THEORY OF COMPUTATION

Handwritten Notes

VIVEKANAND VERNEKAR

GateWallah Parakram C 2024

(Hinglish Weekday)

Other Notes Uploaded Below

https://t.me/pwgatenotes

FINITE AUTOMATA

Basics of TOC

1. Symbol
2. Alphabet
3. String
4. Language

Symbol:

* It is anything. (Symbol) — o, gate
* It is one length. — |o| = 1, |gate| = 1

		Symbol
1.	English Language	letter a, b ... z
2.	Hindi Language	letter अ, आ ...
3.	Decimal Language	digit 0, 1, 2 ... ∞
4.	Binary Language	bit 0, 1
5.	C language	char. 0 ... 9, a, b ... z, +, -1, ...

Alphabet:

* It is a set.
  - ↳ Collection of objects
    - ↳ well defined → distinct, unordered.

Alphabets in TOC

Σ → Input alphabet : Set of i/p Symbols
Δ → Output alphabet : Set of output Symbols
Γ → Stack alphabet : Set of Stack Symbols
T → Tape alphabet : Set of tape Symbols

eg: English Alphabet $\Sigma = \{a, b \cdots z, A, B \cdots Z\}$ (↳ Symbol)

String:

* It is a sequence of Symbols

$\Sigma = \{0, 1\}$

- Zero length String: λ or ε (Empty String)
- One length string: 0, 1
- two length String: 00, 01, 10, 11
- three length String: 000, 001, 010, 011, 100, 101, 110, 111

eg:: $\Sigma = \{0, 1\}$

~~How many k-length Strings?~~

$\hookrightarrow 2^k = |\Sigma|^k$

Symbols → ~~Alphabet~~ → Strings → ~~Languages~~

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

How many 100 length Strings?

$\hookrightarrow 3^{100}$.

Set: Collection of objects

ϕ Empty Set.

$\{\ \}$

$|\phi| = 0$ (set)

Size of ϕ.

String: Sequence of Symbols

ε Empty String

$|\varepsilon| = 0$ (String)

~~Size of~~ length of ε

Language [Set] over Σ

* Set
* Set of Strings
* Collection of Strings

$\Sigma = \{0, 1\}$

Language over Σ =

Language with zero String: $L = \phi = \{\}$ empty language. (Empty Set)

Language with only 1 String: $L_1 = \{\varepsilon\}$, $L_2 = \{1\}$, $L_3 = \{0\}$, $L_4 = \{00\}$ ....

Language with only 2 String: $L_1 = \{\varepsilon, 0\}$, $L_2 = \{0, 1\}$, $L_3 = \{0, 00\}$ ...

given Σ.

- No. of Strings over Σ = Infinite
- No. of language over Σ = Infinite
- No. of k length Strings = $|\Sigma|^k$
- No. of languages having 10 String = Infinite

Languages over $\Sigma = \{0, 1\}$

Finite Languages

ϕ

$\{0\}$

$\{0, 10\}$

$\{0^n 1^n \mid n \leq 10\}$

$\Sigma = \{0, 1\}$

Infinite Languages

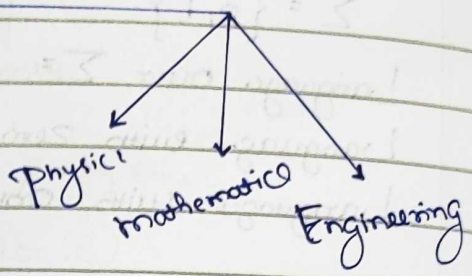
$\{0^n \mid n \geq 10\}$

$\{0^n 1^n \mid n \geq 1\}$ — form of a string.

	English	Binary N. System.	C. Language
Symbols	a, b ... z.	0, 1	Characters
Alphabet	{a, b ... z}	{0, 1}	Character Set.
Strings	words	numbers	tokens
Language.	Sentences	binary language.	Programs.

Applications of Toc:

- Compiler
- AI Applications
- Digital Circuits.
- ⋮



Operations on String.

① Reversal (Unary)

$w = abc$

$w^R = cba$

$|w| = |w^R|$

② Concatenation (Binary)

$w_1 = abc$

$w_2 = bc$

$w_1.w_2 = abbc$

$w_2.w_1 = bcab$

$|w_1.w_2| = |w_2.w_1|$

③ Prefix.

→ Start is fixed

$w = abcd$

Prefixes of w =

ε

a

ab

abc

abcd

④ Suffix

$w = abcd$

Suffixes of w:

ε.

d

cd

bcd

abcd

⑤ SubString.

$w = abcd$

Substrings of w are:

ε

4 { a, b, c, d } — ab, bc, cd } 3 — abc, bcd } 2 — abcd } 1

Prefix(w)	Suffix(w)	SubString(w)
$= \{u \mid uv = w\}$	$= \{v \mid uv = w\}$	$= \{y \mid xyz = w\}$
Beginning sequence of w.	Ending Sequence of w.	$w = abcd$ $\varepsilon . \varepsilon . abcd = w$ $\varepsilon . a . bcd = w$ ⋮

* $w = aaaa$

Prefixes of $w = \varepsilon, a, aa, aaa, aaaa$ — 5

Suffixes of $w = \varepsilon, a, aa, aaa, aaaa$ — 5

Substrings of $w = \varepsilon, a, aa, aaa, aaaa$ — 5

* $w = abcd$

Prefixes of $w = \varepsilon, a, ab, abc, abcd$ — 5

Suffixes of $w = \varepsilon, d, cd, bcd, abcd$ — 5

Substrings of $w = \varepsilon, a, b, c, d, ab, bc, cd, abc, bcd, abcd$ — 10.

How many prefixes for n-length String?

→ $n+1$

How many Suffixes for n-length String?

→ $n+1$

How many Substrings for n-length String?

→ $n+1 \leq$ No. of Substrings $\leq \frac{n(n+1)}{2} + 1$

$n+1$: If all Symbols are Same

$\frac{n(n+1)}{2} + 1$: If all Symbols in the String are distinct

How many different length prefixes for n-length String?

→ $n+1$

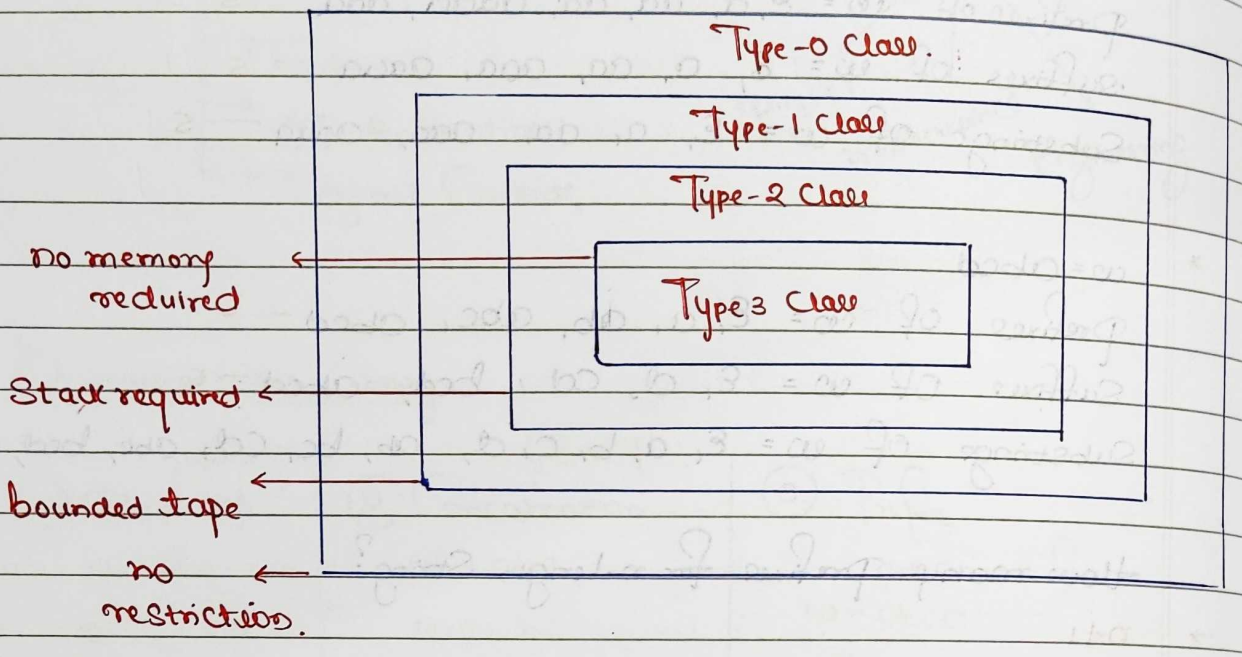
How many different length Suffixes for n-length String?

→ $n+1$

How many different Length Substrings for n-length String?

→ $n+1$

Chomsky Hierarchy: $T_3 < T_2 < T_1 < T_0$



Type 3 class	Type 2 class	Type-1 Class	Type 0 class.
↓	↓	↓	↓
Set of all regular languages.	Set of all context free languages	Set of all context Sensitive language	Set of all recursively enumerable languages.
Finite Automata (FA).	Push Down Automata (PDA)	Linear bound automata (LBA)	Turing Machine (TM)

Problems / Languages

- Computable
  - Regular
  - CFLs
  - CSLs
  - RFLs.
- Not Computable

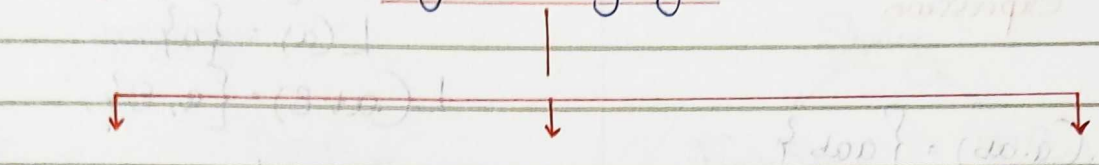
Language
↳ Set of Strings

Automata
↓ (Machine)
Set of states & transitions

Grammar.
↳ Set of rules

Regular language ⟷ FA ⟷ Regular Grammar
CFL ⟷ PDA ⟷ CFG
CSL ⟷ LBA ⟷ CSG
REL ⟷ TM ⟷ UG (REG)

Regular Languages.



Regular Expressions | Finite Automata | Regular Grammar.

Regular Expression :

* It represents a regular language.
* It uses four operators.

Binary
I. OR (+)
II. Concatenation (·)
III. Kleene Star [Kleene Closure] (*)
III. Kleene Plus [positive Closure] (+)

OR	Concatenation	Kleene Star	Kleene Plus
+, ∪, |	·	*	+
Binary	Binary	Unary	Unary
$R_1 + R_2$	$R_1 \cdot R_2$	R^*	R^+
Either R_1 or R_2	R_1 followed by R_2	Zero or more occurance of R	One or more occurances of R.
$a+b$	$a \cdot b$.	$a^* = a^{\geq 0}$.	$a^+ = a^{\geq 1}$.
$\{a, b\}$	$\{ab\}$.	$\{a^0, a^1, a^2, \ldots\}$	$\{a, aa, aaa \ldots\}$
~~ab~~, ~~ba~~, ~~aa~~	~~a~~, ~~b~~, ~~aa~~, ~~ba~~	$\{\varepsilon, a, aa, aaa \ldots\}$	

$L(a+\varepsilon) = \{\varepsilon, a\} = \{a, \varepsilon\}$

$L(a+ab) = \{a, ab\} = \{ab, a\}$

$L(a+b) = \{a, b\}$

$L(\phi) = \{\} = \phi$.

↓ empty expression

↳ empty Set.

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

↓ non empty expression

↓ non empty Set.

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

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

$L(a+\varepsilon) = \{a, \varepsilon\}$.

$L(a.ab) = \{aab\}$

↳ one String

$L(a.\varepsilon) = \{a\}$.

$\varepsilon.\varepsilon = \varepsilon$	$\phi.\phi = \phi$
$a.\varepsilon = a$	$a.\phi = \phi$
$\varepsilon.a = a$	

OR :

1. $\varepsilon + \varepsilon = \varepsilon$
2. $\phi + \phi = \phi$
2. $a + a = a$
4. $\varepsilon + a =$ } $= \varepsilon + a = a + \varepsilon$
5. $a + \varepsilon =$ }
6. $\varepsilon + \phi = \varepsilon$
7. $\phi + \varepsilon = \varepsilon$
8. $a + \phi = a$
9. $\phi + a = a$.

Regular Expression $\cong$ Set.

$\varepsilon \cong \{\varepsilon\}$

$\phi \cong \{\} = \phi$

$a \cong \{a\}$

$R_1 + R_2 \cong L(R_1) \cup L(R_2)$

$R_1 . R_2 \cong L(R_1) . L(R_2)$

eg:: $a + \varepsilon = a + \varepsilon$

either a or ε.

$|a| = 1$

$|\varepsilon| = 0$.

$L(a+\varepsilon) = \{a, \varepsilon\}$.

$a + \varepsilon \neq a$.

$a.\varepsilon = a$.

Concatenation.

1. $\varepsilon . \varepsilon = \varepsilon$.
2. $\phi . \phi = \phi$.
3. $a . a = aa = a^2$
4. $\varepsilon \phi = \phi$.

5. $\phi \cdot \varepsilon = \phi$
6. $a \cdot \varepsilon = a$
7. $\varepsilon \cdot a = a$
8. $\phi \cdot a = \phi$
9. $a \cdot \phi = \phi$.

$\phi \cdot R = \phi$
$R \cdot \phi = \phi$
$R \cdot \varepsilon = R$
$\varepsilon \cdot R = R$.

Language

- Finite Language → $\{a, ab, bbb\}$ (finite string)
- Infinite Language → $\{a, aa, aaa..\}$ (finite string)

String

- Finite String.
- Infinite String. (not exist in Computation).

$(a+ab)\ (\varepsilon + aaa + b)$

$\{a, ab\} \cdot \{\varepsilon, aaa, b\} = \{a, ab, abb, a^4, abaaa\}$

$a \cdot \varepsilon \rightarrow a$

$a \cdot aaa \rightarrow aaaa = a^4$

$a \cdot b \rightarrow ab$

$ab \cdot \varepsilon \rightarrow ab$

$ab \cdot aaa \rightarrow abaaa$

$ab \cdot b \rightarrow abb$

Kleene Star:

R^*

$R^0 + R^1 + R^2 \ldots$

$a^* = \{\varepsilon, a, aa, \ldots\}$

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

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

Language set.

1. $b^* = \{\varepsilon, b, bb, bbb \ldots\} = \{b^n \mid n \geq 0\}$

2. $(ab)^* = \{\varepsilon, ab, abab, ababab \ldots\} = \{(ab)^n \mid n \geq 0\}$
3. $(ba)^* = \{\varepsilon, ba, baba, bababa \ldots\} = \{(ba)^n \mid n \geq 0\}$
4. $(aa)^* = \{\varepsilon, a^2, a^4, a^6 \ldots\} \{a^{2n} \mid n \geq 0\}$.
5. $(a+b)^* = \{\varepsilon, a, b, aa, ab, ba, \ldots\} = \{a, b\}^*$
6. $(a+\varepsilon)^* = \{\varepsilon, a, a^2, a^3 \ldots\} = \{a^n \mid n \geq 0\} = a^*$.
7. $(b+\varepsilon)^* = b^*$.

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

$= \{aa, ab, ba, bb\}$

= Set of all 2 length strings

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

$\Sigma^* = \{a, b\} = (a+b)^*$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

$= \{\varepsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \ldots$

Set of all strings over Σ = Universal language

$a+b$

a ✓
b ✓
ab ✗
ba ✗
bb, aa ✗

$(a+b)^*$

$(a+b)^0 = \varepsilon$
$(a+b)^1 = a, b$
$(a+b)^2 = (a+b)(a+b)$
$= aa, ab, ba, bb.$

$(a+\varepsilon)^*$

$(a+\varepsilon)^0 = \varepsilon$
$(a+\varepsilon)^1 = a, \varepsilon.$
$(a+\varepsilon)^2 = (a+\varepsilon)(a+\varepsilon) \rightarrow a^2, a, \varepsilon$
$(a+\varepsilon)^3 = a^3, a^2, a, \varepsilon$

$\varepsilon, a, a^2, a^3 \ldots\ldots a^*$.

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

Kleene Plus.

R^+

$R^1 + R^2 + R^3 \ldots\ldots$

$a^+ = \{a, a^2, a^3 \ldots\} = \{a^n \mid n \geq 01\}$

$b^+ = \{b, b^2, b^3 \ldots\} = \{b^n \mid n \geq 01\}$

1. $(a+\varepsilon)^+ = a^*$
2. $(b+\varepsilon)^+ = b^*$
3. $(a+b+\varepsilon)^+ = (a+b)^*$
4. $a^+ + \varepsilon = a^+ + a^0 = a^*$.
5. $a^+ + \varepsilon = a^*$.

$R^+ = R^+ + \varepsilon^0$.

$(a+\varepsilon)^1 \rightarrow a, \varepsilon$.

$(a+\varepsilon)^2 = (a+\varepsilon)(a+\varepsilon) = a, \varepsilon, a^2$

$\vdots$

$\varepsilon, a, a^2, a^3 \ldots a^*$.

	OR	Concatenation.
1. Identity.	ϕ (Empty expression)	ε
2. Associative	✓ (holds)	✓ (holds)
3. Commutative	✓ (holds)	✗ (does not hold)
4. Dominator. (Annihilator)	Σ^*	ϕ.

Identity: $R + I = I + R = R$

$R + \phi = \phi + R = R$.

$I = \phi$ for $+$

ϕ is identity for "OR" operation.

$I = \varepsilon$ for $\cdot$

$R.I = I.R = R$

$R.\varepsilon = \varepsilon.R = R$.

ε is identity for concatenation.

Associative: $(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3)$

(left associative) (Right associative.)

$(a+b)+c = a+(b+c)$

$\hookrightarrow \{a, b, c\} \swarrow$

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

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

$\hookrightarrow \{a, b, c\}$ ↙

Commutative :

$R_1 + R_2 = R_2 + R_1$ ✓ holds

$R_1 . R_2 \neq R_2 . R_1$ (need not be) Does not holds.

Dominator :

$R + D = D + R = D$ ∵ Σ^* is dominator for +

$R + \Sigma^* = \Sigma^* + R = \Sigma^*$

$R \cdot D = D \cdot R = D$ ϕ is dominator for .

$R \cdot \phi = \phi . R . = \phi$

Distributive :

(I) OR over Concatenation

(i) Left distribution

$a + (b.c) \stackrel{?}{=} (a+b) \cdot (a+c)$

$\{a, bc\} \neq \{aa, ac, ba, bc\}$

(ii) Right distribution

$(a.b) + c \stackrel{?}{=} (a+c) \cdot (b+c)$

$\{ab, c\} \neq \{ab, ac, cb, cc\}$

} Does not hold. OR Should not be distributed over Concatenation.

(II) Concatenation over OR.

(i) Left distribution :

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

$\{ab, ac\} = \{ab, ac\}$.

(ii) Right distribution

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

$\{ac, bc\} \quad \{ac, bc\}$.

} It holds. Concatenation is distributed over OR.

Simplification:

$R^* = R' + R' \ldots$

$\phi^* = \phi^0 + \phi' + \phi^2 \ldots$

$= \varepsilon + \phi = \varepsilon$

* $(any)^0 = \varepsilon$.
* $any \cdot \phi = \phi$.
* $\varepsilon^* = \varepsilon$.

1. $\phi^* = \varepsilon$
2. $\varepsilon^* = \varepsilon$.
3. $\phi^+ = \phi$
4. $\varepsilon^+ = \varepsilon$.
5. $(a+\phi)^* = a^*$
6. $(a+\phi)^+ = a^+$
7. $(\varepsilon+\phi)^+ = \varepsilon^+ = \varepsilon$.
8. $(\varepsilon+\phi)^* = \varepsilon^* = \varepsilon$
9. $\phi^* \cdot \phi^+ = \varepsilon \cdot \phi = \phi$.
10. $\varepsilon^* \cdot \varepsilon^+ = \varepsilon \cdot \varepsilon = \varepsilon$.
11. $\phi^* \cdot \varepsilon^+ = \varepsilon \cdot \varepsilon = \varepsilon$
12. $\phi^+ \cdot \varepsilon^* = \phi \cdot \varepsilon = \phi$.
13. $a^* \cdot \phi^* = a^* \cdot \varepsilon = a^*$
14. $a^+ \cdot \phi^+ = \phi$.
15. $a^* \cdot a = a^+$
16. $a \cdot a^* = a^+$

$\uparrow\downarrow$ ($a^* = \varepsilon, a, aa \ldots \cdot a$

$= a', a^2, a^3 \ldots = a^+$

17. $a^* + a^* = a^*$
18. $a^* + a^+ = a^*$
19. $a^+ + a^+ = a^+$
20. $a + a^+ = a + (a + a^2 + \ldots) = a^+$.
21. $a + a^* = a^*$
22. $a^{100} + a^+ = a^+$
23. $a^* + a^+ + a + a^{1000} = a^* \cdot aa = a a^+$
24. $(a^+)^2 = a^* \cdot a^{*+} = a^* \cdot a \cdot a = a \cdot a^+$
25. $(a^*)^2 = a^* \cdot a^* = a^*$.
26. $(a^*)^{100} = a^*$.
27. $a^* a^* a^+ a^* = a^+$
28. $(a^*)^* = (a^*)^0 \cup (a^*)' \cup \ldots = a^*$
29. $(a^*)^+ = (a^*) \cup (a^*)^2 \ldots = a^*$
30. $(a^+)^* = (a^+)^0 \cup (a^+)' \ldots = \varepsilon + a^+ = a^*$
31. $(a^+)^+ = (a^+)' \cup (a^+)^2 \ldots = a^+$

Note:

(I) $R + \phi = R$
(II) $R \cdot \phi = \phi$
(III) $R \cdot \varepsilon = R$
(IV) $R^0 = \varepsilon$.
(V) $R + \Sigma^* = \Sigma^*$
(VI) $a \cdot a^* = a^+$
(VII) $a^* \cdot a = a^+$
(VIII) $(a+\varepsilon)^* = a^*$
(IX) $(a+\varepsilon)^+ = a^*$
(X) $a^+ + \varepsilon = a^*$

Homework : Find the min String in the following expressions :

1. $(ab)^+ aaa (b+ab) \rightarrow$ min = abaaab
2. $(ab)^+ aaa . (b+\varepsilon) \rightarrow aaa$
3. $(a+ab)^+ (bb+aaa)^+ (ab)^+ \rightarrow bb$
4. $(ab)^+ + \varepsilon \rightarrow ab$
5. $(ab)(a+b)^* aaa + ba \rightarrow ba$
6. $[(ab^*+a^++ba)ab]^+(a+b)^+ \rightarrow aab$
7. $[((ab)^+a)^* aaa]^+ \rightarrow aaa$
8. $[((aba)^+aba)^+ ab]^+ \rightarrow \varepsilon$
9. $(a^*)^+ \rightarrow a$
10. $((ab)^* a^+)^+ bb^+aa^+ \rightarrow abba$
11. $[(ab)^+ aba]^+ aaa \rightarrow ababaaaa$
12. $(aaa^*aba^+)^+ \rightarrow aaab$
13. $(a+aa+aaa)^+ \rightarrow a$
14. $a^+b^+ \rightarrow ab$
15. $a^*b^* \rightarrow \varepsilon$
16. $a^*b^*c^+d^* \rightarrow c$
17. $(a+b^*+c^+) \rightarrow a$
18. $(a+b+\varepsilon)^+ \rightarrow \varepsilon$
19. $[(a+ab)aa+ab]^+aaa \rightarrow abaaaa$
20. $(ab.aaa.a.ta^+)^+ \rightarrow \varepsilon$

Priority : 1. Kleene plus, Kleene star
2. Concatenation
3. OR.

Q1. How to write Regular expressions ?
→ Regular Language ≅ Regular Expression.

1. $L = \{\}$ over $\Sigma = \{a\}$
$R = \phi$.

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

Note: To represent one regular language, there are infinite regular expressions.

3. $L = \{\varepsilon\}$ over $\Sigma = \{a\}$

$R = \varepsilon = \phi = \phi^* \cdot a^0 \ldots$

4. $L = \{\varepsilon\}$ over $\Sigma = \{a, b\}$

$R = \varepsilon = \phi^0 = a^0 = b^0 \ldots$ (and many more).

5. L = Set of all strings over $\Sigma = \{a\}$

$R = a^* = \Sigma^*$

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

$R = (a+b)^* = \Sigma^*$.

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

$= a^+ + \varepsilon$

$= (a+\varepsilon)^+$

$= (a+\varepsilon)^*$

$= \phi^* \cdot a^*$

$=$ a^* + any over Σ exp.

$a^*b^* \neq b^*a^*$

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

7. $L = \{w \mid w \in \{a, b\}^*,$ w starts with $a\}$

$= \{aw \mid w \in \{a, b\}^*\}$

$R = a(a+b)^* = a\Sigma^*$.

eg:: $A = \{x \mid x \in N, x \geq 2\}$

Cond. 1 Condition 2.

Every natural number greater than 2 $= \{3, 4, 5 \ldots\}$

8. Set of all strings starting with ab over $\Sigma = \{a, b\}$

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

9. $L = \{w \mid w \in \{a, b\}^*,$ w starts with $aaa\}$

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

10. $L = \{w \mid w \in \{a, b\}^*,$ w ends with $a\}$

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

11.

Group I.	Group II.
1. $a(a+b)^*$	I. $(ab^*)^+ = \{a, aa, ab \ldots\}$
2. $b(a+b)^*$	II. $(b^*a)^+ = (a+b)^*a$
3. $(a+b)^*a$	III. $(ba^*)^+ = b(a+b)^*$
4. $(a+b)^*b$	IV. $(a^*b)^+ = (a+b)^*b$.

12. $L = \{w_1 a w_2 \mid w_1, w_2 \in \{a,b\}^*\}$

= Set of all Strings Containing 'a' as Substring

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

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

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

13. $L = \{w \mid w \in \{a,b\}^*, \underbrace{|w| = 2}\}$ exactly 2 length Strings.

$R = ab + ba + aa + bb$

$= (a+b)(a+b)$

$= (a+b)^2 = \Sigma^2$

14. $\{w \mid w \in \{a,b\}^*, \underbrace{|w| \leq 2}\}$ Almost 2 length Strings.

$R = \varepsilon + a + b + aa + ab + ba + ba$

$= \Sigma^0 + \Sigma^1 + \Sigma^2 = (\varepsilon + a + b)^2$.

$|w| = k$

$= (a+b+\varepsilon)^k$.

String	Length.
ε	0
$a+b$	1
$ab+\varepsilon$	0 or 1 ≤ 1
$(a+b+\varepsilon)^2$	$\leq$ 2 Length

15 $\{w \mid w \in \{a,b\}^*, \underbrace{|w| \geq 2}_{\text{Atleast 2 length}}\}$

$R = \underbrace{(a+b)^2}_{\text{exactly 2 length}} \underbrace{(a+b)^*}_{\text{any length}}$

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

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

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

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

16. $\{w \mid w \in \{a,b\}^*, n_a(w) = 2\}$

$R = b^* a b^* a b^*$

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

$= b^* (ab^*)^2$.

$n_a(w)$ → no. of a's in (w)

17. $\{w \mid w \in \{a,b\}^*, n_a(w) \leq 2\}$

$R = b^* + b^* a b^* + b^* a b^* a b^*$.

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

Zero a's $= b^*$

One a $= b^* a b^*$

2 a's $= b^* a b^* a b^*$.

18. $\{w \mid w \in \{a,b\}^*, n_a(w) \geq 2\}$

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

19. $\{w \mid w \in \{a,b\}^*, |w| \text{ is even}\}$.

$R = \left[(a+b)^2\right]^*$

20. $\{w \mid w \in \{a,b\}^*, |w| \text{ is odd}\}$

$R = \underbrace{\left[(a+b)^2\right]^*}_{\text{even}} \underbrace{(a+b)}_{\text{odd}}$! $= (a+b)\left[(a+b)^2\right]^*$.

odd

21. $L = \{xay \mid x,y \in \{a,b\}^*, |x| = 3\}$

$=$ 4th symbol from beginning is a.

$= \underbrace{(a+b)^3}_{\text{3 length}} \underbrace{a}_{\text{4th symbol}} (a+b)^*$.

22. $\{xay \mid x, y \in \{a,b\}^*, |y| = 2\}$
any (2 length.
$= (a+b)^* a (a+b)^2$. } 3rd symbol from end is a.

23. $\{a^m b^n \mid m, n \geq 0\}$
$R = a^* b^*$.

24. $\{a^m b^n \mid m \geq 0, n \geq 1\}$
$R = a^* b^+$

25. $\{a^m b^n \mid m \geq 1, n \geq 1\}$
$R = a^+ b^+$.

