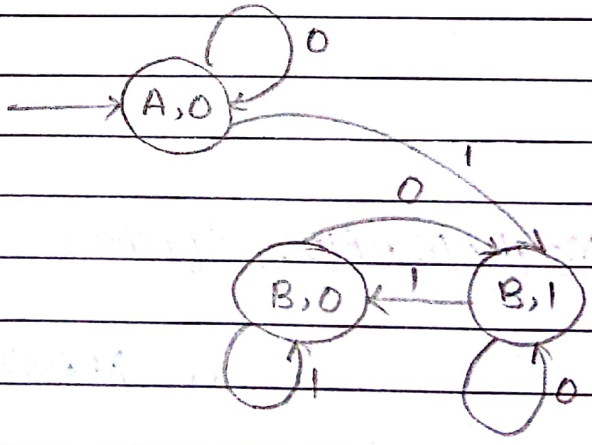
02. 2's complement

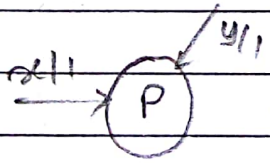
Note: Give an input in reverse order.



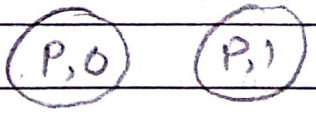


Look incoming o/p's





→ Mealy



→ Moore

Applications:

- Designing digital circuits.
- 1's complement
- 2's complement
- Addition of any two binary nos.
- Increment
- Decrement
- Subtractor
- Adder
- Comparator, etc
- Multiplication of any binary number with any constant.
- Counting sequence

03. Addition of two binary numbers.

without carry ⇒

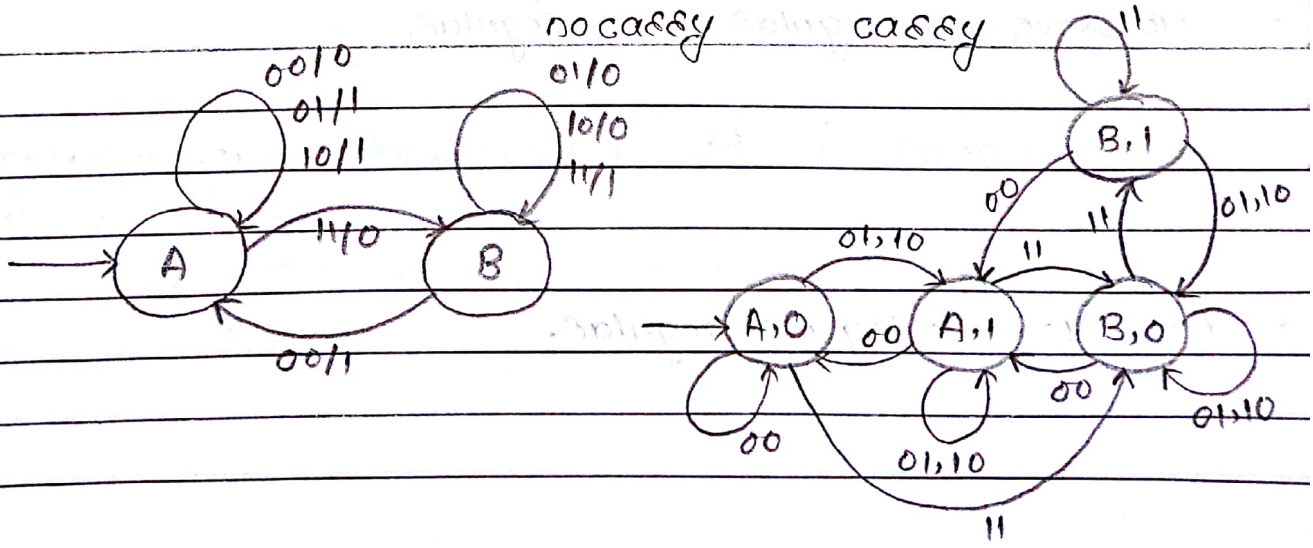
0	0	1	1
0	1	0	1
0	1	1	0 → carry

xy/z
- x: digit of 2nd no
- y: digit of 1st no
- z: output bit.

with carry ⇒

0	0	1	1
0	1	0	1
1	0	0	1

no carry | carry

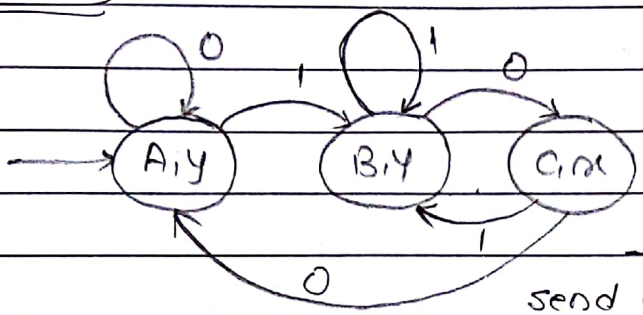


4. Count no. of 10's in binary number.

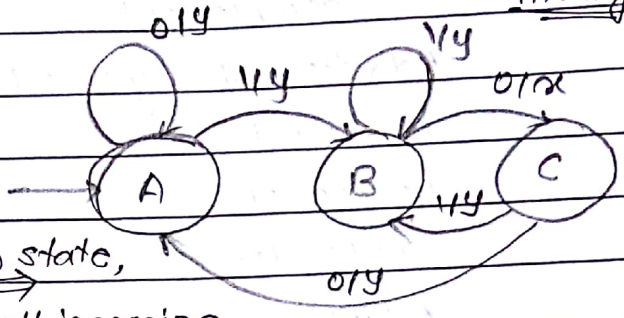
0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 0
↓ x x x x x
y

If '10' occurs produce 'x', else produce y.

Moore.



Mealy



at each state, send o/p to all incoming.

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.

Non-regular →	PL	→ why non-reg by giving some proof

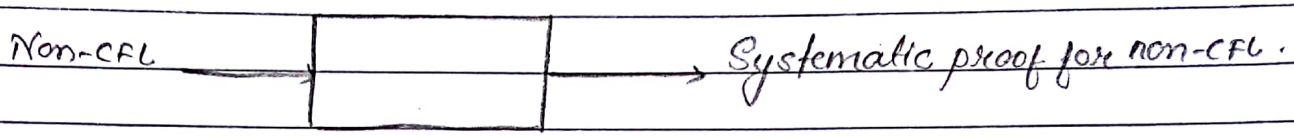
- Proof for proving not regular.

- 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.