26. $\{a^m b^n \mid m = \text{even}, n \geq 0\} = R = (aa)^* b^*$

27. $\{a^m b^n \mid m = \text{odd}, n \geq 1\} = R = a(aa)^* b^+$

28 $\{a^m b^n \mid m \geq 2, n = \text{even}\} = R = aaa^*(bb)^*$

29 $\{a^m b^n \mid m \geq 2, n \geq 3\} = R = aaa^* bbbb^*$

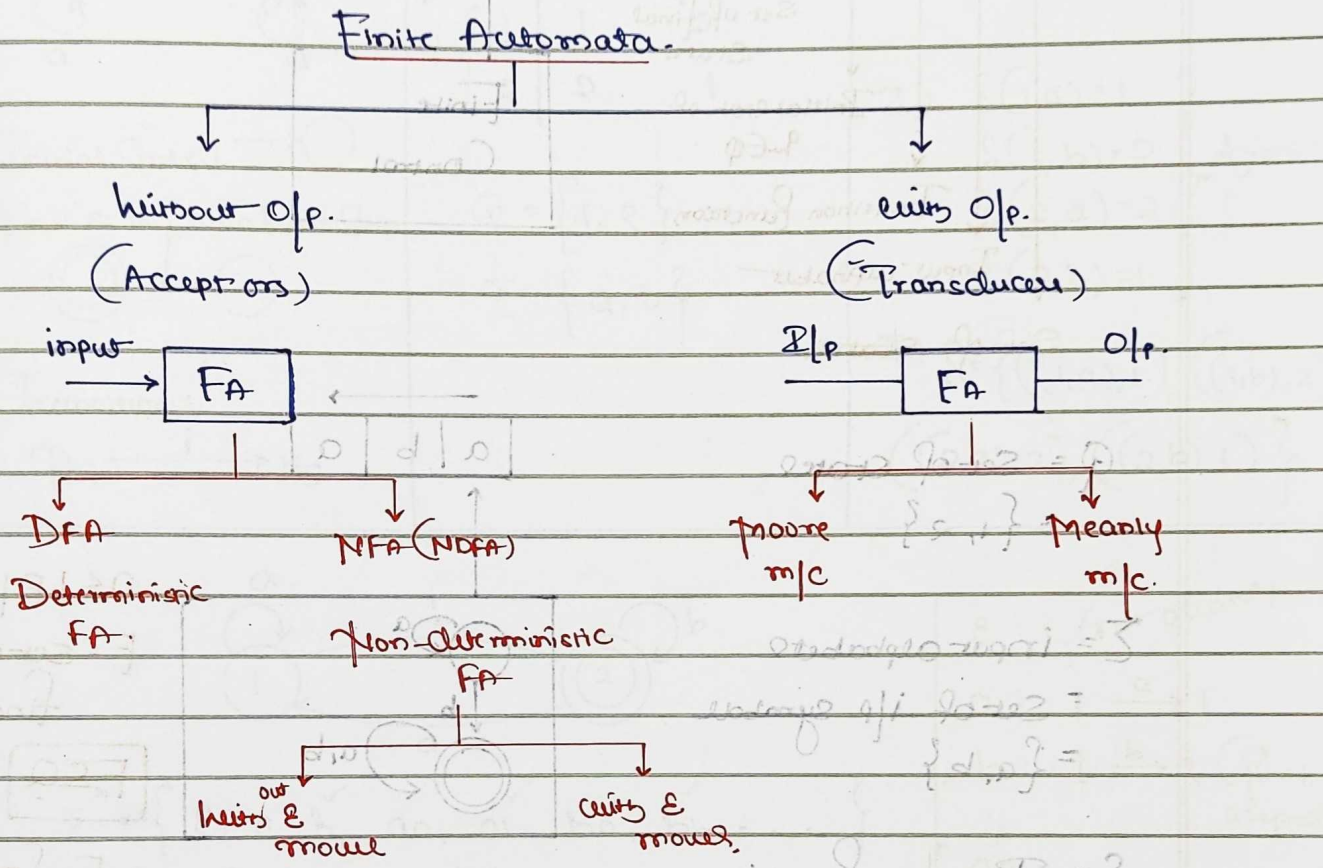
30 $\{a^m b^n \mid m \leq 1, n \geq 2\} = R = \underbrace{(\varepsilon + a)}_{\text{max 1a.}} + \underbrace{bbb}_{\text{min 2 bs.}} b^*$.

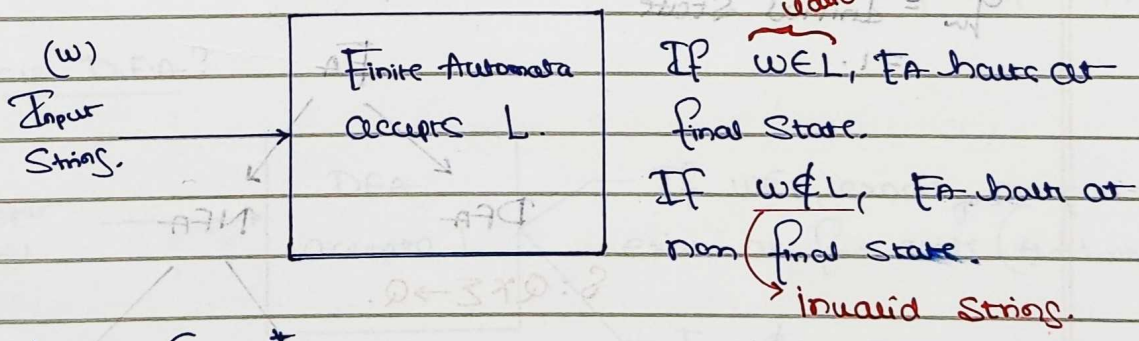
***
31. $\{w \mid w \in \{a,b\}^*, na(w) = \text{odd}\}$
$= b^*(b^* a b^* a b^*)^* b^* a b^*$
$= b^* a b^* (b^* a b^* a b^*)^* b^* a b^*$.
$= b^* a b^* (b^* a b^* a b^*)^* b^*$.

two
find our other equivalent regular exp.

32. $\{w \mid w \in \{a,b\}^*, na(w) = \text{even}\}$
$= b^*(b^* a b^* a b^*)^* b^*$
$= b^*(a b^* a b^*)^* b^*$. $= (b^* a b^* a)^* b^*$.

Finite Automata. : It represents a regular language
(Finite State Machine) (accepts)
(Finite Machine) (recognises).





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

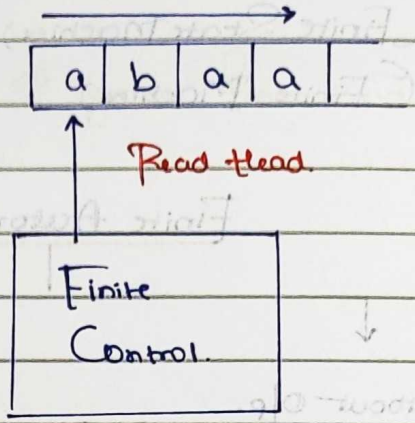
all strings start with ab.

$\Sigma^* = \{ \varepsilon, a, b, aa, ab, ba, bb, aaa, aab, aba \ldots \}$

FA Configuration.

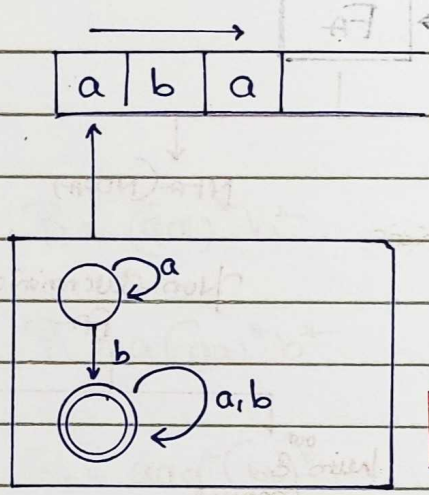
$FA = (Q, \Sigma, \delta, q_0, F)$

- F → Set of final State
- $q_0 \in Q$ → Initial state of
- δ → Transition function
- Σ → Input alphabet
- Q → Set of states.



Q = Set of states
$= \{1, 2\}$

Σ = input alphabets
= Set of i/p symbol
$= \{a, b\}$



$0 \le |F| \le |Q|$

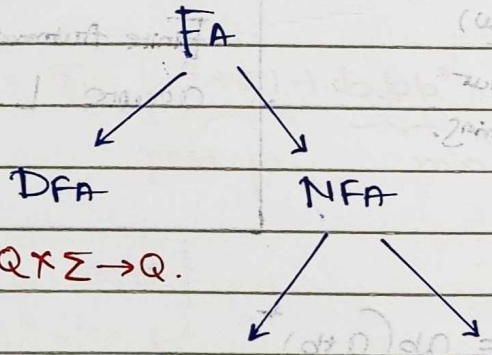
F = set of final states

$F \subseteq Q$

$F = \{2\}$

δ = Transition function

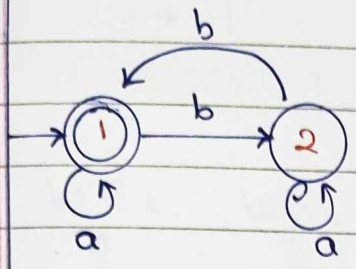
q_0 = Initial State
$= 1.$

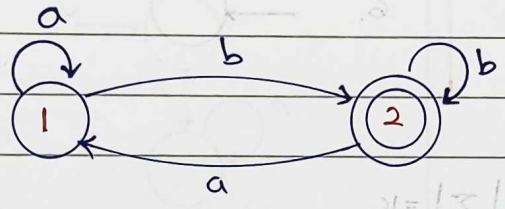


FA Representations.

1. State Digram.
2. Transition Table.
3. Set / Function / Relation.

classmate
Date 02/5/23
Page

Diagram	Table	Set.
	Σ δ | a | b Q { 1 | 1 | 2 2 | 2 | 1	FA = ({1,2}, {a,b}, δ, 1, {1})
Initial State: → ○	Q = {1, 2}	I). δ(1,a) = 1 δ(1,b) = 2 δ(2,a) = 2 δ(2,b) = 1 } transition
Final state / Non Final State 	Σ = {a, b}	II) δ = {((1,a),1), ((1,b),2), ((2,a),2), ((2,b),1)}
Transition: Ps. —i→ Ns		



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

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

ε : 1. (not accepted)

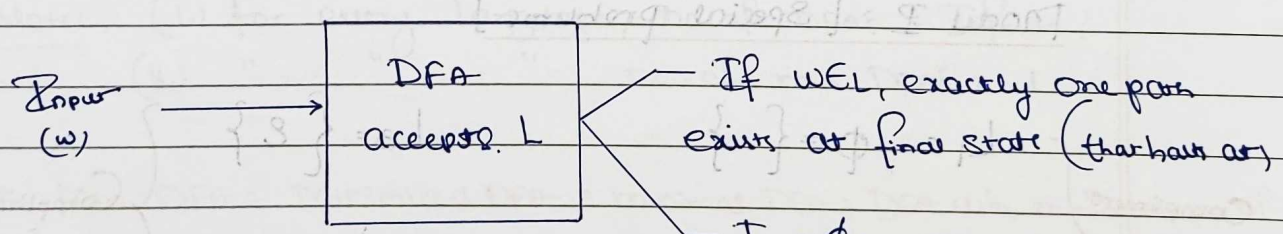
a : 1 —a→ 1

b : 1 —b→ 2 (final, accepted)

aa :

1 —a→ 1 —a→ 1.

→ What is D.F.A?



DFA Definations

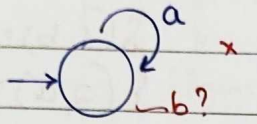
I $\delta : Q \times \Sigma \to Q$

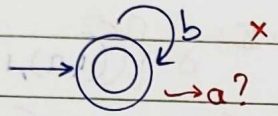
II From every state, for every i/p symbol, exactly one transition to next state.

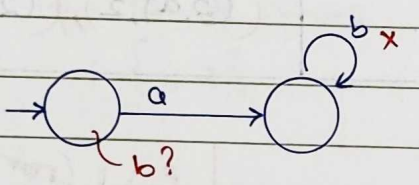
III. $\forall w \in \Sigma^* \Rightarrow$ exactly one path exists.

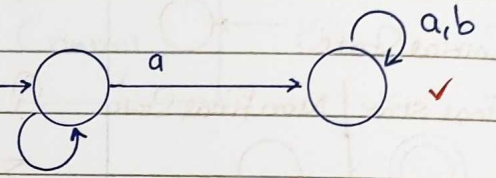
[valid string halts at final
invalid string halts on non-f.]

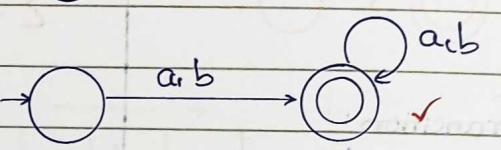
Identify Correct DFA's $\Sigma = \{a, b\}$.

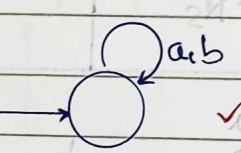
1. 

2. 

3. 

4. 

5. 

6. 

Note: $|Q| = n$ $\quad |\Sigma| = k$

how many transitions in DFA?

$= nk$

$= |Q \times \Sigma| = |Q| \times |\Sigma|$

Module I. [Special problems]

Compliment to each other.

$L_1 = \phi = \{\}$

$L_2 = \Sigma^*$

Universal Set.

$L_3 = \{\varepsilon\}$

$L_4 = \Sigma^+$

Compliment to each other.

→ $\bar{L} = \Sigma^* - L$

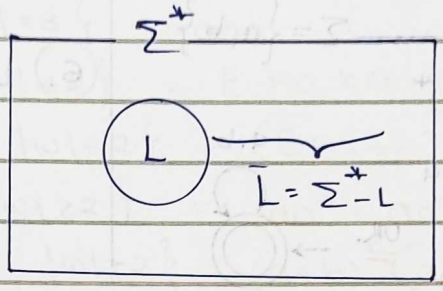
= Universal Set − L

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

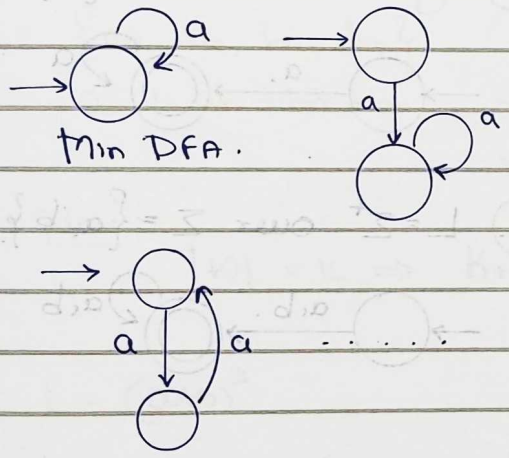
$L \cap \bar{L} = \phi$

$w \in L$ ⟺ $w \notin \bar{L}$

$w \notin L$ ⟺ $w \in \bar{L}$

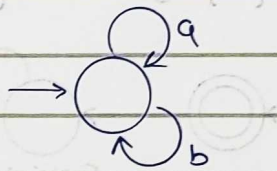


① $L = \{\ \}$ over $\Sigma = \{a\}$



..... .

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



$\varepsilon = A$

$a = A \xrightarrow{a} A$

$b = A \xrightarrow{b} B.$

Note: (I.) For every language, no. of DFAs = Infinite

(II.) " " " no. of min. DFAs = 1.

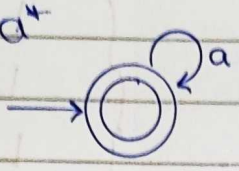
→ Min. DFA = Minimized DFA = minimal DFA = DFA with minimum no of states

(III) If DFA has only non final states then L(DFA) is $\phi = \{\ \}$

(IV) If DFA has only final states then L(DFA) is Σ^*

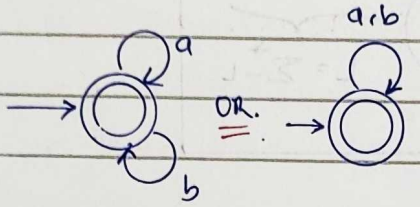
(3.) $L = \Sigma^*$ over $\Sigma = \{a\}$

$= a^*$



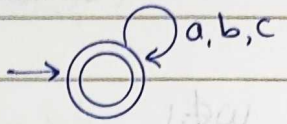
(4.) $L = \Sigma^*$ over $\Sigma = \{a, b\}$

$= (a+b)^*$



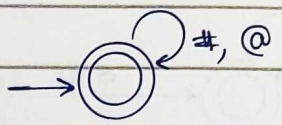
(5.) $L = \Sigma^*$ over $\Sigma = \{a, b, c\}$.

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

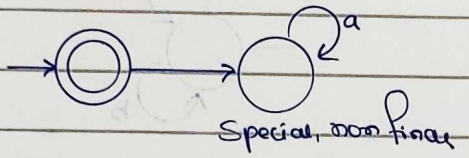


(6.) $L = \Sigma^*$ over $\Sigma = \{\#, @\}$

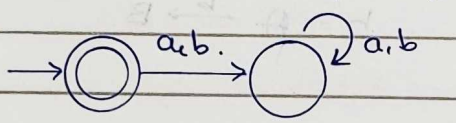
$= (\# + @)^*$.



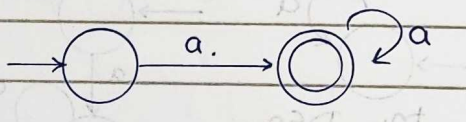
(7.) $L = \{\varepsilon\}$ over $\Sigma = \{a\}$.



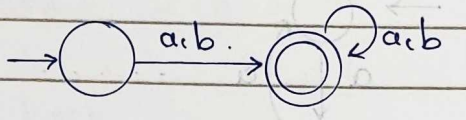
(8.) $L = \{\varepsilon\}$ over $\Sigma = \{a, b\}$.



(9.) $L = \Sigma^+$ over $\Sigma = \{a\}$

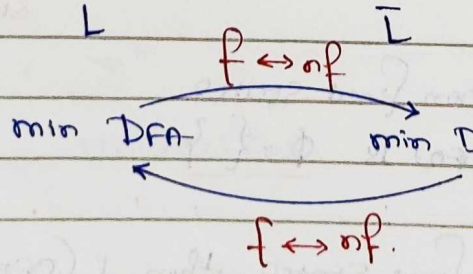


(10.) $L = \Sigma^+$ over $\Sigma = \{a, b\}$.



Note:

If L has min DFA with n states, then $\bar{L}$ has min ~~states~~ DFA with n states.



Model - II [Length based problems]

1. $L_1 = \{w \mid w \in \{a,b\}^*, |w| = 2\} = (a+b)^2$
2. $L_2 = \{w \mid w \in \{a,b\}^*, |w| = 3\} = (a+b)^3$
3. $L_3 = \{w \mid w \in \{a^*\}, |w| = 2\} = aa$
4. $L_4 = \{w \mid w \in a^*, |w| = 3\} = aaa$
5. $L_5 = \{w \mid w \in a^*, |w| \leq 2\} = \varepsilon + a + a = (\varepsilon + a)^2$
6. $L_6 = \{w \mid w \in (a+b)^*, |w| \leq 2\} = (\varepsilon + a + b)^2$
7. $L_7 = \{w \mid w \in a^*, |w| \geq 2\} = aaa^* = aa^+ = a^+a = a^*aa$
8. $L_8 = \{w \mid w \in (a+b)^+, |w| \geq 2\} = (a+b)^2(a+b)^*$.

→ $L_1 = \{w \mid w \in \{a,b\}^*, |w| = 2\} = (a+b)^2$.

= Set of all 2 length strings.

= $\{aa, ab, ba, bb\}$

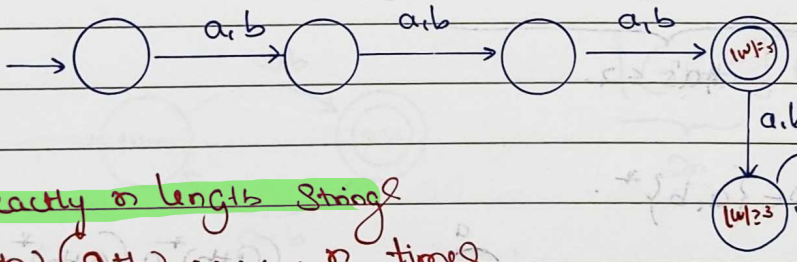
= $aa + ab + ba + bb$ $= (a+b)(a+b)$

$= (a+b)^2$.

Note:

$|w| = k \Rightarrow k+2$ States in DFA

→ $L = (a+b)^3$.



Exactly n length strings

$(a+b)(a+b)\ldots\ldots$ n times

↓

n+2 States in DFA.

Exactly n length
Atmost n length
} Dead State is required $\Rightarrow (n+1) + 1$ → dead state.

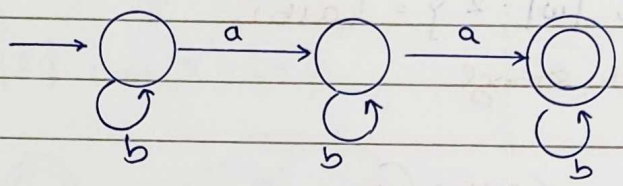
Atleast n length } dead state not required $\Rightarrow n+1$

tlw

Modu - III [No. of Symbols]

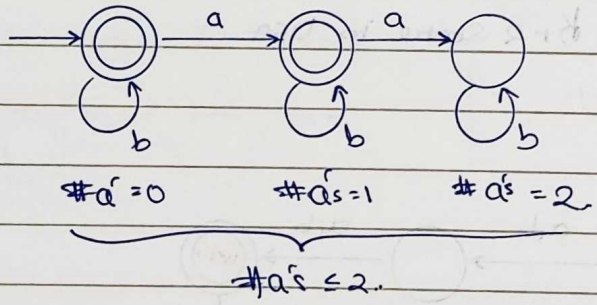
1. $\{w \mid w \in \{a,b\}^*, \ n_a(w) = 2\}$
2. $\{w \mid w \in \{a,b\}^* \ n_a(w) \leq 2\}$
3. $\{w \mid w \in \{a,b\}^* \ n_a(w) \geq 2\}$.
4. $\{w \mid w \in a^*, \ n_a(w) = 2\}$
5. $\{w \mid w \in a^*, \ n_a(w) \leq 2\}$
6. $\{w \mid w \in a^*, \ n_a(w) \geq 2\}$.

1. No. of a's = 2. $w \in \{a,b\}^*$.



#a's = 2
#b's ≥ 0
$b^*ab^*ab^*$.

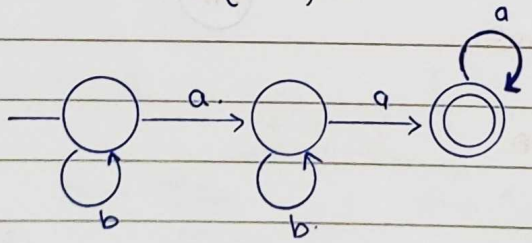
2. No. of a's ≤ 2 $w \in \{a,b\}^*$



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

#a's ≤ 2
#b's ≥ 0.

3. #a's ≥ 2, $w \in \{a,b\}^*$.

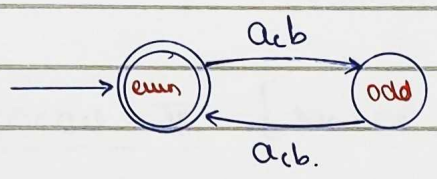


$(a+b)^*a(a+b)^*a(a+b)^*$

Module - IV Length and Remainder

1. $\{w \mid w \in \{a,b\}^*, |w| \text{ is divisible by } 2\}$.
2. $\{w \mid \text{"} \text{"} |w| \text{ is not divisible by } 2\}$ } Compliment to each other.
3. $\{w \text{"} \text{"} \text{"} |w| \text{ is divisible by } 3\}$.
4. $\{w \text{"} \text{"} \text{"} |w| = 3n+2\}$ $n \geq 0$.
5. $\{w \text{"} \text{"} \text{"} |w| = 3n+5\}$ $n \geq 0$.

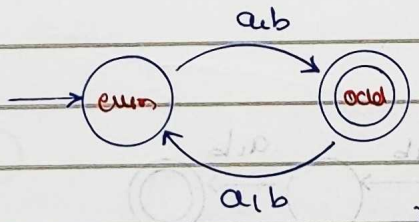
→ $\{w \mid w \in \{a,b\}^*, |w| = \text{even}\}$.



$[(a+b)^2]$

rem = 0	rem = 1.
$\|w\| = 2$	$\|w\| = 1$
" 4	" 3
" 6	" 5
	⋮

→ $\{w \mid w \in \{a,b\}^*, |w| = \text{odd}\}$.



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

$2n$ 1

$2n+1$

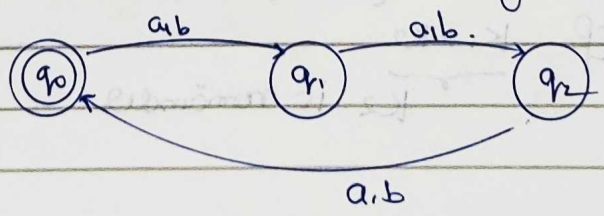
$|w| = 2n+1$

$|w| = 1 \bmod 2$

$(|w| \% 2) = 1$.

rem. 1
$\|w\| = 1$
3
5
⋮

→ $\{w \mid w \in \{a,b\}^* |w| \text{ is divisible by } 3\}$.

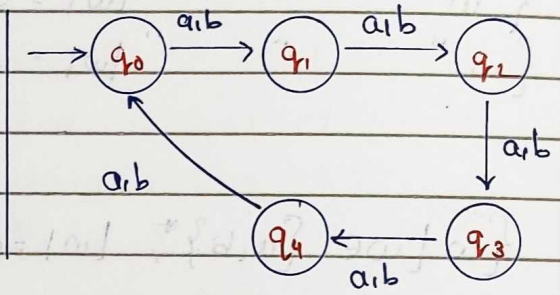


Note: $L = \{w \mid w \in \{a,b\}^*, |w| \text{ is divisible by } k\}$.
No of states in min DFA = k states.

$|w| = 5n \Rightarrow$ rem 0 when divide with 5.

5 States. Final States may differ.

$|w| = 5n+1$ = rem 1.
$|w| = 5n+2$ = rem 2.
$|w| = 5n+3$ = rem 3.
$|w| = 5n+4$ = rem 4



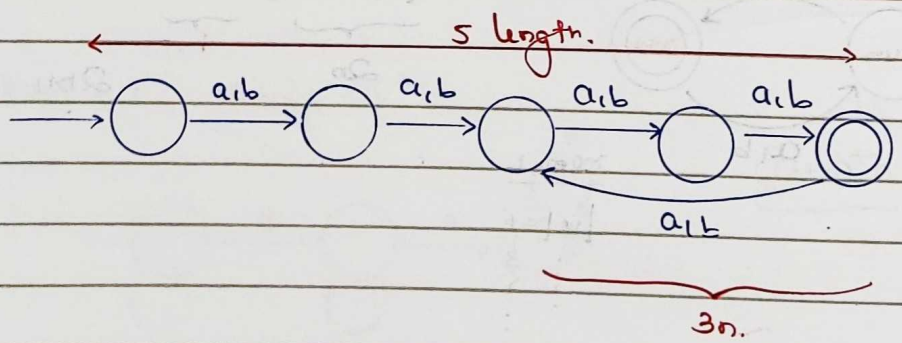
$|w| = 3n+0$
$= 3n+1$
$= 3n+2$
$= 3n+3$
$= 3n+4$
$= 3n+5$
$= 3n +$ constant $(k) \longrightarrow k+1$ States.

3 States.

If $k_1 > k_2$, k_2 is the remainder.

If $k_1 \leq k_2$.

$\rightarrow \{|w| = 3n+5\}$ $3n+2$. $n \geq 0$.
$= 5, 8, 11 \ldots$



Note: $L = \{w \mid w \in \{a,b\}^*, |w| = k_1 n + k_2, n \geq 0\}$.
I.) If $k_1 > k_2$ $\Rightarrow k_1$ states
k_2 is remainder.

II) If $k_1 \leq k_2$ $\Rightarrow$ k_2+1 States.
(k_2 is not remainder.)

Note: $L = \{w \mid w \in \{a,b\}^*, \#a(w) = k_1 n + k_2 \geq 0\}$.

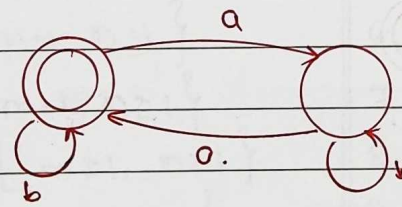
I) If $k_1 > k_2$ (k_2 is remainder) $\Rightarrow k_1$ States.

II) If $k_1 \leq k_2$ (k_2 is not remainder) $\Rightarrow k_2+1$ States.

Model - V [No. of Symbols and Remainder]

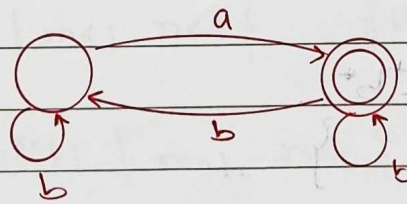
1. $\{w \mid w \in \{a,b\}^*, n_a(w) = \text{even}\}$ } 2 States
2. $\{w \mid$ " " $n_a(w) = \text{odd}\}$. }
3. " " $n_a(w)$ divisible by 3 }.
4. $\{w$ " " $n_a(w)$ not divisible by 3 }.
5. $\{w$ " " $n_a(w) = 100n + 53$, $n \geq 0$ } → 100 States
6. $\{w$ " " " $= 1230n + 1125$, $n \geq 0$. } → 1126 States

(1.)



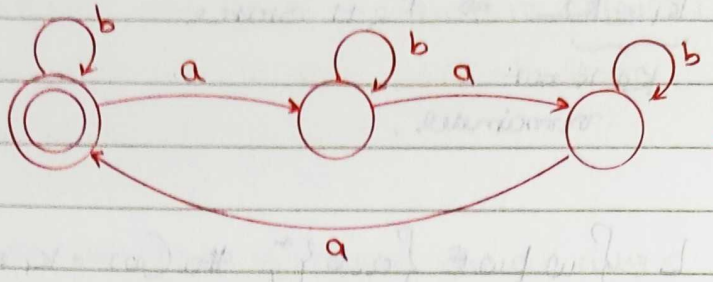
#a's even

(2)



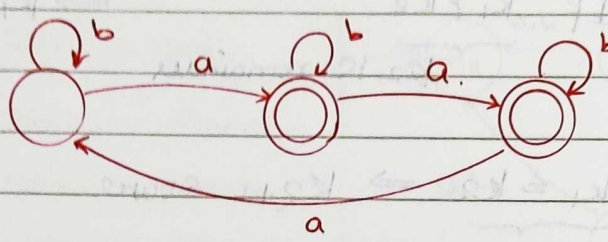
#a's = Odd.

③



#a's is divisible by 3.

④



[#a(w) % 3] != 0
= 1
= 2.

Mode VI. [Sequence]

1. a^+b^+
2. a^+b^*
3. a^*b^+
4. a^*b^*.

} a's followed by b's.

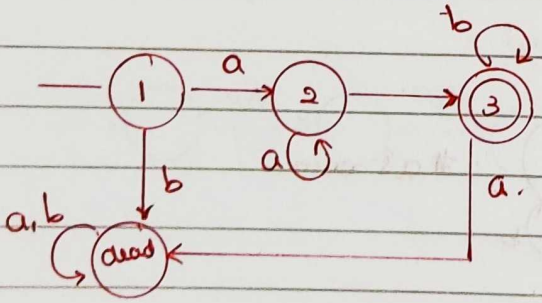
→ $L = \{a^m b^n \mid m \geq 1, n \geq 1\} = a^+b^+$

$= aa^*bb^*$

$= \{ab, aab, abb \ldots\} = a^*abb^*$.

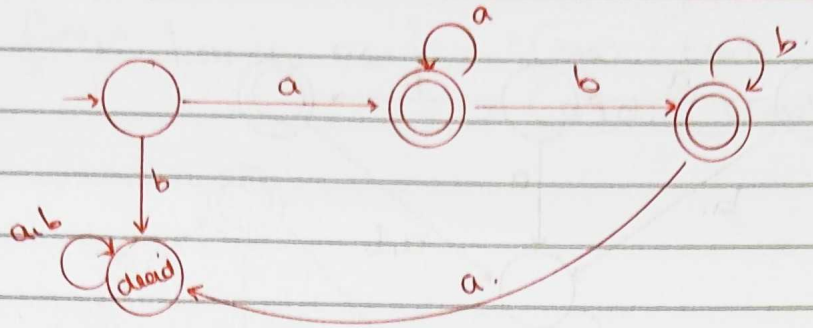
$= a^*a.b^*b \quad = (a^+)^*(b^+)^+$.

} a's will never appear after b's. & b's will never before a's.



→ $\{a^m b^n \mid m \geq 1, n \geq 0\} = a^+b^*$

$= \{a, aa, ab, aab \ldots\}$

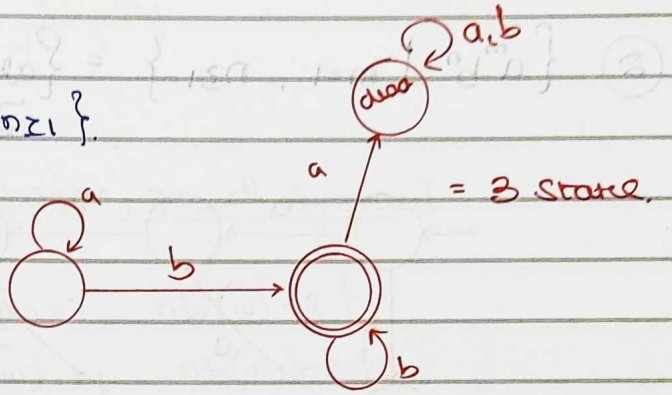


→ $L = a^*b^*$

$= \{a^m b^n \mid m \geq 0, n \geq 1\}$.

$= \{b, \ldots\}$

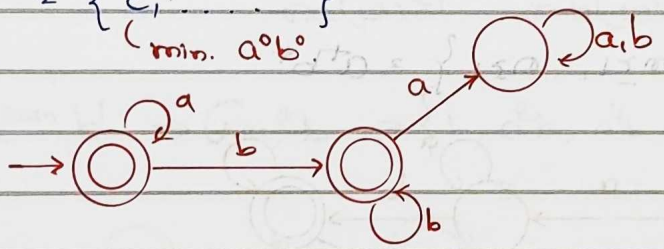
(min a^0b^1)



→ $a^*b^* = \{a^m b^n \mid m \geq 0, n \geq 0\}$

$= \{\varepsilon, \ldots\}$

(min. a^0b^0.



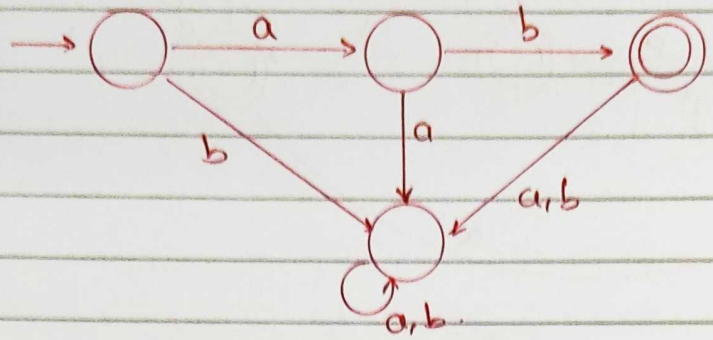
Howwork

5. $L = b^*a^*$
6. $L = b^+a$
7. $L = b^*a^+$
8. $L = b^*a^*$
9. $L = a^+b^+c$
10. $L = b^+a^*c^+$.

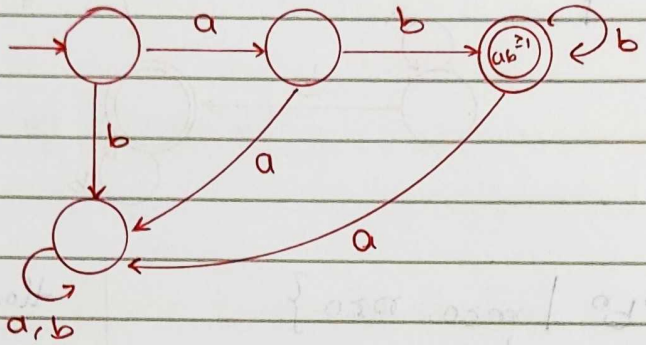
Modu-7: [sequence, no. of symbols]

1. $\{a^m b^n \mid m=1, n=1\}$.
2. $\{a^m b^n \mid m=1, n \geq 1\}$
3. $\{a^m b^n \mid m \geq 1, n \geq 1\}$
4. $\{a^m b^n \mid m \leq 1, n=1\}$
5. $\{a^m b^n \mid m \leq 1, n \leq 1\}$.
6. $\{a^m b^n \mid m = \text{even}, n=1\}$
7. $\{a^m b^n \mid m = \text{odd}, n=1\}$
8. " $m=1, n \neq \text{even}\}$
9. " $m = \text{even}\ n = \text{even}\}$
10. " $m = \text{even}, n = \text{odd}\}$.

① $\{a^m b^n \mid m=1, n=1\} = \{a,b\}$.

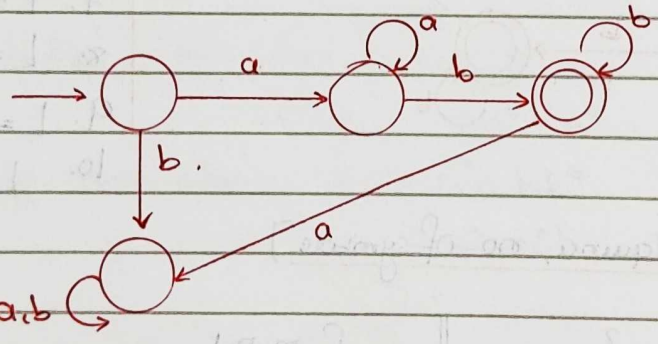


② $\{a^m b^n \mid m=1, n\geq 1\} = \{\underbrace{ab}_{\text{min}}, \ldots\}$.

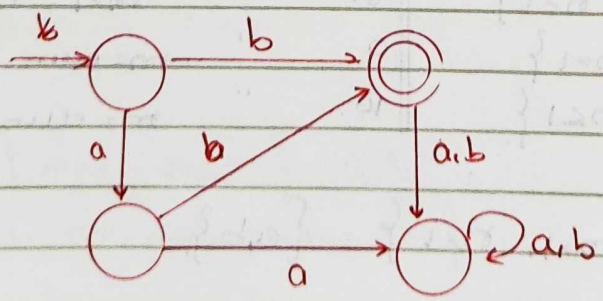


$L = ab^+$.

③ $L = \{a^m b^n \mid m\geq 1, n\geq 1\} = a^+b^+$

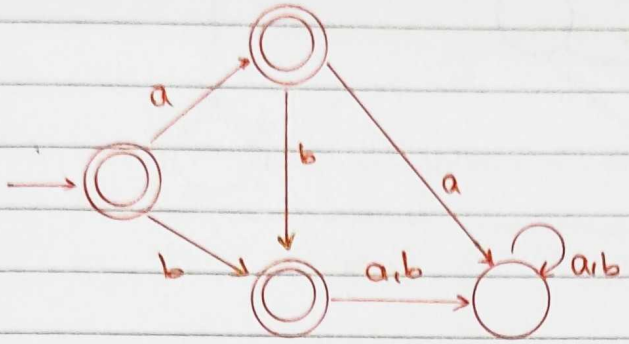


④. $\{a^m b^n \mid m\geq 1, n=1\} = (\varepsilon + a)b = b + ab$.



= 4 States.

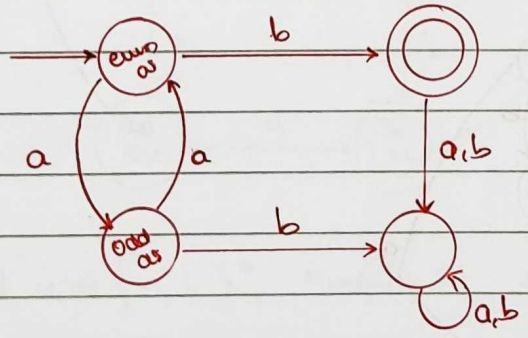
(5) $\{a^m b^n \mid m \le 1, n \le 1\} = (\epsilon + a)(\epsilon + b)$
$= (a^0 + a)(b^0 + b^1) = \{\epsilon, a, b, ab\}$.



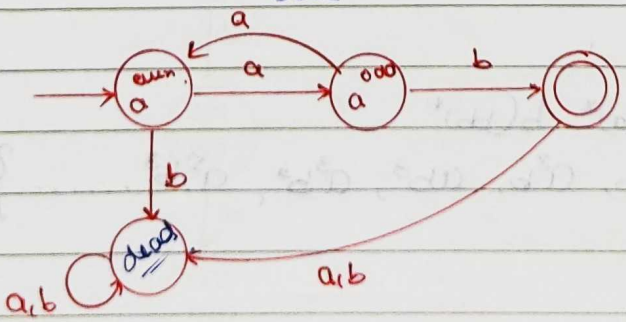
Model - VIII [Multiple conditions, No. of Symbols]

1. $\{w \mid w \in \{a,b\}^*, n_a(w) = 1, n_b(w) = 2\}$
2. $\{w \mid$ " " $n_a(w) \ge 1, n_b(w) \ge 2\}$
3. " " $n_a(w) \le 1, n_b(w) \le 2\}$
4. " " $n_a(w) \le 1, n_b(w) \le 2\}$.
5. " " $n_a(w) \ge 1, n_b(w) \le 2\}$.

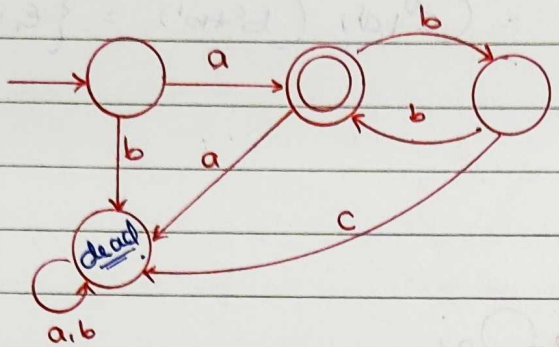
(6) $a^{even} b^1 \Rightarrow (aa)^* b = \{b, a^2b, a^4b, a^6b \ldots\}$



(7) $a^{odd} b^1 \Rightarrow \underbrace{a(aa)^*}_{odd} b = \{ab, a^3b, a^5b \ldots\}$.

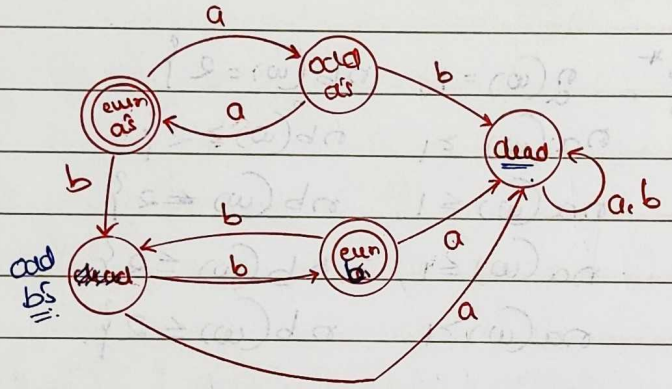


(8) $a^1 b^{even} = a(bb)^* \Rightarrow \{a, ab^2, ab^4 \ldots\}$

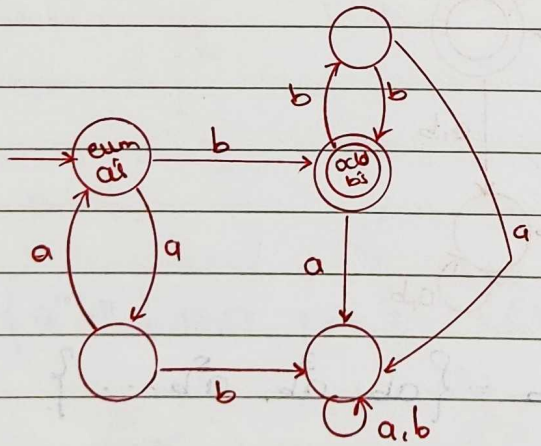


(9) $a^{even} b^{even} = (aa)^* (bb)^*$.

$= \{\varepsilon, a^2, b^2, a^4, b^4, \ldots\}$.

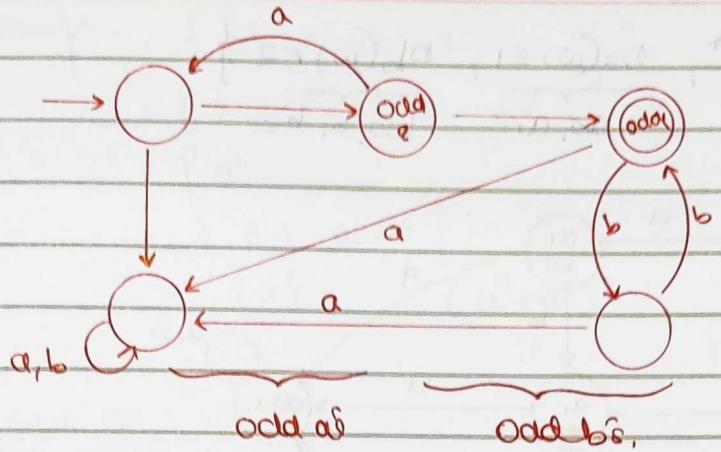


(10) $a^{even} b^{odd} = (aa)^* b (bb)^*$.



(11) $a^{odd} b^{odd} = a(aa)^* b(bb)^*$

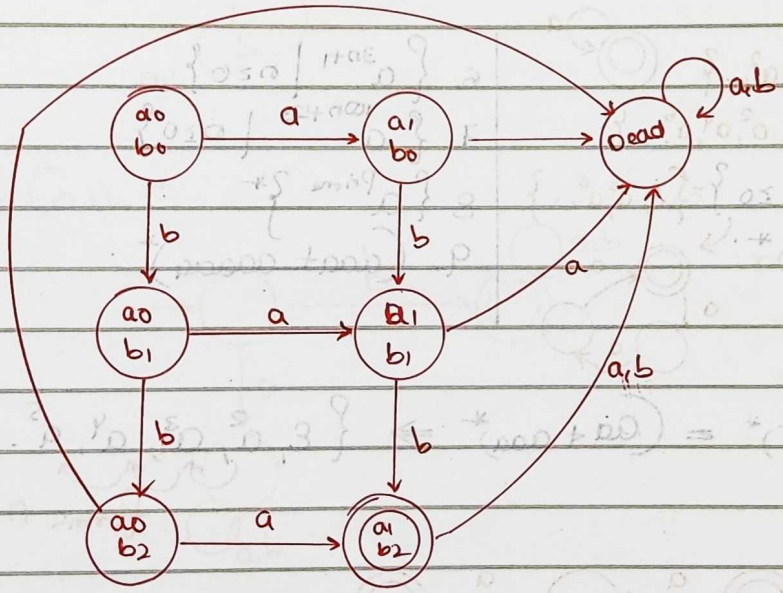
$= \{ab, a^3b, ab^3, a^3b^3, a^5b^3 \ldots\}$



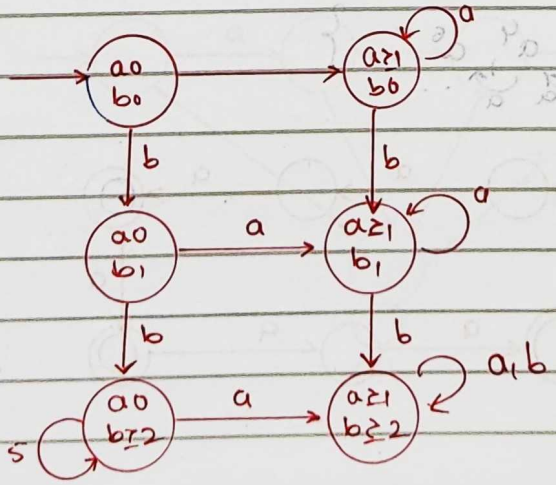
Model VIII (Solutions)

1. $\{ w \mid w \in \{a,b\}^*,\ n_a(w) = 1,\ n_b(w) = 2 \}$

$= \{ abb, bab, bba \}$.



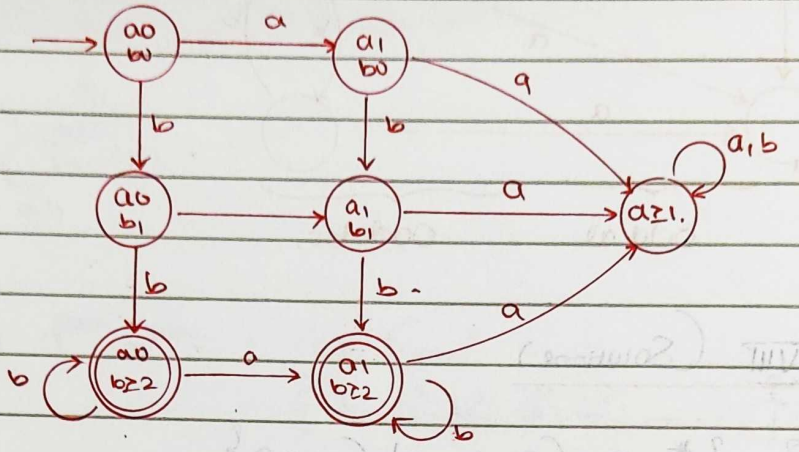
2. $\{ w \mid w \in \{a,b\}^*,\ n_a(w) \geq 1,\ n_b(w) \geq 2 \}$



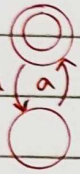
$a_1 \Rightarrow a_{\geq 1}$

$b_2 \Rightarrow b_{\geq 2}$.

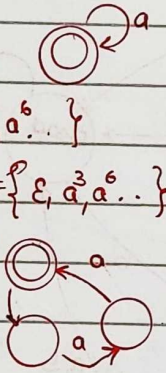
3. $\{ w \mid w \in \{a,b\}^*,\ \underbrace{n_a(w) \le 1}_{a_0, a_1},\ \underbrace{n_b(w) \ge 2}_{b_0, b_1, b_{\ge 2}} \}$



Model IV [Language over 1 Symbol]



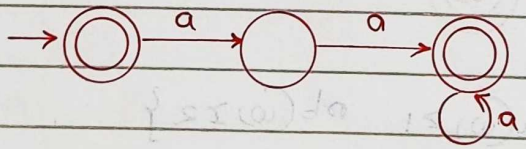
1. $a^* = \{\varepsilon, a^1, a^2, \ldots\}$
2. $(aa)^* = \{\varepsilon, a^2, a^4, a^6 \ldots\}$
3. $\{ \exists n\ a^{3n} \mid n \ge 0 \} = \{\varepsilon, a^3, a^6 \ldots\}$
4. $(aa + aaa)^*$
5. $(a^3 + a^4)$



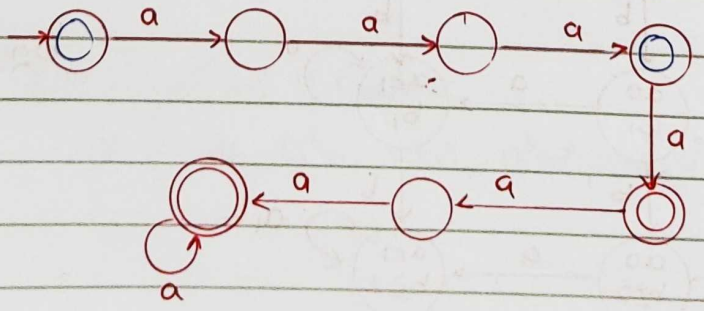
6. $\{ a^{3n+1} \mid n \ge 0 \}$
7. $\{ a^{100n+23} \mid n \ge 0 \}$
8. $\{ a^{\text{Prime}} \}^*$
9. $(aaaa + aaaaa)^*$

(4.) $(a^2 + a^3)^* = (aa + aaa)^* \Rightarrow \{\varepsilon, a^2, a^3, a^4, a^5 \ldots\}$

Forms AP.



(5.) $(a^3 + a^4)^* = \{\varepsilon, a^3, a^4, a^6 \ldots\}$



7. $\{a^{Prime}\}^* = \{a^2, a^3, a^5, a^7 \ldots\}^*$

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

$= \{a^*\}$

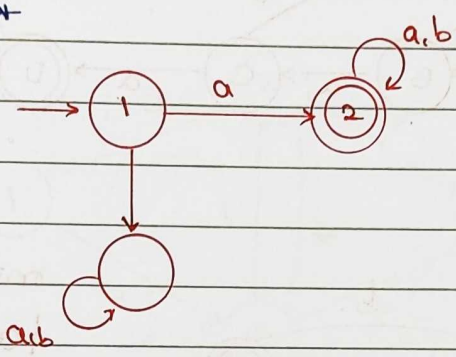
$= \{\varepsilon + aaa^*\}$

$= \{a\} = (a^2+a^3)^*$.

4.

Module II

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



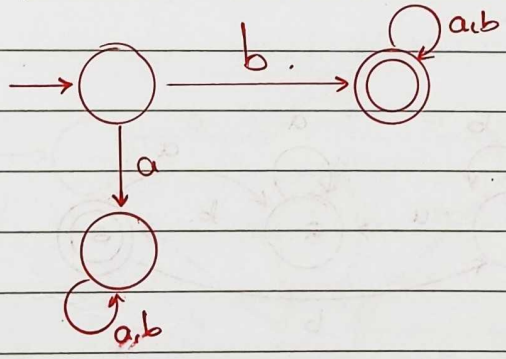
$min = a$

$= (|a|+1)+1.$ ← dead

$= 2+1$

$= 3.$

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



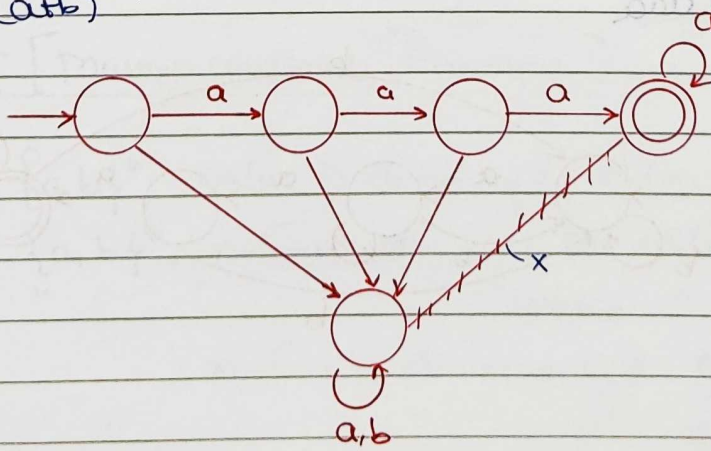
$min = b$

$= |b|+2$

$= 1+2$

$= 3.$

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



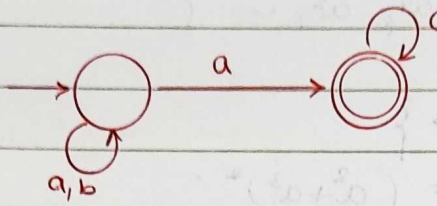
$min = aaa$

$= |aaa|+2$

$= 3+2$

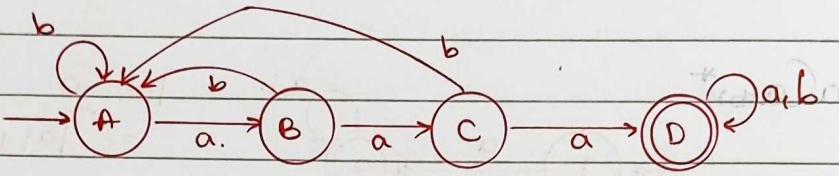
$= 5.$

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



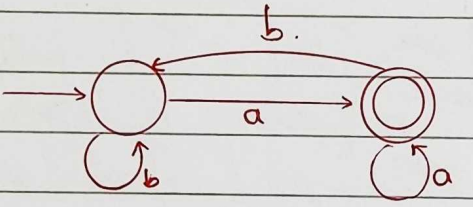
min = a
|a| + 1
= 1 + 1
= 2 State.

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



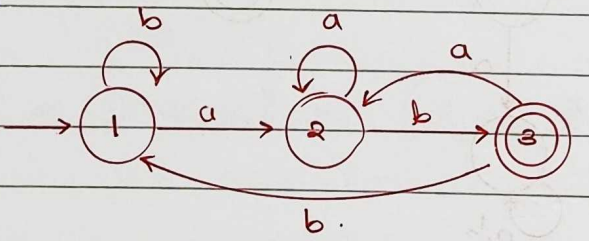
min = aaa
|aaa| + 1
= 3 + 1
= 4.

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



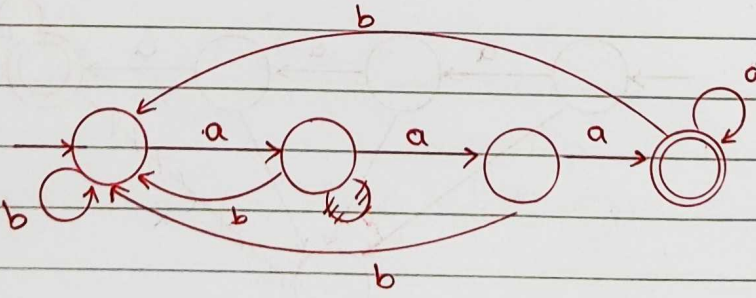
min = a
|a| + 1
= 2 State.

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



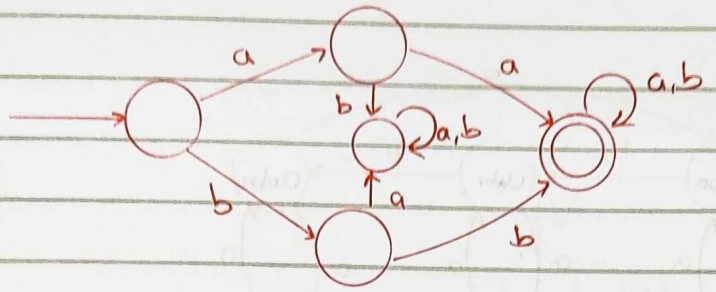
min = ab
|ab| = 1
2 + 1
= 3 State.

8. $L = (a+b)^+ aaa$



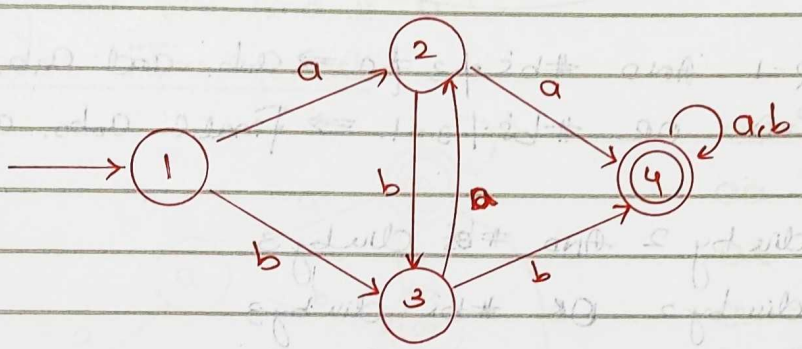
|aaa| + 1
= 3 + 1
= 4 State.

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



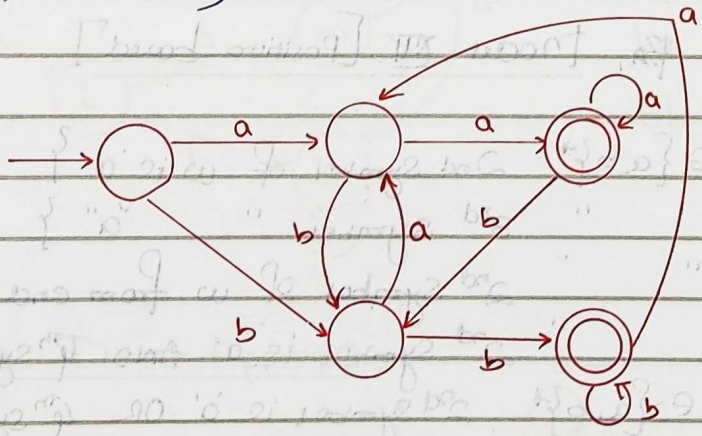
min = aa, bb
= 5 States.

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



min = aa, bb.
= 4 States.

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



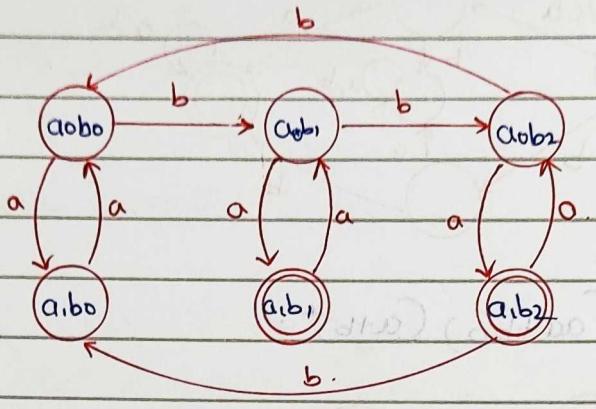
Module - VI [Multiple conditions, Remainders]

1. $\{w \mid w \in \{a,b\}^*$, $n_a(w)$ is divisible by 2, $n_b(w)$ is divisible by 2 $\}$
2. $\{w \mid w \in \{a,b\}^*$, $n_a(w)$ is div by 2 Or $n_b(w)$ is divisible by 2 $\}$
3. $\{$ " " " " but not " " $\}$
4. $\{$ " " $n_a(w)$ is divisible by 3. AND $n_b(w)$ is div by 3 $\}$.

Or
BUT NOT

4. $\{w \mid w \in \{a,b\}^*, n_a(w) \text{ is div by 2 AND } n_a(w) \text{ is div by 3}\}$.

Method 1.



I: #a's %2=1 AND #b's %3 ≠0 ⇒ a_1b_1 and a_1b_2 are finals.

II: #a's %2=0 OR #b's %3=1. ⇒ Final's a_0b_0, a_0b_1 ...

4(i) #a's div by 2 AND #b's div by 3

4.(ii) #a's div by 2 OR #b's div by 3

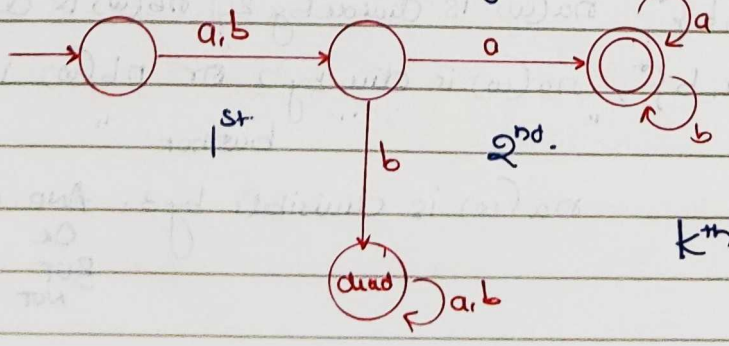
4.(iii) #a's div by 2 BUT NOT #b's div by 3.

Mode III [Position based]

1. $L = \{w \mid w \in \{a,b\}^*$, 2nd Symbol of w is 'a' $\}$
2. $L = \{$ " " 3rd Symbol " "a" $\}$
3. $L = \{w \mid$ " 2nd Symbol of w from end is 'a' $\}$
4. $L = \{$ " " 2nd Symbol is 'a' AND 4th Symbol is 'b' $\}$
5. $L = \{w \mid w \in \{a,b\}^*$, 2nd Symbol is 'a' OR 4th Symbol is 'b' $\}$

1. $(a+b)\,a\,(a+b)^*$.

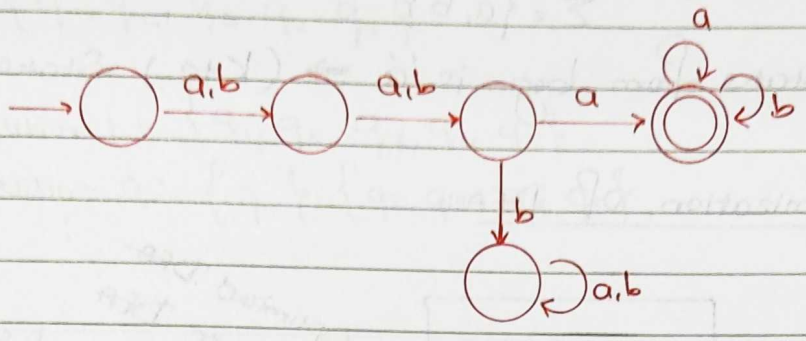
2nd Symbol is 'a'.



min = aa or ba

k^{th} Symbol is 'a'.

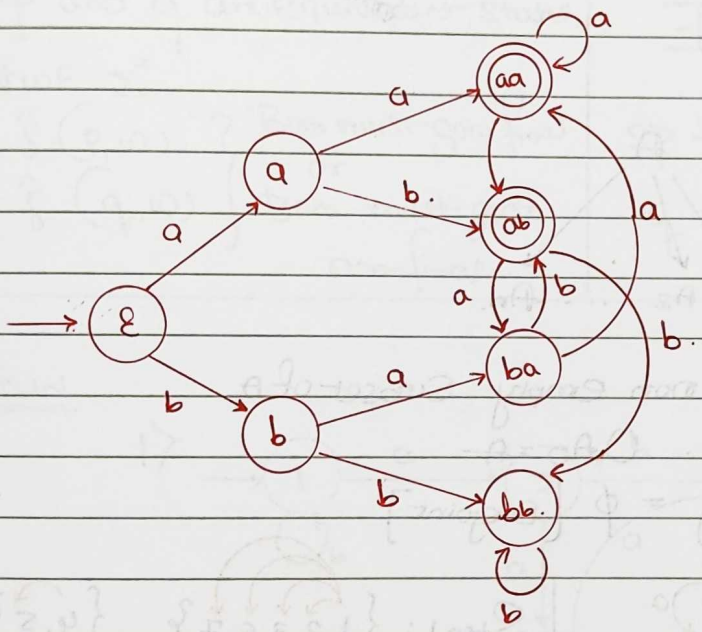
2. $L = (a+b)(a+b)\,a\,(a+b)^*$.



3rd Symbol from begin is 'a'
↓
3+2 = 5 States.

3. 2nd Symbol from end is 'a':

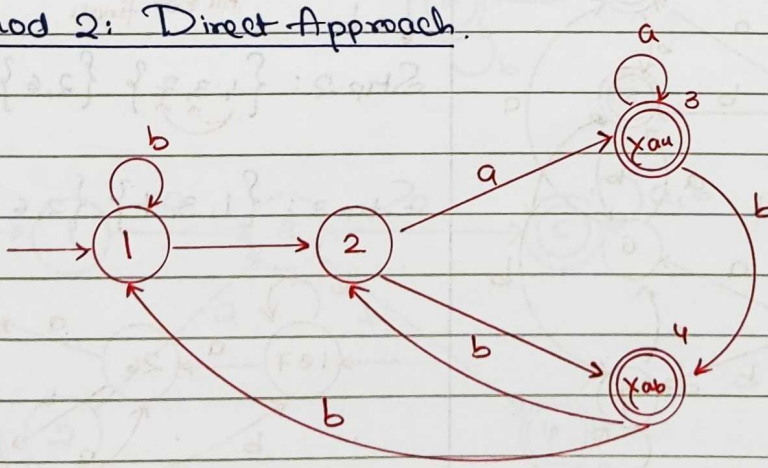
Method 1: Tree Approach:



aa $\xrightarrow{a}$ aa
aa $\xrightarrow{b}$ ab
ab $\xrightarrow{a}$ ba
ab $\xrightarrow{b}$ bb
ba $\xrightarrow{a}$ aa
ba $\xrightarrow{b}$ ab
bb $\xrightarrow{a}$ ba
bb $\xrightarrow{b}$ bb

Trace last 2 Symbols

Method 2: Direct Approach.

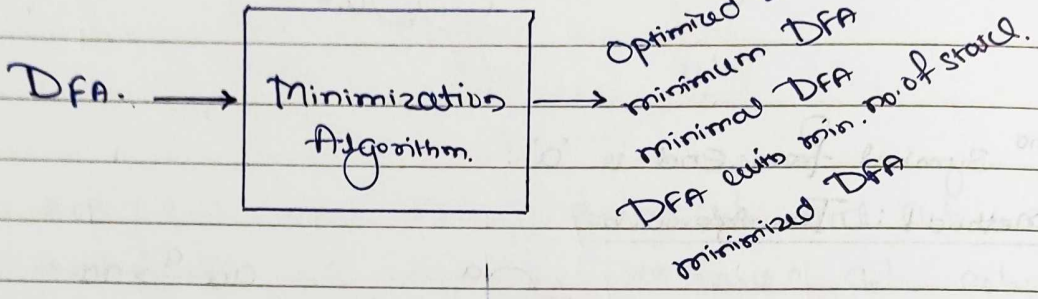


1: waits for a $\frac{a}{b}$
2: wait for a/b
3: Xaa
4. Xab

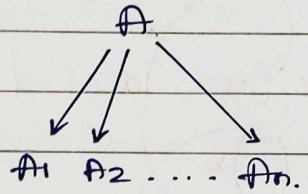
Note: I. K^{th} Symbol from end is 'a' → 2^K state in DFA
$\Sigma = \{a, b\}$.

II. K^{th} State from begin is 'a' ⇒ $(K+2)$ State.

Minimization of DFA:



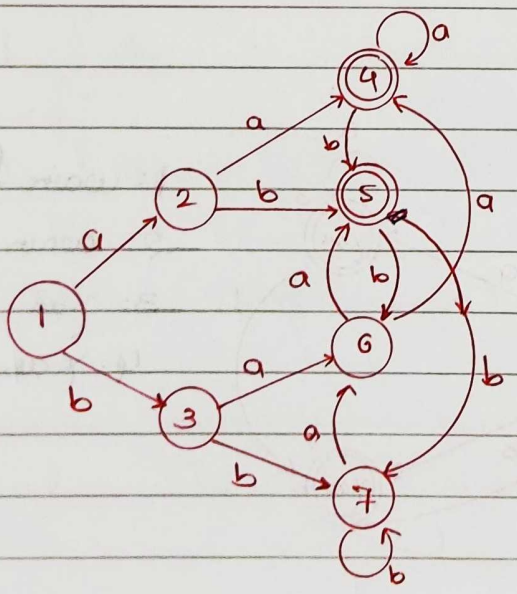
Partition on Set A.



I. Every A_i is non empty subset of A

II. $A_1 \cup A_2 \cup \dots \cup A_n = A$

III. $\forall ij,\ A_i \cap A_j = \phi$ [Disjoint]

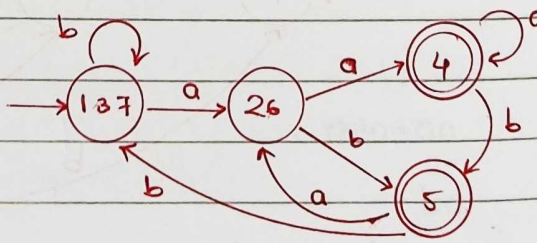


Step 1: {1,2,3,6,7} {4,5}
All non finals. All finals.

↓ a,b

Step 2: {1,3,7} {2,6} {4} {5}.

Step 3: {1,3,7} {2,6} {4} {5}



Partitions on Set Q

$Q = \{q_1, q_2, q_3, q_4, q_5\}$

Partition 1: $\{q_1, q_2, q_3, q_4, q_5\}$

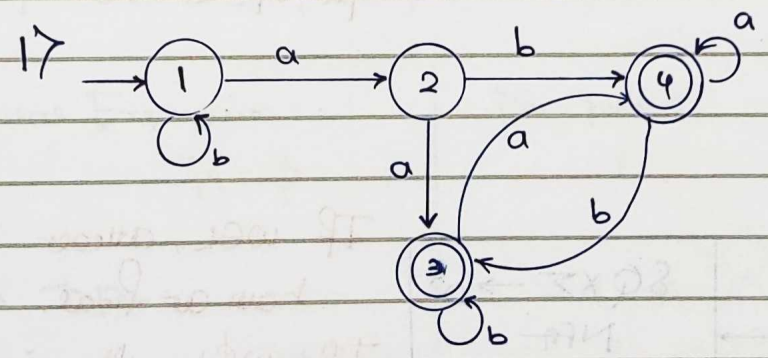
Partition 2: $\{q_1\}\{q_2, q_3, q_4, q_5\}$.

⋮

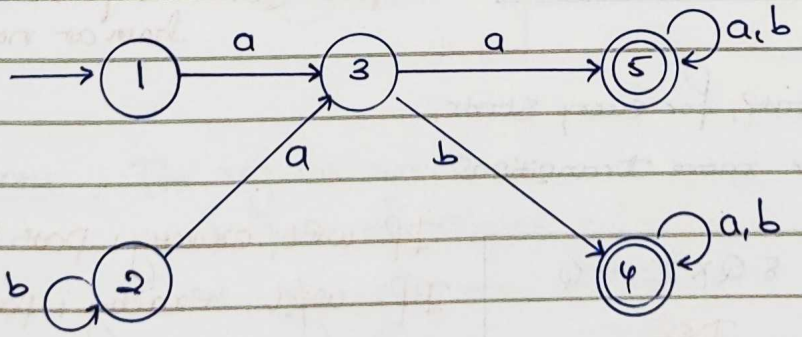
Equivalent States (non distinguishable)	Distinguishable States (non equivalent)
p and q are equivalent states. $\forall w \in \Sigma^*$. $\hat{\delta}(p,w)$ and $\hat{\delta}(q,w)$ } Both must goto final or Both must goto non-final.	$\exists w \in \Sigma^*$. $p \xrightarrow{w}$ final, $q \xrightarrow{w}$ final OR $p \xrightarrow{w}$ non final, $q \xrightarrow{w}$ non final

H.W

1)



2)



Date 10/5/23.

Model - XIII [Special]

1. $\{w \mid w \in \{0,1\}^*$, Decimal(w) is divisible by $8\}$.

   Div by 2^k → $k+1$ States. ($2^3 = 3+1 = 4$ states)

2. $\{w \mid w \in \{0,1\}^*, \#_{01}(w) = \#_{10}(w)\}$.

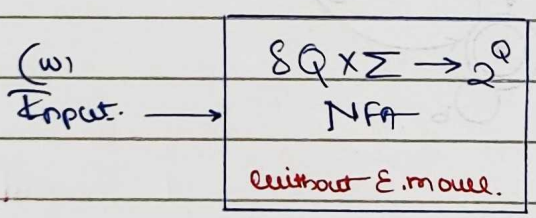
   $= \{\varepsilon, 0, 1, 00, 11, 000 \ldots\}$

3. $\{w \mid w \in \{0,1\}^*$. every prefix p of w satisfied $|n_0(p) - n_1(p)| \leq 1\}$.

   $\{\varepsilon, 0, \ldots\}$.

w	Prefixes	Condition
ε	ε	✓
0	0	$\lvert 1-0\rvert = 1$ ✓
00	00	$\lvert 2-0\rvert = 2$.

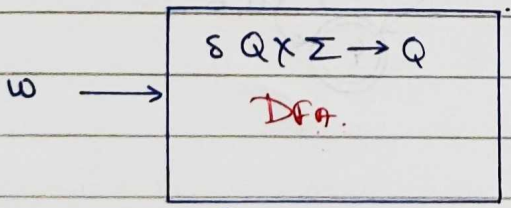
NFA



If $w \in L$, atleast 1 path that halts at final.

If $w \notin L$, there is no path halts at final (either no path or every path halts at non final).

From every state, for every state, Zero or more transitions.



If $w \in L$, exactly 1 path halts at final.

If $w \notin L$, exactly 1 path halts at non final.

For every state, for every i/p Symbol, exactly 1 transition.

Note.

I. Every DFA is NFA

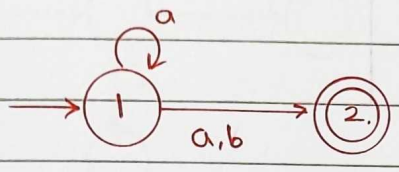
II. NFA may or may not be DFA

III. Min. DFA need not be min NFA.

IV. For every regular language:

η (min DFA) $\geq$ η (min NFA)

eg:



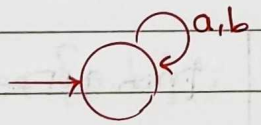
	No. of paths	valid/invalid.
ε.	1	Invalid
a	2	valid
b	1	valid
aa	2	valid
ab	1	valid
ba	0	invalid.

1 $L = \{\}$ ows $\Sigma = \{a, b\}$.

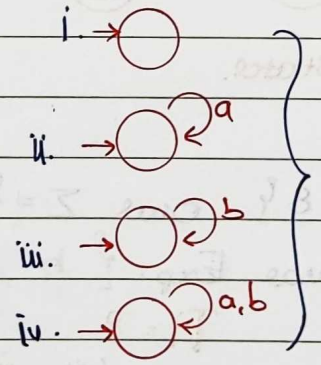
Regular Expression:

$R = \phi$

Min DFA:



Min. NFA:



No. of min NFAs = 4.

Note: For regular set,

i. No. of DFAs = Infinite

ii. No. of min DFAs = 1

iii. No of NFAs = Infinite

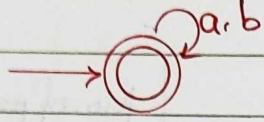
iv. No of min NFAs = 1 or more.

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

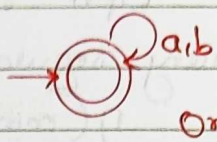
Regular Expression:

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

min DFA:



min NFA:



1 State

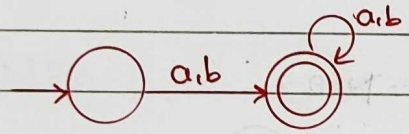
Only 1 min NFA for $L = \Sigma^+$.

3. $L = \Sigma^+$ over $\Sigma = \{a, b\}$.

Regular Expressions:

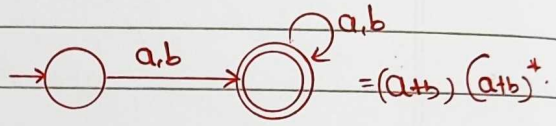
$$R = (a+b)^+ = (a+b).(a+b)^*$$
$$= (a+b)^*.(a+b)$$

min. DFA:

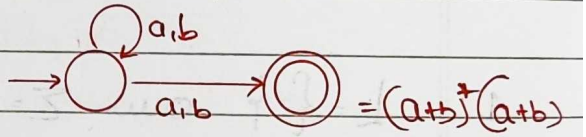


2 States.

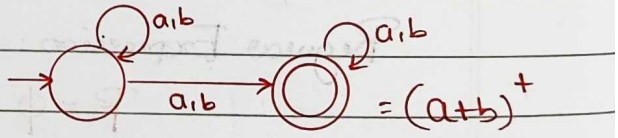
Min NFA: 2 States.



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



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



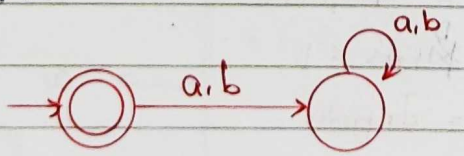
$= (a+b)^+$

4. $L = \{\varepsilon\}$ over $\Sigma = \{a, b\}$.

Regular Exp:

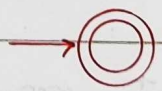
$$R = \varepsilon$$
$$= \phi^* = \varepsilon^* = \varepsilon^+.$$

Min DFA:



2 States.

Min NFA



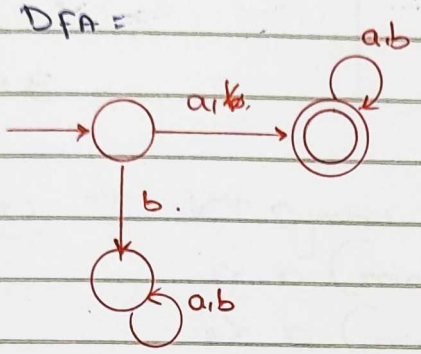
1 State.

5. $L = \{w \mid w \in \{a,b\}^+, w \text{ starts with } 'a'\}$

Regular Expression:

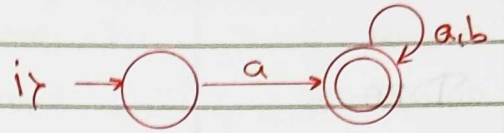
$$R = a(a+b)^*$$
$$= (ab^*)^+$$

Min DFA:

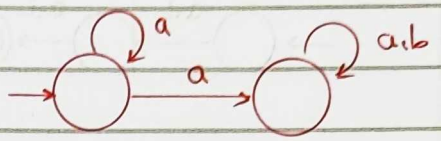


NFA (min):

i)



ii)

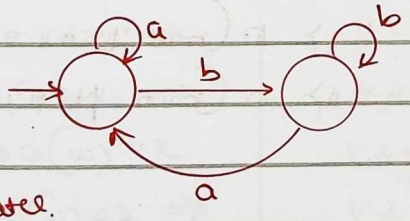


6. $L = \{w \mid w \in \{a,b\}^+, w \text{ ends with } 'b'\}$

Regular Expression:

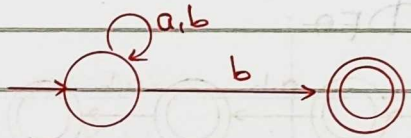
$$R = (a+b)^* b$$
$$= (a^*b)^+$$

DFA (min):



2 states.

NFA (min):



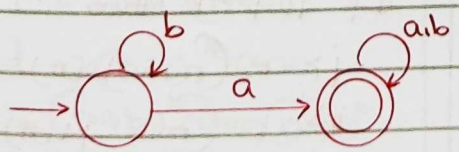
2 states.

7. $L = \{w \mid w \in \{a,b\}^*, w \text{ contains } 'a'\}$

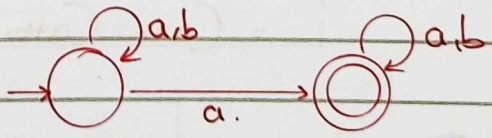
Regular Expression:

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

DFA:



NFA:



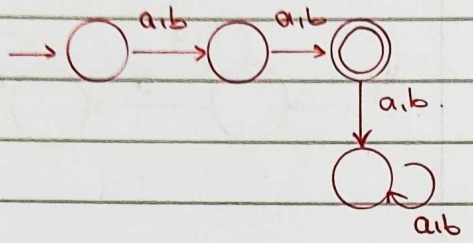
2 states.

8. $L = \{w \mid w \in \{a,b\}^*, |w| = 2\}$.

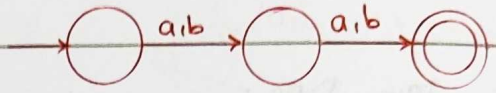
Regular Expression:

$= (a+b)^2$.

DFA:



NFA:



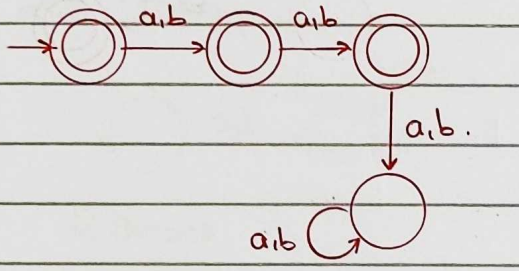
3 States

9. $L = \{w \mid w \in \{a,b\}^*, |w| \leq 2\}$

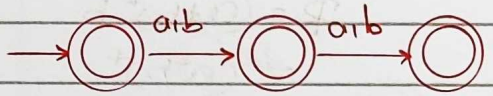
Regular Expression

$= (\varepsilon + a + b)^2$.

DFA:



NFA:



3 States.

If $|w| \leq k$ then

i) n (min DFA) $= k+2$

ii) n (min NFA) $= k+1$.

10. $L = \{w \mid w \in \{a,b\}^*, |w| \geq 2\}$

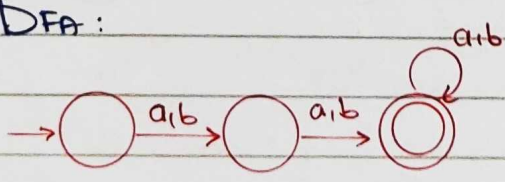
Regular Expression:

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

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

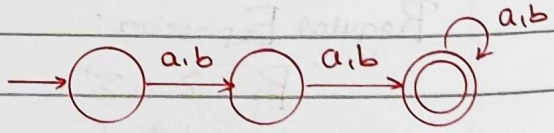
$= (a+b)(a+b)^*(a+b)$

DFA:



3 States.

NFA:



3 States.

If $|w| \geq k$ then

i) n (min DFA) $= k+1$

ii) n (min NFA) $= k+1$

		Min DFA	Min NFA
11	$L = (a+b)^2 a (a+b)^*$	5 States	4 States
12	$L = (a+b)^* a (a+b)^2$	8 States	4 States.
13	$b^* a b^* a b^*$	4 States	3 States
14	$b^* (a+c) b^* (a+\epsilon) b^*$	4 States	3 States
15.	$\Sigma^* a \Sigma^* a \Sigma^*$	3 States.	3 States.

Note: I. K^{th} symbol from begin 'a' $\Sigma = \{a, b\}$

i) n (min DFA) $= K+2$

ii) n (min NFA) $= K+1$.

II. K^{th}. Symbol from end is 'a'

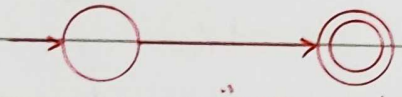
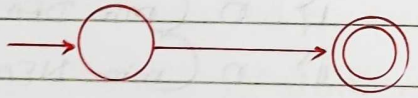
i) n (min DFA) $= 2^K$

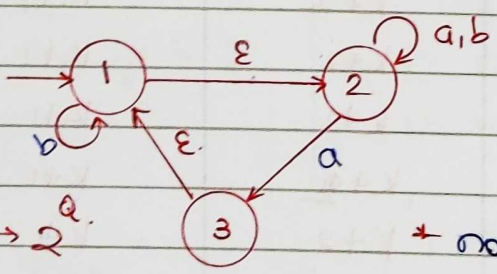
ii) n (min NFA) $= K+1$.

$w \in \{a, b\}^*$.	Min DFA	min NFA.
I. $\|w\| = K =$	$K+2$	$K+1$
II. $\|w\| \leq K \Rightarrow$	$K+2$	$K+1$
III. $\|w\| \geq K \Rightarrow$	$K+1$	$K+1$
IV. $\#a(w) = K$	$K+2$	$K+1$
V. $\#a(w) \leq K \Rightarrow$	$K+2$	$K+1$
VI. $\#a(w) \geq K \Rightarrow$	$K+1$	$K+1$
VII. K^{th} Symbol is 'a'	$K+1$	$K+1$
VIII. K^{th} Symbol from end is 'a'.	2^K	$K+1$.

16.	$L = aaa (a+b)^*$	5	4
17.	$L = (a+b)^* aaa$	4	4
18.	$L = (a+b)^* aaa (a+b)^*$	4	4
19.	$L = (a+b)^2 a (a+b)^3 b (a+b)^*$	9	8

20.	$L = (a+b)a(a+b)^7 a (a+b)^2$.	12	11.
21.	$L = (ab)^*$.	3 States	2 States.
22.	$L = (ba)^*$.	3 States	2 States.
23.	$L = a(ba)^*$.	3 States	2 States.

NFA without ε moves.	NFA with ε moves.
$\delta : Q \times \Sigma \to 2^Q$	$\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$.
	
not NFA without ε moves.	NFA with ε moves.

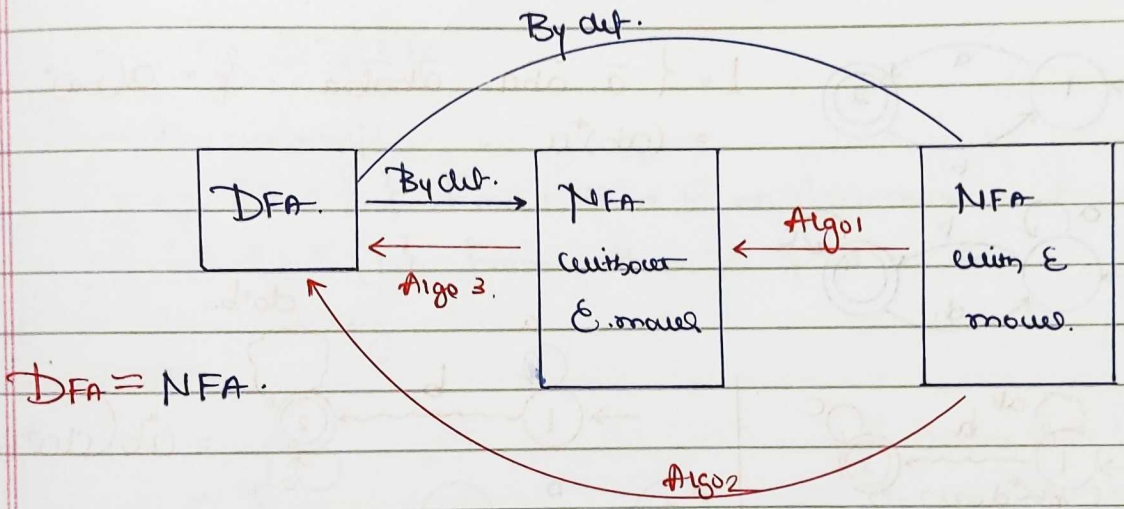


$\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$.

* not DFA
* not NFA without ε moves
* It is NFA.

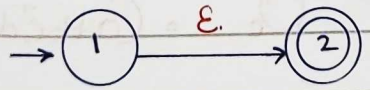
(1, a)	ϕ
(1, b)	{1}
(1, ε)	{2}
(2, a)	{3}
(2, b)	{1, 2}
(2, ε)	{1, 3}
(3, a)	{2, 3}
(3, b)	{1, 2, 3}
(3, ε)	

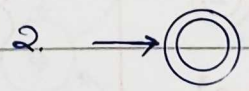
	DFA	NFA
1.	Every DFA is NFA	NFA may or may not be DFA
2.	$\delta: Q \times \Sigma \to Q$	$\delta: Q \times \Sigma_\varepsilon \to 2^Q$
3.	$w \to$ DFA $=$ $w \in L$, ex. one path; $w \notin L$, exactly one path.	$w \to$ NFA $=$ $w \in L$ atleast 1 path; $w \notin L$ 0 or more paths
4.	For every regular, unique min DFA exists.	For every regular, one or more min NFA's exist.

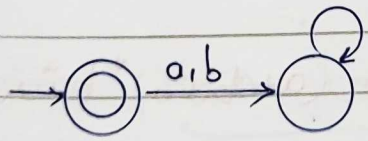


DFA = NFA.

NFA with ε moves:

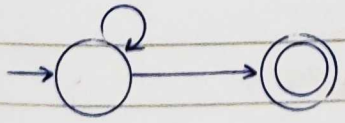
1. 

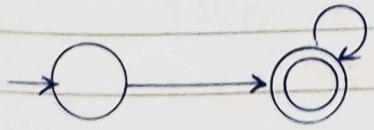
2. 

3. 

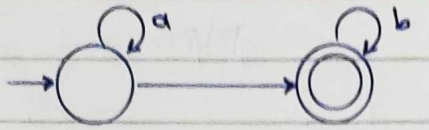
$\delta: Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$

$\delta(1,a) = \{2\}$	$\delta(2,a) = \{2\}$
$\delta(1,b) = \{2\}$	$\delta(2,b) = \{2\}$
$\delta(1,\varepsilon) = \{\}$	$\delta(2,\varepsilon) = \{2\}$

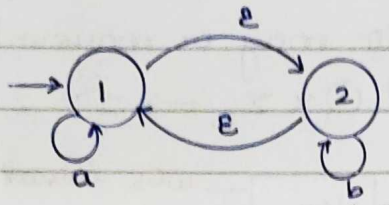
4. 
$L = (a+b)^*$

5. 

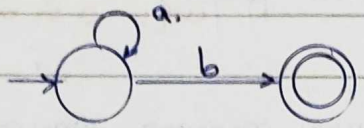
$L = (a+b)^*$

6. 

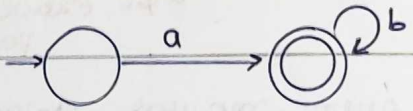
$L = a^*b^*$.

7. 

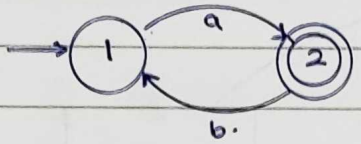
$L = (a+b)^* = (a^*b^*)^*$.
$= (b^*a^*)^*$.

8. 

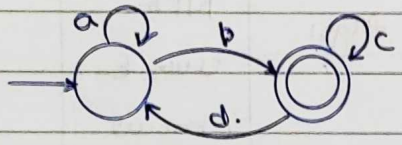
$L = a^*b$

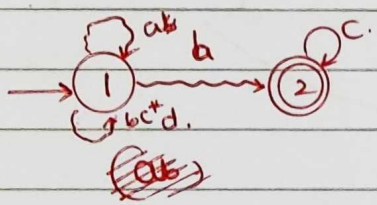
9. 

$L = ab^*$.

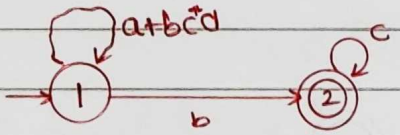
10 

$L = \{ a, aba, ababa \dots \} = a(ba)^*$
$= (ab)^*a$.

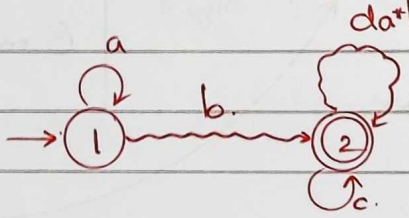
11. 



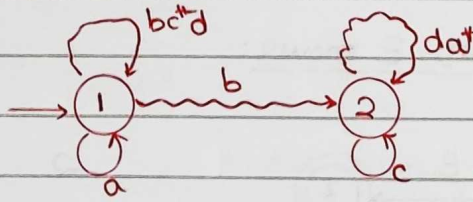
~~(ab)~~



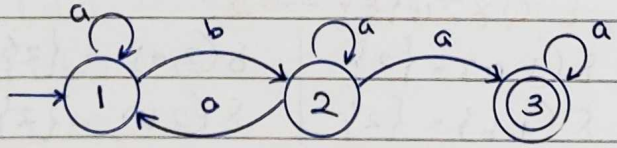
$= (a+bc^*d)^*bc^*$.



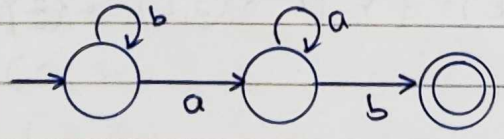
$= a^*b(da^*b+c)^*$.



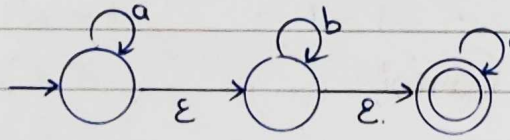
$= \cancel{a^*b(da^*b+c)^*}$
$= (a+bc^*d)b(da^*b+c)^*$

12. 

$L = (a+ba^*a)^*ba^*aa^*$.

13. 

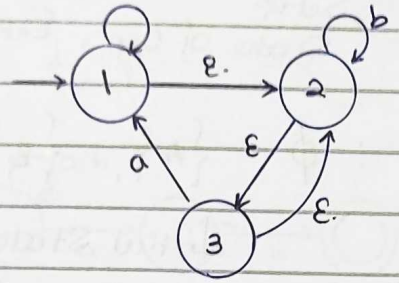
$L = b^*aa^*b$
$= b^*a^+b$

14. 

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

NFA with ε moves:

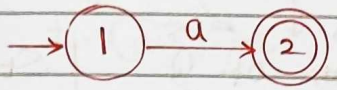
ε- closure (state):



ε-closure (1) = {1, 2, 3}
ε-closure (2) = {2, 3}
ε-closure (3) = {3, 2}.

ε-closure(q) = $\hat{\delta}(q, \varepsilon)$
= Set of all state reachable from state q without reading any string.

eg::

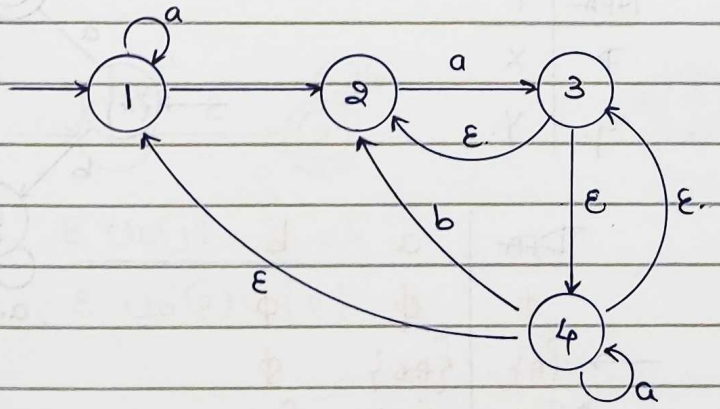


Transition, Symbol/ε

$\delta(1, \varepsilon) = \{\}$ From 1, there is no transition on ε.

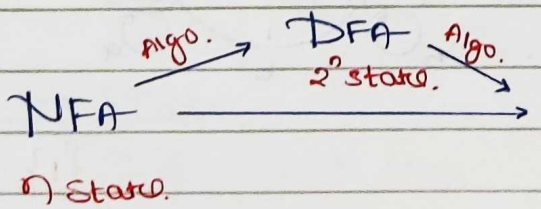
$\hat{\delta}(1, \varepsilon) = \{1\}$ From 1, there is path on ε.

Extended transition. String



ε-closure (1) = {1, 2}
ε-closure (2) = {2}
ε-closure (3) = {3, 1, 2, 4}
ε-closure (4) = {4, 1, 2, 3}.

→ Algo: NFA without ε moves ⇒ DFA
(Subset Construction)

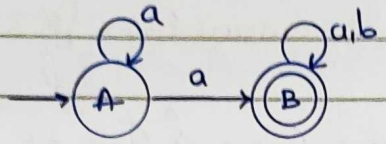


Given NFA ⟶ DFA

$(Q, \Sigma, \delta_{NFA}, q_0, F) \longrightarrow (2^Q, \Sigma, \delta_{DFA}, \{q_0\}, F)$

↓

Set of States of DFA = Set of all Subsets of Q.



NFA

: ϕ $\{A\}$, $\{B\}$, $\{AB\}$

2 States

$Q = \{A, B\}$

Initial States in DFA

$2^Q = \{\phi, \{A\}, \{B\}, \{A,B\}\}$

↓ ↓ ↓

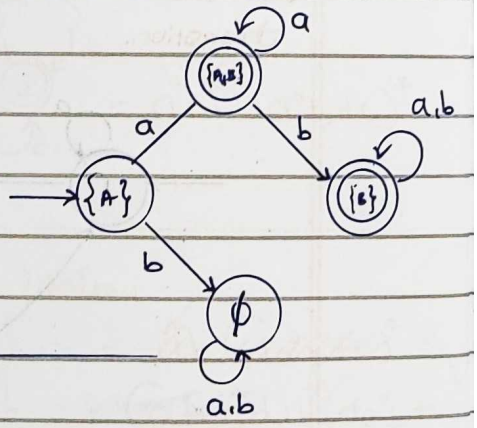
States of Q [State in DFA]

Subsets

→ $\delta_{DFA}(\{p,q\}, i) = \delta_{NFA}(p,i) \cup \delta_{NF}(q,i)$

$= X \cup Y$

NFA	i
p	X
q	Y.

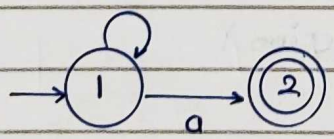


→

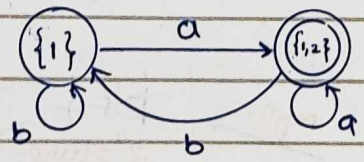
NFA	a	b
→ A	{AB}	ϕ
*B	{B}	{B}

DFA	a	b
ϕ	ϕ	ϕ
→ {A}	{AB}	ϕ
*{B}	{B}	{B}
*{A,B}	{A,B}	{B}

→



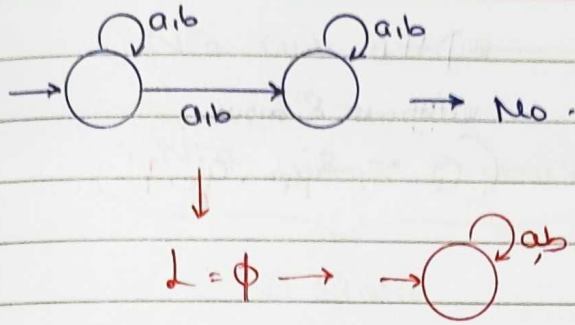
⟹



NFA	a	b
1	{1,2}	{1}
2	ϕ	ϕ

	a	b
{1}	{1,2}	{1}
{1,2}	{1,2}	{1}

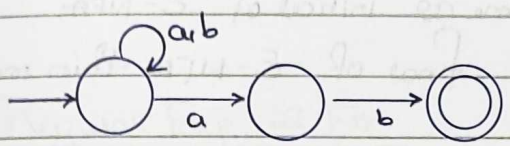
Q1.



→ No. of State in equivalent Min DFA = 1.

$L = \phi$ →

Q2.



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

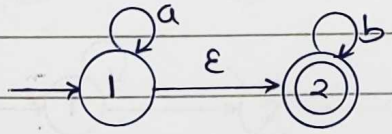
min = ab.

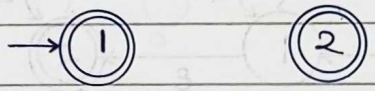
n (min DFA) = 3.

→ NFA → NFA

with ε moves. without ε moves.

$(Q, \Sigma, \delta_{\varepsilon\text{-NFA}}, q_0, F)$ $(Q, \Sigma, \delta_{NFA}, q_0, F')$





Without ε moves.

$\varepsilon\text{-clo}(1) = \{1, 2\}$

$\varepsilon\text{-clo}(2) = \{2\}$.

→ NFA → (Min) DFA.

Inf n State present in NFA then.

$1 \leq$ [no. of State in min. DFA] $\leq 2^n$.

Always important.

→ Min NFA → min DFA

$n \leq$ [no. of State in min. DFA] $\leq 2^n$

→ NFA ⟶ NFA.

with ε moves. without ε moves.

$(Q, \Sigma, \delta_{\varepsilon\text{-NFA}}, q_0, F)$ $(Q, \Sigma, \delta_{NFA}, q_0, F)$.

Note:

I. Initial of NFA is same as initial of ε-NFA.

II. If ε-closure (q) contains final of ε-NFA then make q as final in NFA.

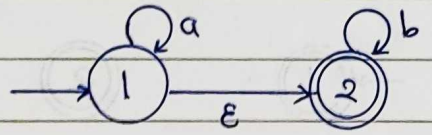
III. $$\delta_{NFA}(q, i) = \varepsilon\text{-clo}\left(\delta_{\varepsilon\text{-NFA}}\left[\underbrace{\varepsilon\text{-closure}(q)}_{1^{st}}, i\right]\right)$$

$\forall q \in Q$ — 2nd

$\forall i \in \Sigma$ — 3rd.

NFA with ε moves ⟶ DFA.

$(Q, \Sigma, \delta_{\varepsilon NFA}, q_0, F) \longrightarrow (2^Q, \Sigma, \delta_{DFA}, \varepsilon\text{-clo}(q_0), F')$



$\delta(1, a) = \{1\}$

$\delta(1, b) = \phi$. $\delta(2, a) = \phi$

$\delta(2, b) = \{2\}$.

ε-clos $(1) = \{1, 2\}$

ε-clos $(2) = \{2\}$.

Note: I. Initial State of DFA = ε-clo (Initial of NFA)

II. $\delta_{DFA}(\{p, q\}, 1) = \varepsilon\text{-clo}\left[\delta_{\varepsilon NFA}(\{p, q\}, 1)\right]$

III. Final of DFA: if any subset contain final of ε-NFA of Q.

NFA with ε-moves
or
NFA without ε-moves

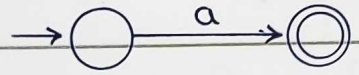
⟹ No. of states in min. DFA is atmost 2^n.

n States — Max no. of state in min. DFA is 2^n

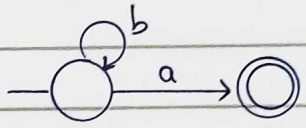
Regular Exp. ≅ FA:

I. FA ⇒ Regular Expression [using State elimination]
II. Regular Expression ⇒ FA [Induction method].

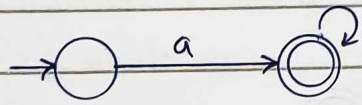
FA → Regular Expression

1. 

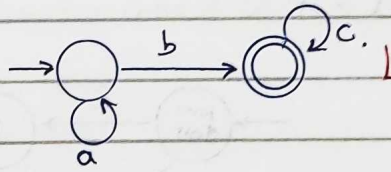
$L_1 = a$.

2. 

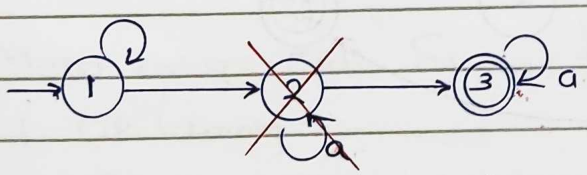
$L_2 = b^*a$

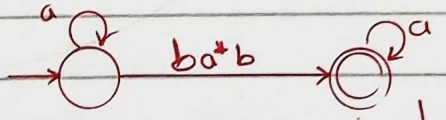
3. 

$L_3 = ab^*$.

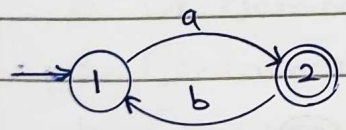
4. 

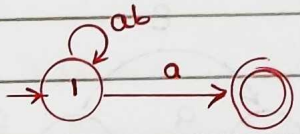
$L_4 = a^*b\,c^*$.

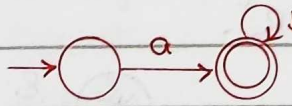
5. 



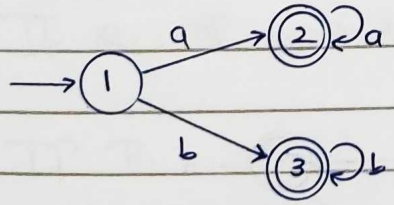
$L = a^*ba^*ba^*$.

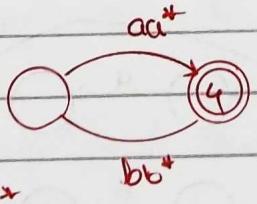
6. 



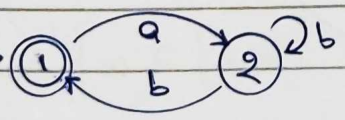
$= (ab)^*a$


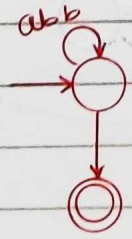
$= a(ba)^*$.

7. 



$L = aa^+ + bb^+$
$= a^+ + b^+$.

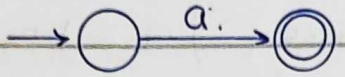
8. 



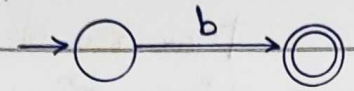
$L = (ab^*b)^*$.

Regular Expression → FA.

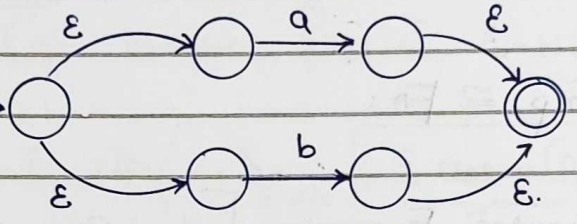
1. $R = a$



2. $R = b$

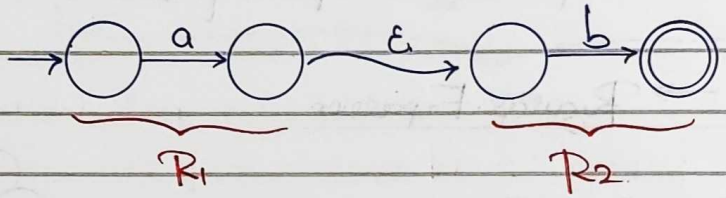


3. $R = a+b$
$= R_1 + R_2$.

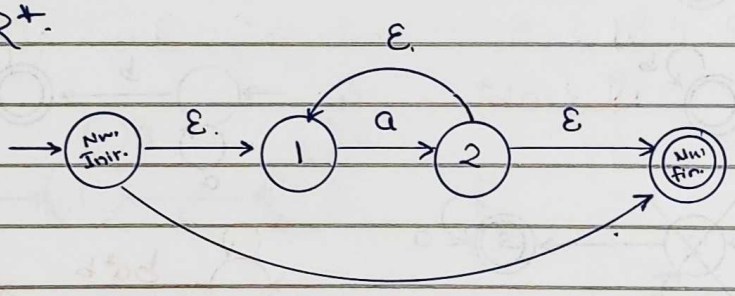


4. $R = a.b$

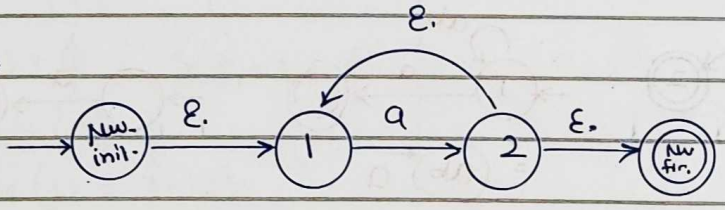
Q1. $R_1.R_2$.



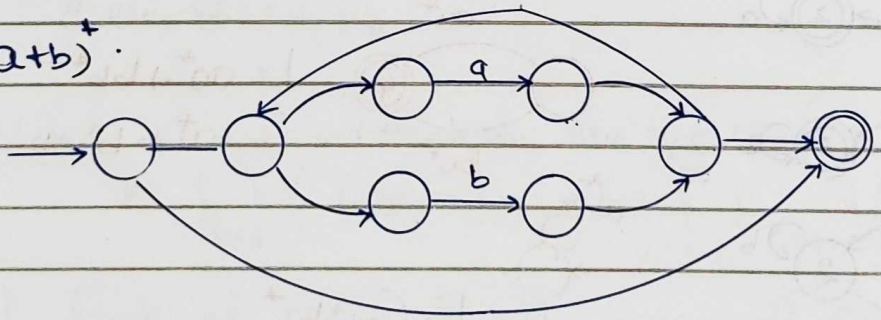
5. $a^+ = R^+$.



6. $a^+ = R^+$.



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



Q1. Number of prefixes of "n" length string is $n+1$

Q2. $((ab)^* \cdot \phi)$ is equivalent to ϕ.

Q3. $(\phi^* \cup (bb^*)) = b^*$.

Q4. Which of the following is true?
$(ab)^* a = a(ba)^*$.

Q5 OR operator in regular expressions satisfies.
Associative and Commutative

Q6. Concatenation Operator in regular expression satisfies.
Associative property.

Q7. Which of the following distribution is valid in regular exp.?
Concatenation over OR

Q8. Match groups over Sigma = {a, b}.

1. OR identity	Epsilon
2. OR dominator	Empty Expression
3. Concatenation identity	$(a+b)^*$
4. Concatenation dominator	$(a a)^*$.

Q9. If $R_1 = a^*$, and $R_2 = (aa)^*$ then $R_1 \cdot R_2 = R_1$.

Q10. If $R_1 = a(aa)^*$, and $R_2 = (aa)^*$ then $R_1 + R_2 = R_1^*$.

Q11. $L = \{ w \in \{a, b\}^* : \#_a(w) \leq 3 \}$.

$b^*(a \cup \varepsilon) b^* (a \cup \varepsilon) b^* (a \cup \varepsilon) b^*$.

Homework:

Find the min. string in the following expressions:

1. $(ab)^+ aaa(b+ab) \longrightarrow$ min: abaaaab
2. $(ab)^+ aaa\,(b+\varepsilon) \longrightarrow aaa$
3. $(a+ab)^+ (bb+aaa)^+ (ab)^* \longrightarrow abb$
4. $(ab)^+ + \varepsilon \longrightarrow \varepsilon$.
5. $(ab)(a+b)^* aaa + ba \longrightarrow ba$
6. $[(ab^* + a^+ + ba)ab]^+ (a+b)^* \longrightarrow aab$
7. $[((ab)^+ a)^+ aaa]^+ \longrightarrow aaa$
8. $[((aba)^+ aba)^+ ab]^* \longrightarrow \varepsilon$
9. $(a^*)^+ \longrightarrow \varepsilon$.
10. $((ab)^+ a^+)^+ bb^+ aa^+ \longrightarrow abb^+ aa^+$.
11. $[(ab)^+ aba]^+ aaa \longrightarrow ababaaaaa$
12. $(aaa^+ aba^*) \longrightarrow aaab$
13. $(a+aaa+aa) \longrightarrow a$
14. $a^+ b^+ \longrightarrow ab$
15. $a^+ b^* \longrightarrow \varepsilon$
16. $a^* b^* c^+ d^* \longrightarrow c$
17. $(a+b^*+c^+)^+ \longrightarrow \varepsilon$
18. $(a+b+c)^+ \longrightarrow \varepsilon$

Q12. $L = \{w \in \{a,b\}^* : \#a(w) >= 3\}$.

$(a \cup b)^* a (a \cup b)^* a (a \cup b)^* a (a \cup b)^*$.

Q13. $L_1 = a^* b^*$. $L_2 = a^+ b^+$.

Find $L_2 - L_1 \longrightarrow$ None

Q14. $L_1 = a^+ + b^+$ and $L_2 = a^* b^*$

Which of the following is true? $\longrightarrow L_1^* = L_2^*$

Q15. $L = a^* + b^*$ and $L_2 = a^* b^*$.

Which of the following is true?

L_1 is subset of L_2.

Q16. $L_1 = a^+b^+$ and $L_2 = a^*b^*$.
Which of the following is FALSE? $L_1^* = L_2^*$.

Q17. $L_1 = a^+ + b^+$ and $L_2 = a^* + b^*$.
Which of the following is TRUE? $L_1 \cup L_2 = L_2$.

Q18. $L_1 = a^+$ and $L_2 = a^*$.
Which of the following is TRUE? $L_1^+ = L_2^+$.

Q19. $A = a^*$ and $B = b^+$.
$AB = ?$ $\{a^m b^n \mid m, n \geq 0\}$

Q20. $A = aa^*$ and $B = bb^*$ $(A \cup B)^* = ?$ $(a+b)^*$

Q21. Given the language $L = \{ab, aa, baa\}$, which of the following strings are not in L^*?
baaaaabaaaab.

Q22. The length of the shortest string NOT in the language (over $\Sigma = \{a, b\}$) of the following regular expression is $a^*(ba)^*a^*$ is 1.

Q23. $L = a(a+b)^*$ is equivalent to $(ab^*)^+$, $(a^+b^+)^+$, $a^*(ab^*)^+$.

Q24. $L = (a+b)^*b$ is equivalent to $b^*(ab^*)^*b$

Q25. $(b+ba)(b+a)^*(ab+b)$ $b(a+b)^+b$

Q26. $\{w \in \{a, b\}^* : \#a(w) = 3n\}$
$(b^*ab^*ab^*a)^*b^*$.

Q27. $(a \cup b)^*(a \cup \varepsilon)b^+ \rightarrow (a+b)^*$.

Q28. $L = \{w \in \{a,b\}^* \mid w$ has bba as a substring$\}$
Which of the following describes L?
$(a \cup b)^* bba (a \cup b)^*$.

Q29. $L = \{w \in \{a,b\}^*\}$.
How many of below are equivalent to given L?
$(a+b)^*$ $\quad (a^*b^*)^*$
$(a+b+\varepsilon)^+$ $\quad (b^*a^*)^*$
$\varepsilon + (a+b)^+$. $\quad \therefore$ 5 are equivalent.

GATE PYQs

1. Which two of following regular expressions are valid?
Ans. $(00)^*(\varepsilon+0) = 0^* \longrightarrow 0^*$ $\quad \therefore$ (i) and (iii)

2. If the regular set A is represented by $A = (01+1)^*$ and regular set 'B' is represented by $B = ((01)^*1)^*$, which of the following is true?
Ans. $01 = x$
$1 = y$. $\quad A = (x+y)^*$
$B = (x^*y^*)^* \quad \therefore A = B$.

3. The string 1101 does not belong to the set represented by
Ans. c. $(10)^*(01)^*(00+11)^*$.
d. $(00 + (11)^*0)^*$.

4. Let S and T be languages over $\Sigma = \{a,b\}$ represented by the regular expressions $(a+b^*)^*$ and $(a+b)^*$, respectively. Which of the following is true?
Ans. $S \subseteq T$ $\quad (a+b^*)^* = (a+b)^*$
$S \supseteq T$ $\quad (a^*+b)^* = (a+b)^*$. $\quad \therefore S = T$
$T \subseteq S$

HW

5. Consider the set Σ^* of all strings over the alphabet $\Sigma = \{0, 1\}$ Σ^* with the Concatenation operator for strings.

6. The regular expression $0^*(10^*)^*$ denote the same set as.

Ans $(1^*0)^*1^* = (0+1)^*$.

7. Which of the following languages over alphabet $\{0,1\}$ is

Ans described by regular expression $(0+1)^*0(0+1)^*0(0+1)^*$

The set of all Strings Containing atleast two 0's.

8. Consider the language $L_1 = \phi$ and $L_2 = \{a\}$. Which one of the following represents $L_1L_2^* \cup L_1^*$?

Ans $\{\varepsilon\}$.

9. The length of the Shortest String NOT in the language (over $\Sigma = \{a, b\}$) of the following regular expressions $a^*b^*(ba)^*a^*$.

Ans 3

10. Which of the following regular expression represents the language: the set of all binary strings having two consecutive 0's and two consecutive 1's?

Ans $(0+1)^*(00(0+1)^*11 + 11(0+1)^*00)(0+1)^*$.

11. Let $r = 1(1+0)^*$, $s = 11^*0$ and $t = 1^*0$ be three regular expressions. Which one of the following is true?

Ans $L(s) \subseteq L(r)$ and $L(s) \subseteq L(t)$

$L(s) \subseteq L(t)$ and $L(s) \subseteq L(r)$

$r = 1(1+0)^*$ } $s \subseteq r$

$s \subseteq t$ { $s = 1^*0$, $t = 1^*0 = 0 + 1^+0$

12. Which of the following regular expression identities are true?

Ans. $(r^*s^*)^* = (r+s)^*$

13. Which one of the following regular expressions represents the set of all binary strings with an odd number of 1's?

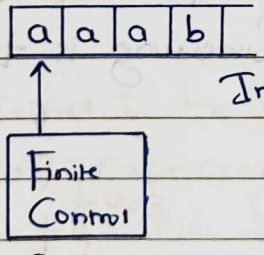
Ans. $= 0^*(0^*10^*10^*)^*0^*10^*$.

14. Which one of the following regular expressions over $\{0,1\}$ denotes the set of all strings not containing 100 as a substring?

Ans. $0^*(10+1)^*$.

Acceptor	Transducer
Input → FA → Accepted / Rejected.	Input → FA → Output
FA without O/p: — NFA — DFA.	FA with O/p: — Moore M/c — Mealy M/c
Applications: Token recogniser, Spell checker.	Applications: Sequence detectors, Generators, Inc/Dec, Addition .. etc.

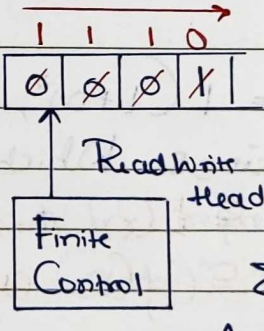
FA Configuration.



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

δ DFA $Q \times \Sigma \rightarrow Q$

δ NFA $Q \times \Sigma_\varepsilon \rightarrow 2^Q$.



$\Sigma = \{0,1\}$ = I/p alphabet

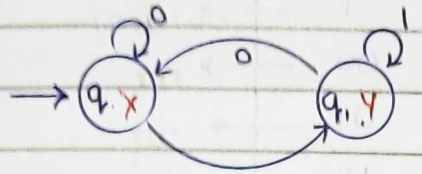
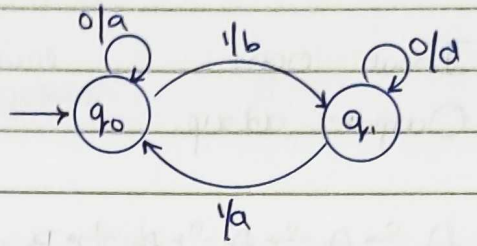
$\Delta = \{0,1\}$ = O/p alphabet.

$FA = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$

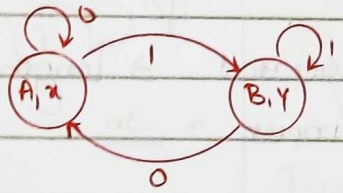
$\delta\ Q \times \Sigma \rightarrow Q$ (DFA)

λ : O/p function

Δ : O/p alphabet

Moore M/C	Mealy M/C.
1. $\lambda : Q \to \Delta$	$\lambda : Q \times \Sigma \to \Delta$
2. Output is associated with State.	Output is associated with transition.
3. 	
$\Delta = \{x, y\}$ $\lambda(q_0) = x$ $\lambda(q_1) = y$.	$\lambda(q_0, 0) = a$ $\lambda(q_1, 0) = d$ $\lambda(q_0, 1) = b$ $\lambda(q_1, 1) = a$.

Moore Machine.



ε : A (x)

0 : A (x) $\xrightarrow{0}$ A (x)

A → B → B

x y y

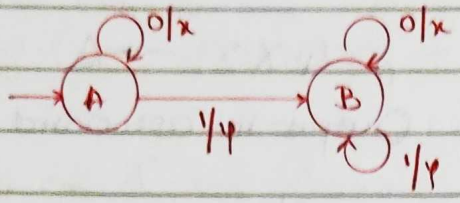
By default, o/p associated with initial state will be produced.

Input.	Output
ε	x
0	xx
1	xy
00	xxx
01	xxy
10	xyx
11	xyy

I. If n length input is given then output length for given input will be n+1

II. If n length input is given and every state is associated with 3 length output then output for the given i/p : 3*(n+1)

Mealy Machine:



I/P	O/P
ε	nothing
0	x
1	y
00	xx
01	xy
10	yx
11	yy

Input: 0001
Output: xxxy.

$A \xrightarrow[x]{0} A \xrightarrow[x]{0} A \xrightarrow[x]{0} A \xrightarrow[y]{1} B$

I. If n length input given then Output length for the given input = n

II. If every transition is associated 3 length Output then Output length for n length input = 3n.

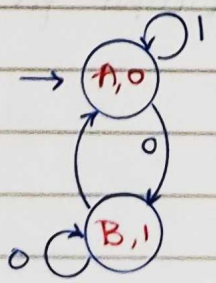
Moore M/C Mealy M/C

Common points between both
- $\delta: Q \times \Sigma \to Q$
- Both are DFA's
- Final States are not there.

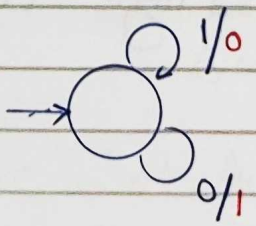
(1.) 1's Compliment.

011 → FA → 100

Moore M/C.



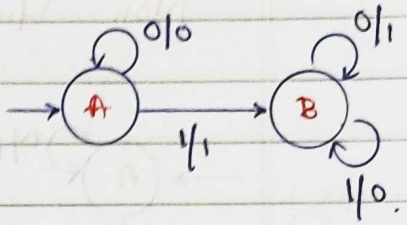
Mealy M/C



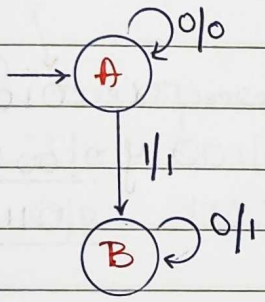
(2) 2's Complement.

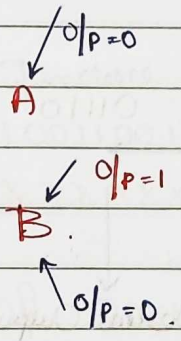
Reverse
001010 → FA → 0011010
Actual input.
101010
↓ Reverse
0101100
Actual output.

Mealy M/c

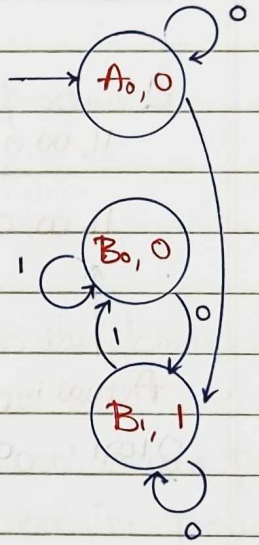


Mealy M/c ⟶ Moore M/c.



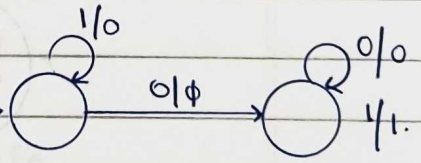


Find output which are incoming towards every state.

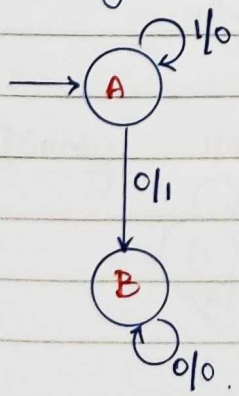


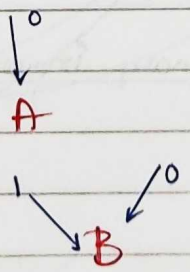
(3) Increment of Binary.

1101000 → FA → 0011000
(Reversed) (Reverse)

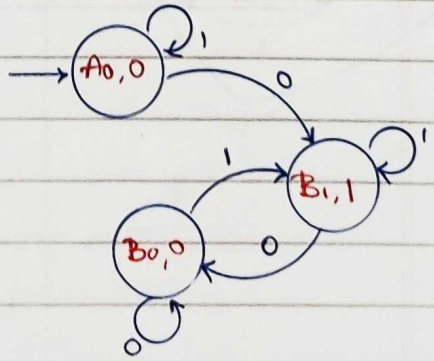


Mealy:



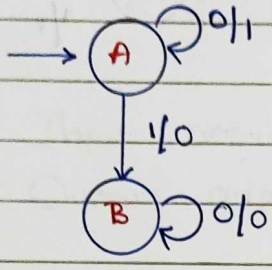


Moore M/c



4. Decrement of Binary Number $f(x) = x - 1$.

Note: Given input should be in reverse Order.



000110 — Search for first "1".

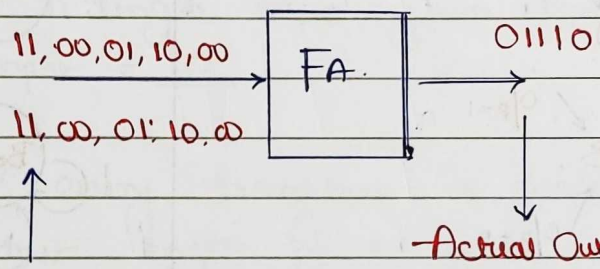
1

000101

Same. Comp

0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 0 (carry)

5. Addition of 2 Binary inputs



Actual input.

01001 00101.

Actual Output.

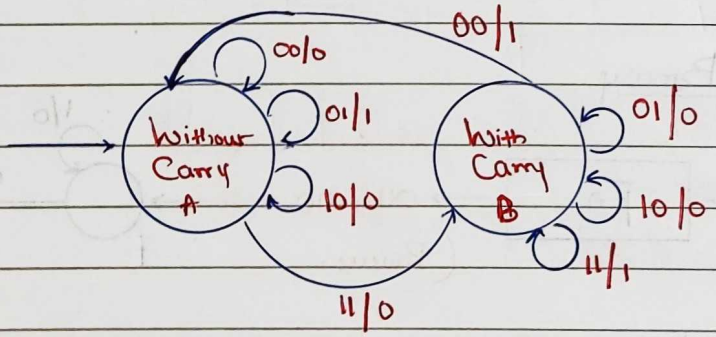
01110

5th 1st.

x = 01001

y = 00101

01110 → with carry.



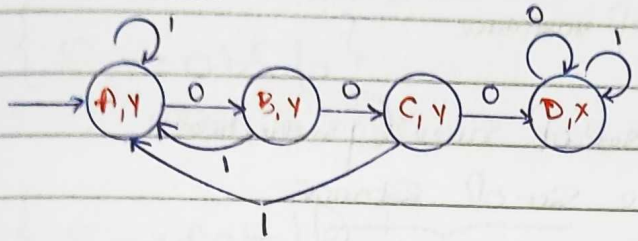
HW.

6. Substraction of two Binary Inputs

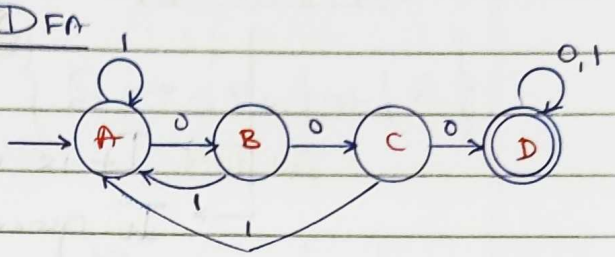
(7.) Sequence Detector.

Detect '000' Sequence in the given binary input.

Moore M/C



DFA



Note: Containing '000' as Substring → DFA.

8. If '000' present, Produce X. Otherwise Produce Y.

I/P: 001000100111001.

O/P: YYYYYXXXXXXXX

(8.) Find the number of occurences of '000' in the binary input.

I/P: 01001000100111 0.

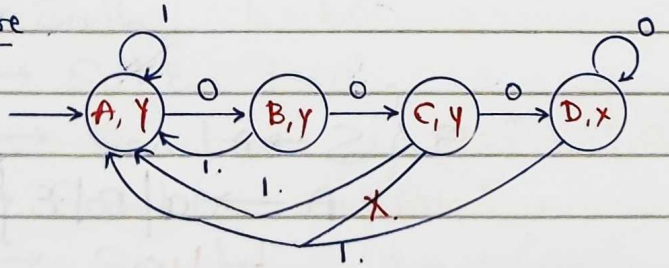
O/P: YYYYYYXXXYYYXYYY.

3 times.

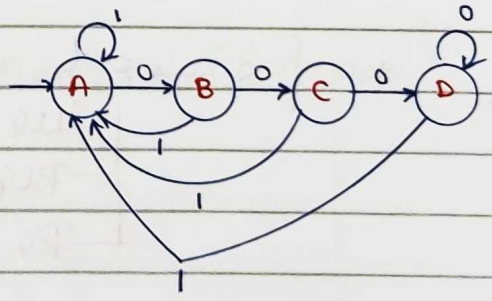
If '000' occurs produce 'X' otherwise produce 'Y'.

Note: Ending with '000' → DFA

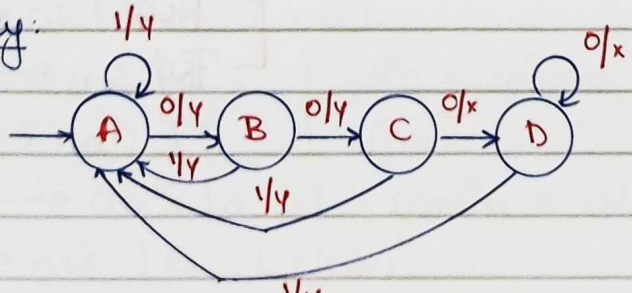
Moore



DFA:



Mealy:



HW

(9.) Produce the Sum of present bit and previous bit of input.

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

- V → Set of variables
- T → Set of terminals
- S → Set of Symbols
- P → Set of rules.

→ It is a set of rules (productions)

→ It generate Set of Strings (Language)

Regular Grammar (RG)

→ It is left linear grammar (LLG) or Right Linear grammar (RLG)

LLG	RLG
$V \rightarrow VT^* \mid T$	$V \rightarrow T^*V \mid T$
Example: $S \rightarrow Sab \mid \varepsilon \mid ab$	Example: $S \rightarrow E \mid a \mid ab \mid aabS$.

Identify RG.

1. $\{ S \rightarrow \varepsilon. \}$
   - LLG
   - RLG
   - RG

2. $\{ S \rightarrow a \mid bc \mid \varepsilon. \}$
   - LLG
   - RG
   - RLG

3. $\{ S \rightarrow A$
   $A \rightarrow a \mid S \mid \varepsilon. \}$
   - LLG
   - RLG
   - RG.

4. $\{S \to Sa,\ S \to bS,\ S \to a\}$ Neither LLR or RLG.

5. $\{S \to Sa|b\}$ LLG ✓ RLG ✗ RG ✓

6. $\{S \to abS|a\}$ RLG and RG.

7. $\{S \to aSb|ab|\varepsilon\}$ not RG

8. $\{S \to AaB|\varepsilon\}$ not RG

9. $\{S \to ABa|\varepsilon\}$ not RG

Find Regular Language from following RGs.

1. $S \to \varepsilon$. $L = \{\varepsilon\}$.
2. $S \to a|bb|\varepsilon$ $L = \{\varepsilon, a, bb\}$.
3. $S \to \varepsilon|a|aa|aaa$ $L = \{\varepsilon + a + aa + aaa\} = \{a^n | n \le 3\}$
4. $S \to Aa$ $L = \{\} = \phi$.
5. $S \to Aa|b$ $L = \{b\}$
6. $S \to Aa$ $A \to \varepsilon$. $L = \{a\}$
7. $S \to Aa$ $A \to a|b|\varepsilon$. $L = (\varepsilon + a + b)a = a + aa + ba$
8. $S \to Aa|Bb$ $A \to c$ $B \to d$ $L = \{ca, db\}$.
9. $S \to AA|Bb$ $A \to \varepsilon|a$ $B \to \varepsilon|b$. $L = \{aa, a, bb, b\}$
10. $S \to Sa|b$ $L = ba^*$.
11. $S \to Sa|\varepsilon$ $L = a^*$.
12. $S \to Saa|\varepsilon$ $L = (aa)^* = \{a^{2n} | n \ge 0\} = \{a^n | n = \text{even}\}$
13. $S \to Sab|\varepsilon$ $L = (ab)^*$.
14. $S \to Sab|c$ $L = c(ab)^*$.
15. $S \to aS|b$. $L = a^*b$
16. $S \to aS|a$ $L = a^*a = aa^* = a^+$.
17. $S \to aaS|\varepsilon$. $L = (aa)^*$
18. $S \to aaS|a$ $L = (aa)^*a = a(aa)^*$.
19. $S \to abS|\varepsilon$ $L = (ab)^*$.

20. $S \to Sa \quad L = \phi$.

21. $S \to Sa \mid b \mid c. \quad L = (b+c)a^* = ba^* + ca^*$.

22. $S \to abS \mid c \mid de \mid fgh. \quad L = (ab)^*(c+de+fgh)$.

23. $S \to Sa \mid b \mid c \quad L = (b+c)a^*$

24. $S \to Sa \mid \varepsilon \mid b \quad L = (\varepsilon+b)a^*$.

25. $S \to Sa \mid Sb \mid c. \quad L = c(a+b)^*$.

26. $S \to Sa \mid Sb \mid \varepsilon \quad L = \varepsilon(a+b)^* = (a+b)^*$.

27. $S \to aS \mid bS \mid \quad L = (a+b)^*\varepsilon = (a+b)^*$.

28. $S \to Sa \mid Sb \mid a \quad L = a(a+b)^*$

29. $S \to aS \mid bS \mid b \quad L = (a+b)^*b$.

30. $S \to Aa \quad A \to Aa \mid Ab \mid \varepsilon. \quad L = (a+b)^*a$.

31. $S \to aA \quad A \to ab \mid bA \mid \varepsilon. \quad L = a(a+b)^*$.

32. $S \to Sa \mid Sb \mid a \quad L = a(a+b)^*$.

33. $S \to aS \mid bS \mid a. \quad L = (a+b)^*a$.

34. $S \to Sa \mid Sb \mid A \quad A \to aa \mid ba \quad L = S(a+b)^* = (aa+ba)(a+b)^*$

35. $S \to Aa \mid Ab \quad A \to Ba \quad B \to Ba \mid Bb \mid \varepsilon$.

$S = L = A(a+b)$

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

36. $S \to aS \mid bS \mid cS \mid \varepsilon. \quad L = (a+b+c)^*$.

37. $S \to Aa \mid Ab \mid \varepsilon \quad A \to a \mid b \mid \varepsilon$

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

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

38. $S \to Sa \mid Sb \mid A \quad A \to Bab \quad B \to Ba \mid Bb \mid \varepsilon$.

$L = S.(a+b)^* = A.(a+b)^* = (a+b)^*ab(a+b)^*$

39. $S \to aS \mid bS \mid A \quad A \to abB \quad B \to aB \mid bB \mid \varepsilon$.

$L = (a+b)^*A = (a+b)^*ab(a+b)^*$.

40. $S \to aA \mid bA \quad A \to aB \mid bB \quad B \to aB \mid bB \mid \varepsilon$.

$S = (a+b)A. = (a+b)^2(a+b)^*$.

41. $S \to Aa \mid Ab \quad A \to Ba \mid Bb \quad B \to Ba \mid Bb \mid \varepsilon$

$S = A(a+b) = (a+b)^*(a+b)^2 = (a+b)^2(a+b)^*$.

42. $S \to Sa \mid Sb \mid \varepsilon \mid A \quad A \to Aa \mid Ab \mid \varepsilon$.

$S = S.(a+b)^* = (\varepsilon+A)(a+b)^* = (a+b)^*.(a+b)^* = (a+b)^*$.

43. $S \to Sa \mid Ab \mid c \quad A \to Sb \mid d$.

$L = S(a+bb)^* = (db+c)(a+bb)^*$.

44. $S \to Saa \mid Ab \quad A \to Sb \mid Ac \mid e$.

Step 1: $A \to Ac \mid Sb \mid e$

$A = (Sb+c)c^* = Sbc^* + ec^*$.

Step 2: $S \to Saaa \mid Ab$

$S \to Saa \mid Sbc^*b \mid ec^*b \quad L = ec^*b(aa+bc^*b)^*$.

45. $S \to Sa \mid Saa \mid \varepsilon \quad L = a^* = (a+aa)^*$

46. $S \to Sa \mid Sb \mid a \mid b \quad L = (a+b)^+$

47. $S \to aS \mid bS \mid a \mid b \mid \varepsilon \quad L = (a+b)^*S = (a+b)^*(a+b+\varepsilon)$

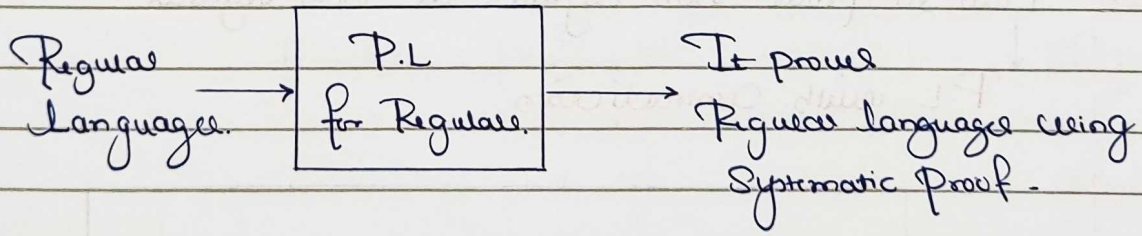
48. $S \to aS \mid bS \mid abS \mid \varepsilon \quad L = (a+b+ab)^* = (a+b)^*$

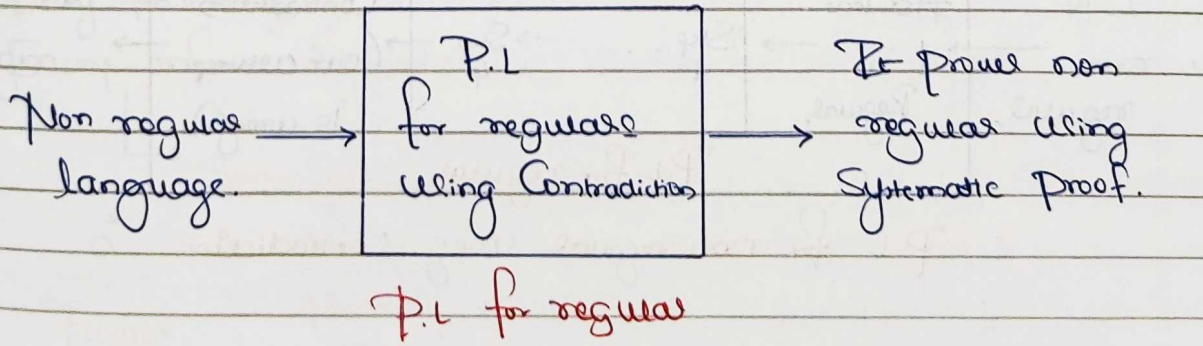
49. $S \to Sa \mid Ab \mid a \quad A \to Ab \quad B \to Ba \mid Bb \mid \varepsilon \quad L = S = a^+$

Pumping Lemma for Regular Languages:

* It is a Lemma (proof).
* It satisfies regular language.
* It can also be used to prove non regulars using contradictio

Note: This pumping lemma can't be used to check unknown language is regular or not.





P.L for regular languages.

Pumping Lemma for Regular Language:

Step 1: Choose any constant P.

Step 2: Select any string $w \in L$ such that $|w| \geq P$.

Step 3: Divide the String w into 3 parts

$w = xyz$ $\quad \underbrace{|y| \geq 1}_{y \neq \varepsilon}$, $|xy| \leq P$

Step 4: $\forall i \geq 0 \; xy^iz \in L$ if L is regular.

Q1. P.L satisfies Regular language.

Q2. P.L proves Non-Regular language using contradiction.

Q3. P.L uses Pigeon hole Principle.

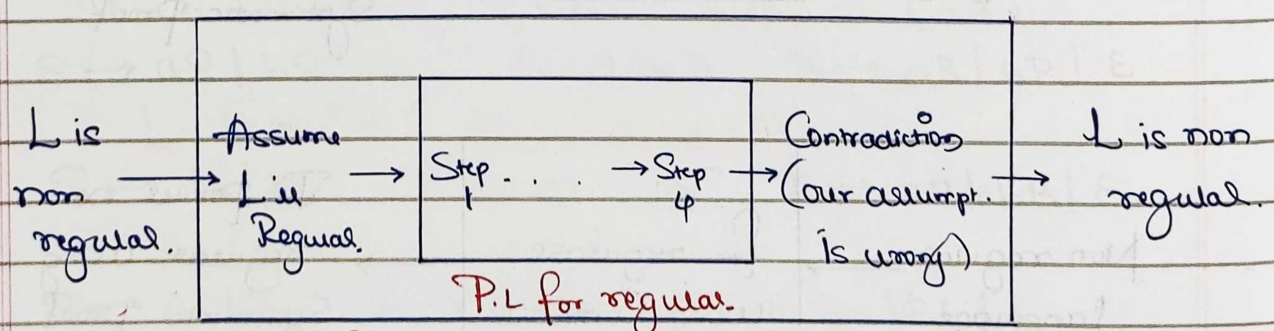
Q4. For $L = (a+b)^* aaa (a+b)^*$, which of the following can't be pumping constant?

Can't be constants:
- ~~A.~~ 1
- ~~B.~~ 2

Can be pumping constants:
- ~~C.~~ 5
- ~~D.~~ 9.

Min length = 3
min States in min DFA = 4
$P \geq 4$

→ How to prove non-regular as non-regular?
↓
P.L with Contradiction.



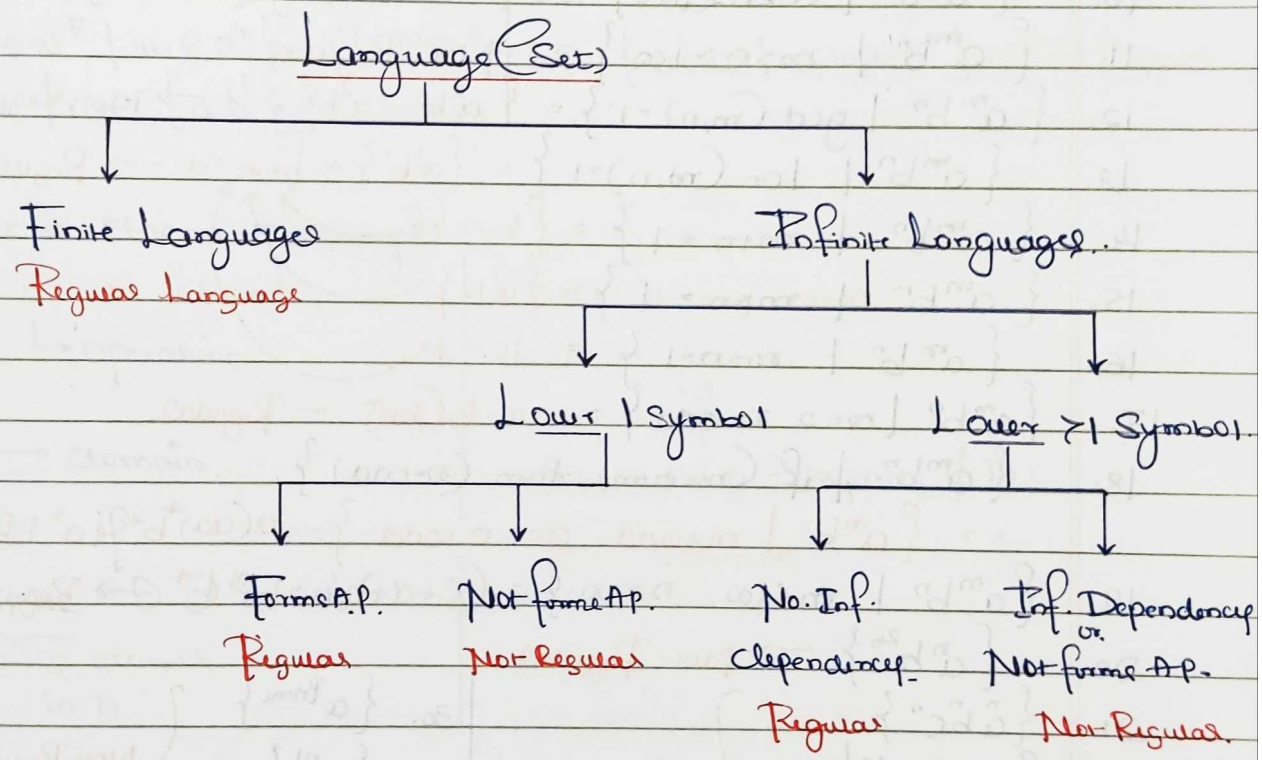
P.L for non regular using contradiction.

classmate
Date 20/5/23
Page

Pumping Lemma for Non-regular using P.L for regular with contradiction.

Step 1: Assume L is regular
Step 2: Choose P
Step 3: Select $w \in L$, $|w| \geq p$.
Step 4: $w = xyz$, $|y| \neq \Sigma$, $|xy| \leq p$
Step 5: $\forall i\ xy^i z \in L$ if L is regular
But if fail, Some i, $xy^i z \notin L$
Step 6: Contradiction. So L is non-regular.

Assumption
Choose p
~~D~~ String w
~~D~~ Divide
Regular
Contradiction.



* Every finite language is Regular [TRUE]
* Every infinite language is Regular [FALSE]
  → Some infinite languages are regular and some are non-regular.

a^* — Regular
a^{Prime} — Not regular.

Identify Regulars and not Regulars:

1. $\{a^m b^n\} = a^* b^*$ → Regular.
2. $\{a^m b^n \mid m > n\}$ → Not Regular language
3. $\{a^m b^n \mid m < n\}$ → Not Regular language.
4. $\{a^m b^n \mid m = even, n = even\} = (aa)^* (bb)^*$
5. $\{a^m b^n \mid m = n = even\} = \{a^{2n} b^{2n}\}$ ⇒ Not Regular.
6. $\{a^n b^n \mid n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb \ldots\}$ → Not Regular.
7. $\{a^n, b^{n+1}\}$
8. $\{a^{n+2}, b^{n+100}\}$ } Not Regular.
9. $\{a^m b^n \mid m = n \text{ or } m \neq n\} = a^* b^*$ → Regular
10. $\{a^m b^n \mid m < n < 100\}$ → Finite language → Regular
11. $\{a^m b^n \mid m > n > 100\}$ → Not Regular.
12. $\{a^m b^n \mid gcd(m,n) = 1\} = \{a^2 b^3, a^2 b^5 \ldots\}$ → Not Regular
13. $\{a^m b^n \mid lcm(m,n) = 1\} = \{a^1 b^1\}$ = finite set → Regular.
14. $\{a^m b^n \mid m + n = 1\} = \{b, a\}$
15. $\{a^m b^n \mid m * n = 1\} = \{a^1 b^1\}$
16. $\{a^m b^n \mid m = n = 1\} = 15. = 13.$
17. $\{a^m b^n \mid m = 2\ n = odd\} = aa\ b(bb)^*$ = Regular
18. $\{a^m b^n \mid \text{if } (m = even) \text{ then } (n = odd)\}$.
    $= \{a^m b^n \mid m = odd \text{ or } n = odd\} = a(aa)^* b^* + a^* b(bb)^*$ → Regular
19. $\{a^m b^n \mid m < 100,\ n > 100\} = (\varepsilon + a + a^2 \ldots)\, b^{101} b^*$ → Regular.
20. $\{a^n b^{2n}\}$ → Not Regular.
21. $\{a^n b^n c^n\}$
22. $\{a^n b^{2n} c^{3n}\}$
23. $\{a^n b^{n+1} c^{n+2}\}$
24. $\{a^n b^n c^*\}$
25. $\{a^n b^* c^n\}$
26. $\{a^n b^{n+5} c^{2n}\}$

} Not Regular.

27. $\{a^n \mid n \geq 0\} = a^*$
28. $\{a^{2n}\} = (aa)^*$
29. $\{a^{3n+100}\}$.

} Regular.

30. $\{a^{Prime}\}$
31. $\{a^{n!}\}$
32. $\{a^{2^n}\}$

} Not Regular.

33. $\{a^{n^2}\}$
34. $\{a^{100^n}\}$
35. $\{a^{n^{45}}\}$

} Not Regular

36. $\{a^{100n}\}$
37. $\{a^{2024n+11}\}$
38. $\{a^{100n}\} \cup \{b^{10k}\}$

} Regular.

39. $\{a^{2n}\}^*$ } Regular
40. $\{a^{\text{Prime}}\}^*$ }
41. $\{a^{n^2}\} = \{a, a^4, \ldots\}$ → Not Regular
42. $\{a^{m^n}\} = \{\varepsilon, a, a^2, \ldots\}$ → Regular.
43. $\{a^{2^n}\}^* = a^*$ } equal. Regular.
44. $\{a^{n^2}\}^* = a^*$ }
45. $\{a^{m^n}\}^* = a^*$ }
46. $\{a^{m^n}\}^+ = a^+$ }

47. $\{a^*.b^{\text{Prime}}\}$ → Not Regular
48. $\{a^{2m}.b^{3m}\}$ → Regular
49. $\{a^{2n}.b^{m^2}\}$ }
50. $\{a^{n!}\ b^{\text{Prime}}\}$ } Not Regular
51. $\{a^{n^2}b^{2^n}\}$ }
52. $\{a^n b^{n^2}\}$ }
53. $\{a^{n^2}.b^{2^n}\}$ } Not Regular.
54. $\{a^n b^{n!}\}$ }

56. $\{ww \mid w \in a^*\} = a^{2n}$ → Regular.
57. $\{w\#w \mid w \in a^*\} = a^n\#a^n$. Not R
58. $\{ww^R \mid w \in a^*\} = 56$. Regular.
59. $\{w\#w^R \mid w \in a^*\} = 57$. Non. R.
60. $\{ww \mid w \in \{ab\}^*\}$ → Non Reg.

61. $\{w\#w \mid w \in \{a.b\}^*\}$ } Not Regular
62. $\{ww^R$ " " $\}$ }
63. $\{w\#w^R$ " " $\}$ }
64. $\{w_1w_2 \mid w_1, w_2 \in a^*\} = a^*$ → Regular.

jaha v dependacy milega wo non regular hoga.

What is closure property?

(D, o) is closed.
- o → operation
- D → domain.

if.

$\forall x_1, x_2 \in D \ni x_1 o x_2 \in D$

(for all 2 elements in D.)

(D, o) is not closed.

if.

$\exists x_1, x_2 \in D \ni x_1 o x_2 \notin D$

(some 2 elements in D.)

For Finite Languages, which of the following are closed?

I) Union → Closed

II) Intersection

Finite, ∪ Finite 2 → Finite

Finite, ∩ Finite 2 → Finite

Domain: Set of finite language.

	Union	Intersection	Compliment
For Finite Language.	closed $F_1 \cup F_2$ = Finite	closed $F_1 \cap F_2$ = Finite.	not closed. $\overline{F_2}$ = Infinite.
For Infinite Language.	closed $Inf_1 \cup Inf_2$ = Inf.	not closed $I_1 \cap I_2$ = either inf or finite.	not closed. $\overline{I}$ = either finite or infinite.

→ Inf.1 ∩ Inf.2 = either finite or infinite
(need not be infinite)

Example 1: $a^*_{inf} \cap a^+_{inf.} = a^+_{inf.}$

Example 2: $a^+ \cap b^+ = \{\varepsilon\}$
Inf. Inf. finite lang
$\{\varepsilon \ldots\}$ $\{\varepsilon \ldots\}$

***

1. Compliment of infinite language is either finite or infinite
2. Compliment of finite language is always infinite.
3. Compliment for infinite language is not closed.
4. Compliment for infinite language is not closed.

Types of Domain:

- → Set of finite language
- → Set of infinite languages
- → " " regular "
- → " " not regular "
- → " " languages.

Operations	Finite languages	Infinite languages
Union	✓	✓
Intersection	✓	✗
Compliment	✗	✗
Difference	✓	✗
Concatenation	✓	✓
Reversal	✓	✓
Kleene Star	✗	✓
Kleene Plus	✗	✓
Subset	✓	✗
Symmetric Difference	✓	✗
$Dem_0(L) = L \cup L^{Rev}$	✗	✓

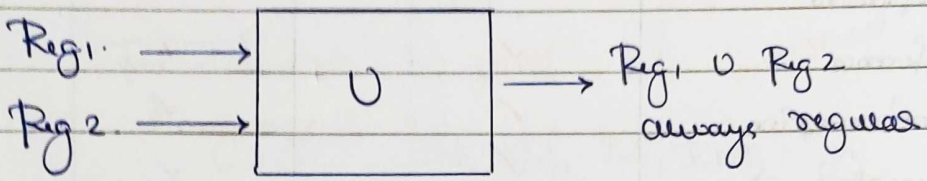
Closure properties for regular languages:

1. Union
2. Intersection
3. Compliment
4. Difference
5. Concatenation
6. Reversal
7. Kleene plus
8. Kleene star.
9. Symmetric diff.
10. ~~Subset.~~
11. Prefix
12. Suffix
13. Substring
14. Substitution
15. Homomorphism.
16. ε-free Homomorphism
17. Inverse Homomorphism.
18. Quotient
19. $\frac{1}{2}(L)$ = half(L)
20. Second half (L)
21. One-third $(L) = \frac{1}{3}(L)$
22. Middle $1/3(L)$
23. Last $\frac{1}{3}(L)$
24. Finite $\cup$
25. Finite $\cap$
26. Finite $-$
27. Finite $\cdot$
28. Finite $\subseteq$
29. Finite f.
30. ~~Infinite $\cup$~~
31. ~~Infinite $\cap$~~
32. ~~Infinite $-$~~
33. ~~Infinite $\cdot$~~
34. ~~Infinite $\subseteq$~~
35. ~~Infinite f~~

① Union for regular language.

↳ Closed.

$Reg_1 \cup Reg_2 \rightarrow$ Always Regular.



Proof:

Reg. Lang ∪ Reg. Lang → Reg Lang.

Reg exp1. Reg exp 2.

Add '+' in between

Reg exp1 + Reg exp2.

"new Reg exp"

eg:

$L_1 = \phi$, $L_2 = \Sigma^+ \rightarrow L_1 \cup L_2 = \Sigma^*$.

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

$L_1 =$ Any, $L_2 = \phi \rightarrow L_1 \cup L_2 = L_1$

$L_1 = \Sigma^*$, $L_2 =$ Any $\rightarrow L_1 \cup L_2 = \Sigma^*$

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

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

Reg ∪ Non Regular → Either Reg or non-regular.

Case 1: $\Sigma^* \cup$ Any non-regular → Regular

Case 2: $\phi \cup$ Any non-regular → Non regular.

Non Regular ∪ Non Regular → Either regular or non regular.

Case 1: $a^n b^n \cup \overline{a^n b^n} \rightarrow (a+b)^*$

Regular.

Case 2: $a^n b^n \cup a^n b^n \rightarrow a^n b^n$

Not regular.

HW

1. Reg. Language ∪ Reg. Language → Regular
2. Regular language ∪ Non Regular Language → either reg / non regular
3. Non regular language ∪ Non regular Language → either reg / non regular
4. If $L_1 \cup L_2$ is Regular then L_1 is Either reg or non regular.
   Reg ∪ L_2 → Regular possible
   Non regular ∪ L_2 → Regular possible.
5. If $L_1 \cup L_2$ is Not Regular than L_1 is Either regular / not regular.
   Reg ∪ L_2 → Not regular possible
   Not Regular ∪ L_2 → Not regular possible

② Intersection for regular language
↳ closed.

$Reg_1 \cap Reg_2$ → Always Regular.

(i.) $\phi \cap L \rightarrow \{\} \cap L \rightarrow \{\} \rightarrow \phi$

(ii.) $\Sigma^* \cap L \rightarrow$ Set of all Strings $\cap L \rightarrow L$

(iii.) $a^* \cap b^* \rightarrow \{\varepsilon, a, a^2 \ldots\} \cap \{\varepsilon, b, b^2 \ldots\} \rightarrow \{\varepsilon\}$

(iv.) $a^+ \cap b^+ \rightarrow \phi$

(v.) $a^+ b^+ \cap a^+ \rightarrow a^+$.

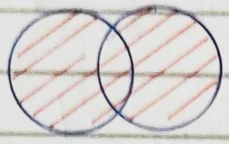
I. If L_1 is Regular and L_2 is Regular then $L_1 \cap L_2$ is Regular

II. If L_1 is Regular and L_2 is not regular then $L_1 \cap L_2$ is Either regular or non regular.

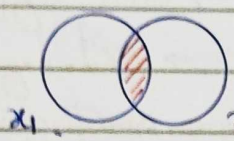
III. If L_1 is not regular and L_2 is not regular then $L_1 \cap L_2$ is Either regular or not regular.

IV. If $L_1 \cap L_2$ is regular then L_1 is either regular or not regular

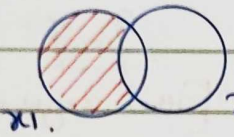
V. If $L_1 \cap L_2$ is not regular then L_2 is either reg or non regular.



$$x_1 \cup x_2 = \{ x \mid x \in x_1 \text{ or } x \in x_2 \}$$



$$x_1 \cap x_2 = \{ x \mid x \in x_1 \text{ and } x \in x_2 \}$$



$$x_1 - x_2 = \{ x \mid x \in x_1 \text{ and } x \in x_2 \}$$

x is only in x_1.

If $x_1 \cap x_2 = \phi$ then x_1 and x_2 are Disjoint Sets.

(3) Compliment for Regular Language

↳ Closed.

$\overline{\text{Regular}}$ → Always Regular

I. L is Regular iff $\overline{L}$ is Regular

II. L is not regular if $\overline{L}$ is not regular.

III. Not regular → Not regular

Proof:

L is regular.

↓↑

Design DFA

↓↑

Modified DFA

↓↑

$\overline{L}$ is regular.

i. $L = \phi \rightarrow \overline{L} = \Sigma^*$

ii. $L = \Sigma^* \rightarrow \overline{L} = \phi$

iii. $L = a\Sigma^* \rightarrow \overline{L} = \varepsilon + b\Sigma^*$

iv. $L = b\Sigma^* \rightarrow \overline{L} = \varepsilon + a\Sigma^*$

v. $L = \Sigma^* a \rightarrow \varepsilon + \Sigma^* b$

vi. $L = \Sigma^* a \Sigma^* \rightarrow \overline{L} = b^*$

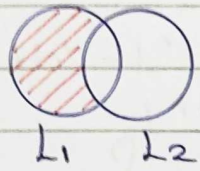
vii. $L = \Sigma^* b \Sigma^* \rightarrow \overline{L} = a^* = \varepsilon + a^+$.

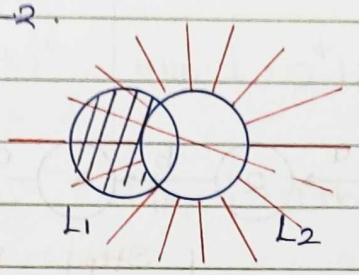
④ Difference for regular language.

↳ Closed.

Reg$_1$ − Reg$_2$ → Always Regular.

Def I $L_1 - L_2 = L_1 \cap \overline{L_2}$.





$a^* - b^* \rightarrow a^+$

Def II. $L_1 - L_2 = L_1 - (L_1 \cap L_2)$

i. $L_1 = \phi$, L_2 = Any → $L_1 - L_2 = \phi$, $L_2 - L_1 = L_2$.

ii. $L_1 = \Sigma^*$, L_2 = Any → $L_1 - L_2 = \Sigma^* - L_2 = \overline{L_2}$

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

iii. $L_1 = a^*$, $L_2 = b^*$ → $L_1 - L_2 = a^* - b^* \rightarrow a^+$

$L_2 - L_1 = b^* - a^* \rightarrow b^+$

iv. $L_1 = a^*$, $L_2 = (aa)^*$ → $L_1 - L_2 = a(aa)^*$

$L_2 - L_1 = \phi$.

⑤ Concatenation for regular languages.

↳ Closed

$L_1 . L_2$ → Always Regular.

reg. reg.

(i) $L_1 = \phi$, L_2 = Any → $L_1 . L_2 = \phi$, $L_2 . L_1 = \phi$.

(ii) $L_1 = \Sigma^*$, $L_2 = \{a\}$ → $L_1 . L_2 = \Sigma^* a$, $L_2 . L_1 = a\Sigma^*$.

(iii) $L_1 = a^*$, $L_2 = b^*$ → $L_1 . L_2 = a^* b^*$ $L_2 . L_1 = b^* . a^*$.

(iv) $L_1 = a^*$, $L_2 = (aa)^*$ → $L_1 . L_2 = L_1$ $L_2 . L_1 = L_1$.

(v) $L_1 = \Sigma^*$, $L_2 = a\Sigma^*$ → $L_1 . L_2 = \Sigma^* a \Sigma^*$ $L_2 . L_1 = L_2$.

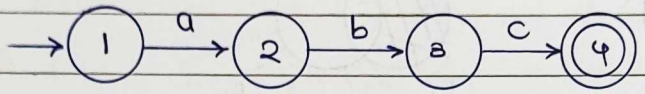
⑥ Regular Language Reversal / Reversal for regular language

↳ closed.

If L is Regular, L^{Rev} = Regular.

$(Reg)^{Reverse}$ = Regular

eg:: L = {abc}.

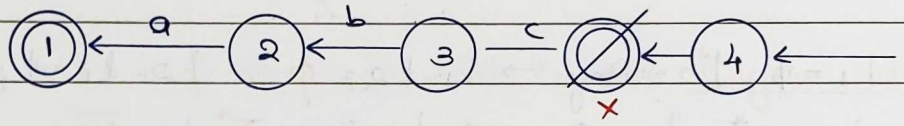


L^{Rev} = {cba}

Step 1: make 1 as final

Step 2: Initial ↔ Final

Step 3: Reverse all transitions.



i. $L = \phi \rightarrow L^{Rev} = \phi = L$

ii. $L = \Sigma^* \rightarrow L^{Rev} = \Sigma^* = L$

iii. $L = a\Sigma^* \rightarrow L^{Rev} = \Sigma^* a$

iv. $L = \Sigma^* b \rightarrow L^{Rev} = b\Sigma^*$

v. $L = ab^* \rightarrow L^{Rev} = b^* a$.

vi. $L = \Sigma^* a \Sigma^* \rightarrow L^{Rev} = \Sigma^* a \Sigma^*$

vii. $L = a^* b^* \rightarrow L^{Rev} = b^* a^*$.

viii. $(L^{Rev})^{Rev} = L$

ix. $(\overline{L}) = L$

I. Reversal of regular language is Regular

II. Reverse of non-regular language is non regular.

Note: * L is Regular if L^{Rev} is regular

* L is Regular if L^{Rev} is regular.

⑦ Kleene Star (Kleene closure)

If L is Reg. L^* is Regular

Note: * If L^* is regular then L is either reg / non regular.

* If L^+ is regular then L is either reg / non regular.

⑧ Kleene Plus (positive closure)

If L is Reg. L^+ is Regular.

{ $\{a^{Prime}\}^+$ is Regular but $\{a^{Prime}\}$ is not regular.

i. $L = \phi \rightarrow L^* = \{\varepsilon\}$
$L^+ = \phi = L$

ii. $L = \Sigma^+ \rightarrow L^* = \Sigma^* = L$
$L^+ = \Sigma^+ = L$

iii. $L = \{a, b\} \Rightarrow L^* = (a+b)^*$
$L^+ = (a+b)^+$

iv. $L = a^* + b^+ \rightarrow L^* = (a+b)^*$
$L^+ = (a+b)^+$

v. $L = a^* b^* \rightarrow L^* = (a+b)^* \cdot L^+ = (a+b)^*$

vi. $L = b^* a^* \rightarrow L^* = (a+b)^*$
$L^+ = (a+b)^*$.

vii. $L = a^+ + b^+ \rightarrow L^* = (a+b)^*$
$L^+ = (a+b)^+$.

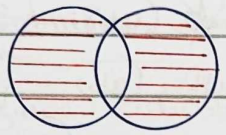
viii. $L = a^+ b^+ \rightarrow L^* = (a^+ b^+)^*$ $\neq (a+b)^+$.
$L^+ = (a^+ b^+)^+$

ix. $L = a + b^* \rightarrow L^* = (a+b)^*$ $L^+ = (a+b^*)^+$.

(9.) Symmetric Difference for regulars.

↳ closed.

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



eg:: 1. $L_1 = a^+$, $L_2 = b^*$ $\Rightarrow L_1 \Delta L_2 = a^+ + b^+$

2. $L_1 = a\Sigma^*$, $L_2 = b\Sigma^*$ $\Rightarrow L_1 \Delta L_2 = a\Sigma^* + b\Sigma^*$

(10.) Subset for Regulars.

↳ Not closed.

Every subset of a regular language is Either regular or not regular.

$L = a^*$.

ϕ $\{\varepsilon\}$ a^* a^+ a^{100} a^{Prime}.

Regular | Non regular.

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

ϕ a^* b^+ $a^n b^n$ a^{Prime}.

Regular | Non regular.

I. Every subset of finite language is Finite Language (Regular)

II. Every subset of infinite language is either infinite or not infinite.

III. Every subset of regular language is either regular or not regular.

IV. Every subset of non regular language is either regular or non regular

For Language.

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

$\text{Suffix}(L) = \{v \mid uv \in L\}$

$\text{SubString}(L) = \{y \mid xyz \in L\}$

For String:

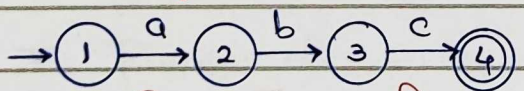
$\text{Prefix}(\omega) = \{u \mid uv = \omega\}$ $\quad \text{prefix}(ab) = \{\varepsilon, a, ab\}$

$\text{Suffix}(\omega) = \{v \mid uv = \omega\}$ $\quad \text{Suffix}(ab) = \{\varepsilon, b, ab\}$

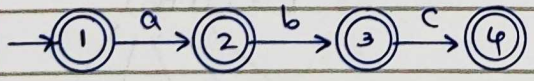
$\text{Substring}(\omega) = \{y \mid xyz = \omega\}$ $\quad \text{Substring}(ab) = \{\varepsilon, a, b, ab\}$

(11) Prefix (Reg Language) is Regular

$L = \{abc\}$



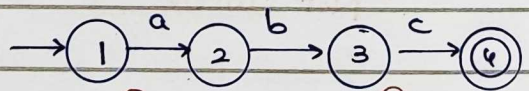
from initial to final state path, make every state final.



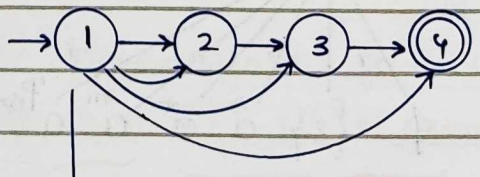
$\text{prefix}(L) = \{\varepsilon, a, ab, abc\}$.

(12) Suffix (Regular) is Regular.

$L = \{abc\}$



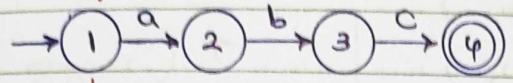
from initial to final state path, add ε move, from initial to every state.



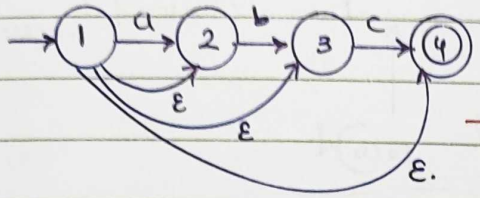
$\text{Suffix}(L) = \{\varepsilon, c, bc, abc\}$.

(13) Substring (Reg) is Regular.

$L = \{abc\}$.



↓ Combine both prefix algorithm and suffix algorithm.



→ Substring (L) = $\{\varepsilon, a, b, c, ab, ba, cabc\}$

(14) Substitution for Regular Language.

↳ closed.

Substitution of a regular language is Regular Language.

(Regular Substitution).

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

$L = a^*ba$ (Reg) → f → f(L) is Regular

$= 1^*(01(11)^*)^*1^*$

f(a) = Reg. Language.

$= 1^*$

f(b) = Reg. Language.

$01(11)^*$

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

1. $L = (ab)^*$ and $f(a) = \{\varepsilon\}$, $f(b) = 0^*$.

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

$= (\varepsilon \cdot 0^*)^*$

$= 0^*$.

2. $L = a(a+b)^*$; $f(a) = 0^*$, $f(b) = 1^*$.

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

$= f(a) \cdot (f(a) + f(b))^*$

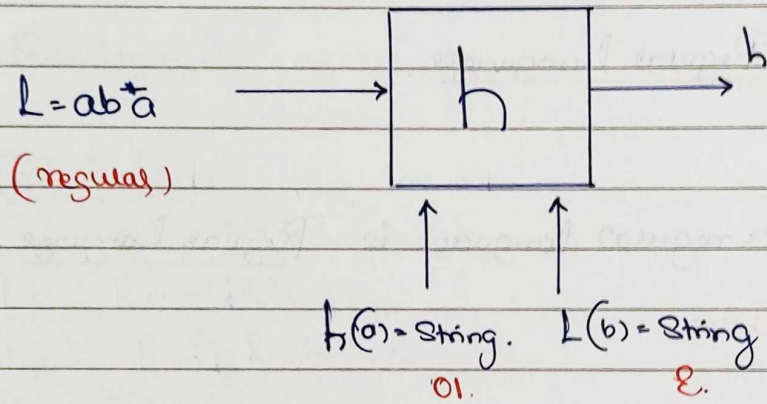
$= 0^*(0^* + 1^*)^*$

$= (0+1)^*$

Σ { may or may not be same.
Δ {

(15.) Homomorphism

[String Substitution]



h(L): In given reg L over Σ, every symbol is substituted with some string over Δ.

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

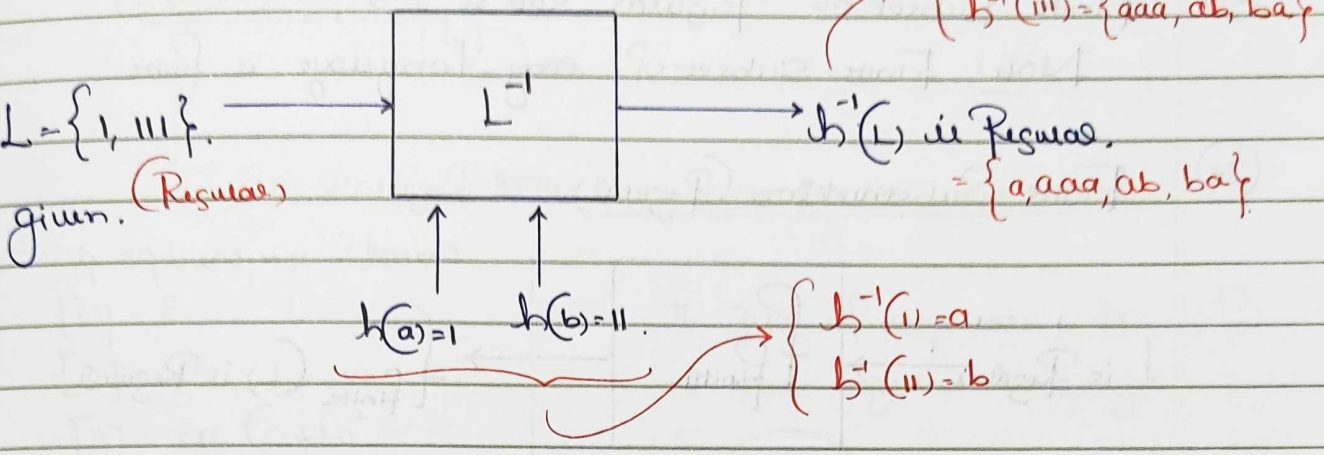
Set of symbols Set of strings.

(16) ε-free Homomorphism.

$h : \Sigma \longrightarrow \Delta^+$

In given reg L over Σ, every symbol is substituted with some non empty string (other than ε) over Δ.

(17) Inverse Homomorphism for Regular Language
↳ closed.



(18) Quotient (/)

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

eg::
1. $abc/\varepsilon = abc$
2. $abc/a = x$
3. $abc/c = ab$
4. $abc/ab = x$
5. $abc/bc = a$
6. $abc/abc = \varepsilon$.
7. $\varepsilon/abc = x$
8. $\varepsilon/\varepsilon = \varepsilon$
9. $L/\varepsilon = L$
10. $a/a = \varepsilon$.

$L = \{\varepsilon, a, ab, abc, abcd, abcda, abbbcd\}$.

eg::
(19) $\frac{1}{2}(L) = \{x \mid xy \in L, |x| = |y|\} = \{\varepsilon, a, ab, abb\}$
(20) Second $1/2(L) = \{y \mid xy \in L, |x| = |y|\} = \{\varepsilon, b, cd, bcd\}$
(21) $1/3(L) = \{x \mid xyz \in L, |x| = |y| = |z|\} = \{\varepsilon, a, ab\}$
(22) Modulo $1/3(L) = \{y \mid$ " " " $\} = \{\varepsilon, b, bb\}$
(23) Last $1/3(L) = \{z \mid$ " " " $\} = \{\varepsilon, c, cd\}$.

(24) Finite union is closed for regular languages.
$L_1 \cup L_2 \cup L_3 \cdots \cup L_k \rightarrow$ Regular.

(25) Finite intersection is closed for regulars.
$L_1 \cap L_2 \cap L_3 \cdots \cap L_k \rightarrow$ Regular.

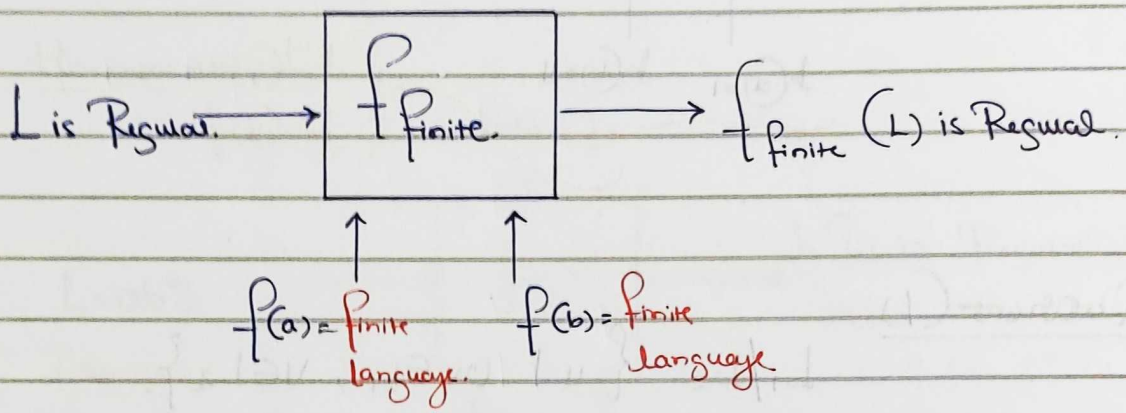
(26) $L_1 - L_2 - L_3 \ldots - L_k \rightarrow$ Regular

(27) $L_1 * L_2 * L_3 \ldots * L_k \rightarrow$ Regular

(28) Finite Subset of Regular set is finite set (Regular)

Note: Finite subset of any language is finite

(29) Finite Substitution (Regular) → Regular.



(30) $R_1 \cup R_2 \cup R_3 \ldots \rightarrow$ either regular or non regular

(31) $R_1 \cap R_2 \cap R_3 \ldots \rightarrow$ "

(32) $R_1 - R_2 - R_3 \ldots \rightarrow$ "

(33) $R_1 . R_2 . R_3 \ldots \rightarrow$ "

(34) Infinite subset of regular language → either regular or non regular

(35) Infinite ~~subset~~ Substitution of regular language → "

Equivalence Classes [Myhill – Nerode equivalence classes]

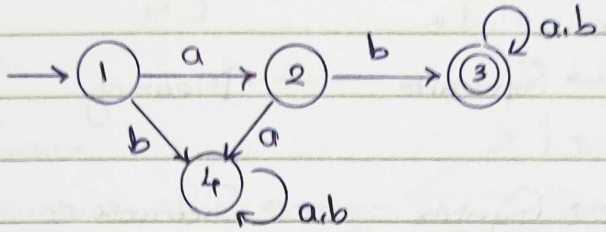
I. No. of states in min DFA for Reg = no. of equivalence classes for regular.

→ Find no. of equivalence class for $L = ab(a \cup b)^*$

Regular Language.

→ Step 1: Construct min DFA

PUSH DOWN AUTOMATA



↳ Step 2: Each state represents one equivalence class.

= 4 equivalence classes

[1] = ε.

[2] = a

[3] = ab (a+b)*

[4] = (b+aa)(a+b)*.

II. No. of equivalence classes for non-regular languages.

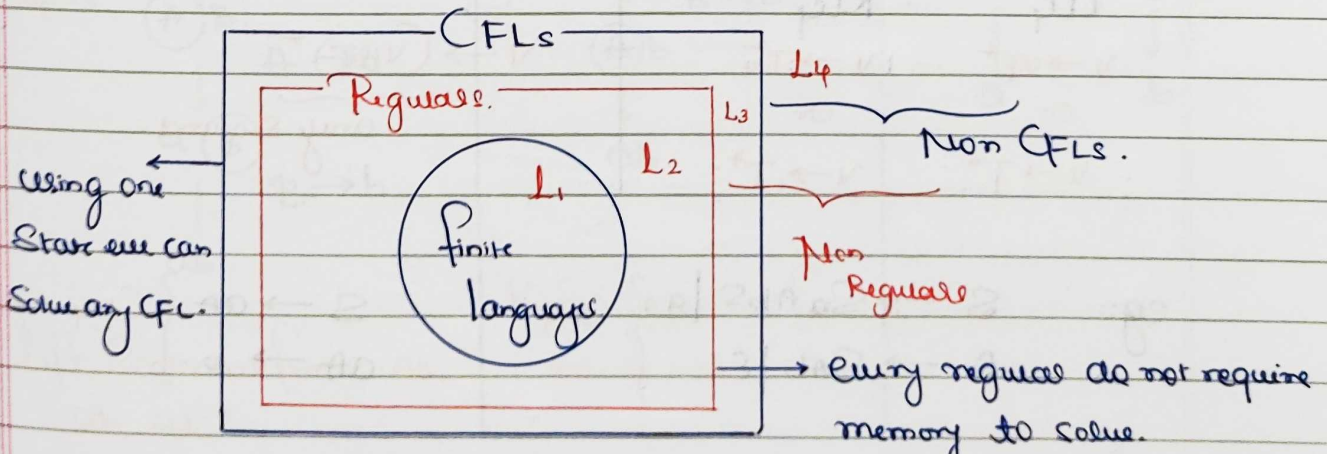
Regular.

a^*b^* → finite no. of equivalent classes.

a^nb^n → Infinite no. of equivalent classes.

Not Regular.

Context Free Languages



I. L_1 is finite language (Reg, CFL)

II. L_2 is regular but not finite (Infinite & Regular)

III. L_3 is CFL but not regular. (CFL and not regular)

IV. L_4 is not CFL. (not regular and not finite).

Date 01/6/23

Regular Language. CFL CSL

Words → Structure → Meaning

Tokens → Syntax → Semantic

Words → Sentences → English language meaning

CFL Applications:

1. Every regular application
2. Messages
3. Call history
4. Web page history
5. DFS
6. Balanced parenthesis

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

RG

- LLG: $V \to VT^*$ or $V \to T^*$
- RLG: $V \to TV^*$ or $V \to T^*$

CFG

It represents CFL.

$V \to (V \cup T)^*$ — any sequence.

eg:: $S \to SaAbS \mid a$, $A \to SAb \mid \varepsilon$ } CFG ✓

$S \to aA$, $aA \to b$ } CFG ×

Note:
(i) Every RG is CFG
(ii) CFG need not be RG
(iii) Every Regular Language is CFL
(iv) CFL is may or may not be regular.

CFG.

- Definition
- Derivation of a string. → LMD (Left most derivation)
  → RMD (Right most derivation)
  → Parse tree (Derivation tree)
- Type of CFG. → Ambiguous cfg
  → Unambiguous cfg.
- CFG vs CFL
- Inherently ambiguous CFL.

eg::

CFG

S → AB | ....
A → a | ...
B → b | ...

String ab.

Derivation:

① LMD.

S

No. of Steps = 3.

A B. A → a

a B. B → b.

ab

Left Sentential forms: AB, aB, ab.

② RMD.

S

No. of Steps = 3.

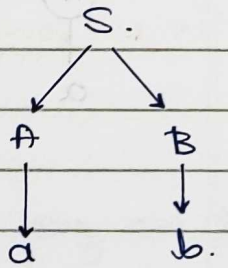
A B. B → b.

A b

ab.

Right sequential forms: AB, Ab, ab.

③ Parse Tree.



LMD: To derive a string, in every sequential form "left most non terminal" is substituted.

aAbB. Q1. left most symbol = 'a'. Q2. left most terminal = 'a'

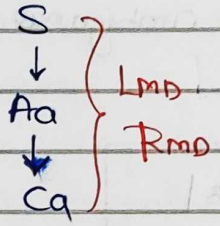
Q3. left most non terminal = 'A'.

RMD: To drive a string, in every sequential form "right most non-terminal" is substituted.

Note: In general, LMD and RMD need not be same.

eg: $S \to Aa \mid AB$.
$A \to c$
$B \to d$.

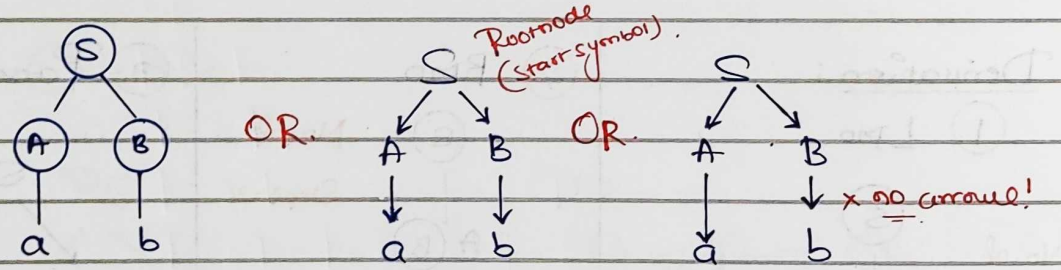
$w = ca$
LMD and RMD are same.



$w = cd$
LMD, RMD are different.

LMD	RMD
S	S
AB	AB
cB	Ad
cd	ca

Parse Tree (Derivation Tree).



collect all leaf nodes = ab

Leaf node = Terminal or ε.
Non leaf node (Intermediate node) = Variable (non-terminal).

For a given string: (If only one derivation exist)

No. of derivation steps = No. of steps in LMD
= No. of steps in RMD
= No. of non leaf nodes in parse tree.

No. of LMDs
No of RMDs
No of parse trees.

No. of derivations → How many ways String can be derived.

≠

No. of derivative Steps : for a particular derivation, how many steps?

→ It is possible to answer if we have only one derivation.

→ No of Substitutions in RMD.
" " in LMD
" " non leaf node " Parse tree.

Types of CFGs:

[I.] Ambiguous CFG
→ Some String (atleast one) has more than 1 derivation. (you have to find one string which has >1 Parse trees).

(not ambiguous).

[II.] Unambiguous CFG.
→ every String derived from CFG has only one derivation.

CFG vs CFL.

1. $S \to AB$, $A \to \varepsilon$, $B \to b$
$L = \{b\}$.

2. $S \to SS \mid \varepsilon$.
$L = \{\varepsilon\}$

3. $S \to Sa \mid \varepsilon$.
$L = a^*$.

4. $S \to aS \mid \varepsilon$.
$L = a^*$.

5. $S \to Sa \mid a$
$L = a^+$

6. $S \to aS \mid a$
$L = a^+$

7. $S \to aS \mid bS \mid \varepsilon$.
$L = (a+b)^*$

8. $S \to aS \mid bS \mid a \mid b$
$L = (a+b)^+$

9. $S \to Sa \mid Sb \mid \varepsilon$.
$L = (a+b)^*$

10. $S \to Sa \mid Sb \mid a \mid b$
$L = (a+b)^+$.

11. $S \to aaS \mid \varepsilon$
$L = (aa)^*$

12. $S \to aS \mid bS \mid cS \mid \varepsilon$
$L = (a+b+c)^*$

13. $S \to abS \mid aaS \mid \varepsilon$
$L = (ab+aa)^*$

14. $S \to aS \mid bS \mid b$.
$L = (a+b)^* b$.

15. $S \to AB$
$A \to aA \mid \varepsilon$ ($A = a^*$)
$B \to Bb \mid \varepsilon$ ($B = b^*$)

It is not regular grammar but it generates a regular language.

$L = a^* b^*$.

16. $S \to aS \mid Sb \mid \varepsilon$.
$\to a^* S b^* = a^* b^*$.

17. $S \to bS \mid Sa \mid \varepsilon \to b^* a^*$.

18. $S \to aS \mid Sb \mid a \to L = a^* a b^* = a^+ b^*$.

19. $S \to aS \mid Sb \mid b \to L = a^* b b^* = a^* b^+$.

20. $S \to aS \mid Sb \mid a \mid b \to L = a^* (a+b) b^* = a^+ b^* + a^* b^+$

21. $S \to aS \mid Sb \mid a \mid b \to L = a^* (a+b) b^* = a^+ b^* + a^* b^+$

21. $S \to aS \mid Sb \mid ab \to L = a^* ab b^* = a^+ b^+$.

22. $S \to aSb \mid \varepsilon \to L = a^n S b^n = a^n b^n$.
$= \{a^n b^n \mid n \geq 0\}$. $= \{a^* b^* \mid \#a_i = \#b_i\}$.

23. $S \to bSa \mid \varepsilon$. $L = \{b^n a^n \mid n \geq 0\}$.

24. $S \to aSb \mid a$ $L = \{a^n a b^n \mid n \geq 0\} = \{a^{n+1} b^n\}$.

25. $S \to aSb \mid b$. $L = \{a^n b^{n+1} \mid n \geq 0\}$

26. $S \to aaSbb \mid \varepsilon$. $L = \{a^{2n} b^{2n}\}$.

27. $S \to aaaSbbb \mid \varepsilon$. $L = \{a^{3n} b^{3n}\}$.

28. $S \to aSbb \mid \varepsilon$ $L = \{a^n b^{2n} \mid n \geq 0\}$.

29. $S \to aaSb \mid \varepsilon$. $L = \{a^{2n} b^n\} = \{a^m b^n \mid m = 2n\}$.

30. $S \to aaSbbb \mid \varepsilon$ $L = \{a^{2n} b^{3n} \mid n \geq 0\}$
$= \{a^i b^j \mid i = 2n, j = 3n\}$

31. $S \to aSb \mid ab$

$L = \{ab, a^2b^2, a^3b^3 \ldots\} \neq a^+b^+$

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

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

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

$= \{a^{n-1}\, b^{n-1} \mid n \geq 2\}$.

Note: $L_1 = \{a^n \mid n \geq 0\} = a^*$

$L_2 = \{b^n \mid n \geq 0\} = b^*$.

$L_1 \cdot L_2 = \{a^n\} \cdot \{b^n\}$

$= \{\varepsilon, a^1, a^2 \ldots\} \cdot \{\varepsilon, b, b^2 \ldots\}$

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

$= \{a^{n_1} b^{n_2} \mid n_1, n_2 \geq 0\}$.

32. $S \to aSb \mid A$

$A \to aA \mid \varepsilon$.

$L = \{a^n a^+ b^n\} = \{a^i b^j \mid i \geq j\}$.

33. $S \to aSb \mid A$

$A \to bA \mid \varepsilon \longrightarrow b^*$

$L = \{a^n b^{m+n}\} = \{a^i b^j \mid i \leq j\}$.

34. $S \to aSb \mid A$

$A \to bB \mid \varepsilon$

$S \to aSb \mid \varepsilon$

$L = a^n b^n$.

35. $S \to SA \mid \varepsilon$

$A \to a$

$L = A^* = a^*$.

36. $S \to SA \mid \varepsilon$.

$A \to aa$. $L = A^* = (aa)^*$

37. $S \to SA \mid \varepsilon$

$A \to a \mid b$ $L = A^* = (a+b)^*$.

38. $S \to SA \mid \varepsilon$.

$A \to aAb \mid \varepsilon$.

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

39. $S \to SA \mid \varepsilon$

$A \to bAa \mid \varepsilon$.

$L = \{b^n a^n\}^*$.

40. $S \to SA \mid \varepsilon$

$A \to ab$. $L = (ab)^*$

41. $S \to aSa \mid bSb \mid \varepsilon \to L = \{ww^R \mid w \in \{a,b\}^*\}$

= Set of all even length palindromes

$= \{x \mid x \in \{a,b\}^*, x = x^R, |x| = \text{even}\}$.

42. $S \to aSa \mid bSb \mid a \mid b \to L$ = length of all odd length palindromes

43. $S \to aSa \mid bSb \mid a \mid b \mid \varepsilon \to L$ = set of all palindromes

$= \{x \mid x = x^R \; x \in \{a,b\}^*\}$

$= \{wxw^R \mid w \in \{a,b\}^*, x \in \{a,b,\varepsilon\}\}$

44. $S \to aSa \mid bSb \mid \#$

$L = \{w\#w^R \mid w \in \{a,b\}^*\}$.

45. $S \to aSb \mid A$

$A \to aAc \mid \varepsilon \to L = a^k a^n c^n b^k = \{a^{k+n} c^n b^k\}$.

46. $S \to AB$

$A \to aAb \mid \varepsilon \qquad B = c^n d^n \qquad L = \{a^k b^k c^n d^n\}$.

$B \to cBd \mid \varepsilon. \qquad A = a^k b^k$

47. $S \to aSb \mid A$ $\Big\}$ $A = c^n b^n$

$A \to cAb \mid \varepsilon$ $\Big\}$ $L = a^k c^n b^n b^k = \{a^k c^n b^{n+k}\}$.

48. $S \to ABC$

$A \to aA \mid \varepsilon \quad -a^*$

$B \to bB \mid \varepsilon \quad -b^*$

$C \to cC \mid \varepsilon. \quad -c^*$

$\Big\}$ $L = a^* b^* c^*$.

49. $S \to SS \mid (S) \mid \varepsilon$.

L = Set of all balanced parenthesis

50. $E \to E+E \mid E*E \mid (E) \mid a$

L = Set of all arithematic expressions using $+, *$

$\Big\}$ functionality (meaning)

51. $S \to aSb \mid bSa \mid SS \mid \varepsilon$

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

52. $S \to aSb \mid bSa \mid SS \mid aS \mid \varepsilon$

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

53. $S \to aSb \mid bSa \mid SS \mid bS \mid \varepsilon$.

$L = \{w \mid w \in \{a,b\}^*, n_a(w) \leq n_b(w)\}$.

54. $S \to aSb \mid bSa \mid SS \mid aS \mid a$

$n_a(w) > n_b(w)$

ss. $S \rightarrow SaSbS \mid SbSaS \mid \varepsilon$.

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

→ Every Regular Language is CFL.

→ Some non regular languages are CFLs.

$\{a^n b^n\}$ ← non reg, CFL.

$\{ww^R \mid w \in \{a,b\}^*\}$

$\{w \# w^R \mid \text{"} \text{"}\}$

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

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

Push Down Automata.

FA

DFA ≅ NFA

$Q \times \Sigma \rightarrow Q$ DFA.

$Q \times \Sigma_\varepsilon \rightarrow 2^Q$ NFA.

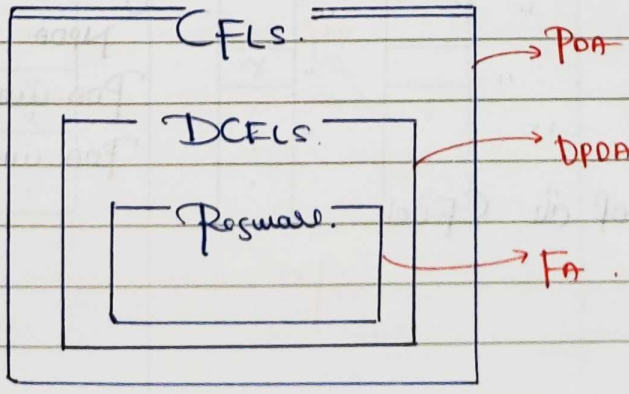
PDA

DPDA < NPDA

(PDA)

DPDA	PDA
≅	≅
DCFL.	CFL

- Every DCFL is CFL
- CFL need not be DCFL
- Every DPDA is PDA
- PDA need not be DPDA.



FA < DPDA < PDA.

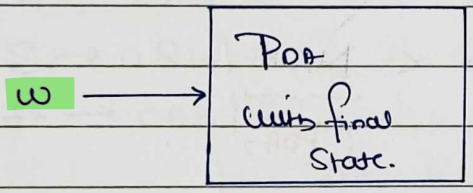
→ Which of the following statements is "True?"

1. Regular Language is subset of CFL. [False]
2. Set of all regulars is subset of set of all CFLs. [True]

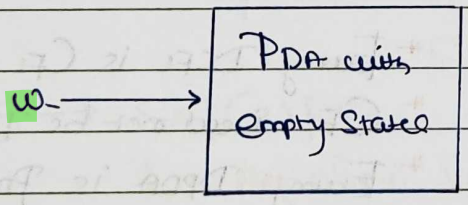
Push Down Automata.
- Push Down → Stack
- Automata → Computation

PDA.
- PDA with Final State mechanism.
- PDA with Empty Stack mechanism.

PDA (NPDA)
- → by default non-deterministic m/c
- → it accepts CFL.



If w∈L, atleast one path exist that halts at final state.



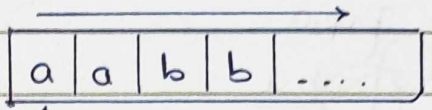
If w∈L, atleast one path exist that makes empty stack

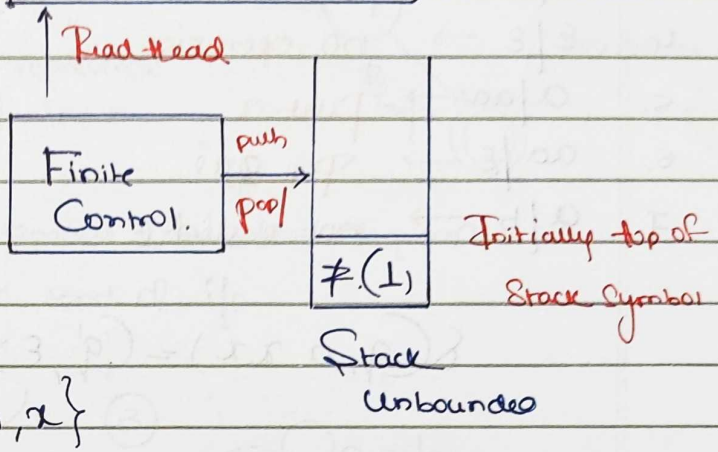
→ L_1 = Set of all languages represented by PDA NPDA
→ L_2 = " " " PDA with final State
→ L_3 = " " " PDA with Empty State
→ L_4 = " " "
→ L_5 = Set of all CFLs.

$L_1 = L_2 = L_3 = L_4 = L_5$

all are same.

Push Down Automata.



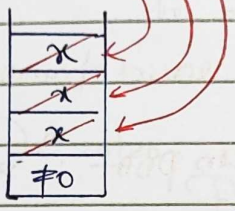


$a^n b^n$. ← input alphabet.

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

aaabbb

push



$\Gamma = \{\neq 0, x\}$

Stack Alphabet.

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

$\delta_{NFA} : Q \times \Sigma_\epsilon \rightarrow 2^Q$

$\delta_{DFA} : Q \times \Sigma \rightarrow$

$PDA = FA + \text{Stack}$

$PDA = (Q, \Sigma, \delta, q_0, F, Z_0, \Gamma)$

$(\perp)$

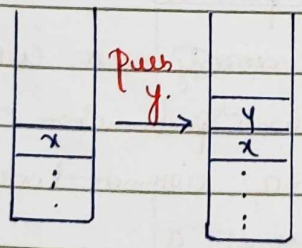
→ Stack alphabet (set of Stack Symbols)

→ initial top of stack Symbol (bottom of Stack Symbol).

$\delta_{PDA} = Q \times \Sigma_\epsilon \times \Gamma_\epsilon^* \rightarrow 2^{Q \times \Gamma^*}$

$\delta_{DPDA} = Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$.

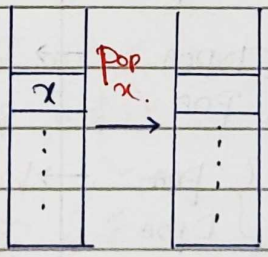
Push.



$x \rightarrow yx$

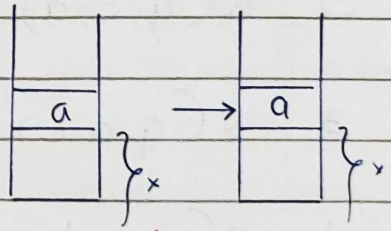
x/yx

Pop.



x/ϵ.

No Operation



a/a

ϵ/ϵ.

1. $a/a \rightarrow$ no operation performed on stack
2. $a/\varepsilon \rightarrow$ pop a
3. $\varepsilon/a \rightarrow$ push a
4. $\varepsilon/\varepsilon \rightarrow$ no operation " "
5. $a/aa \rightarrow$ push a
6. $aa/\varepsilon \rightarrow$ pop 2 a's
7. $a/b \rightarrow$ not valid

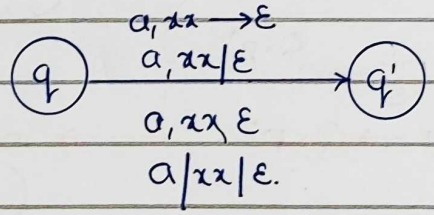
$\delta(q, a, x) = (q', \varepsilon)$ → not valid transition in DPDA
→ valid in PDA.

DPDA = $Q \times \Sigma \times \Gamma \rightarrow$
PDA = $Q \times \Sigma_\varepsilon \times \Gamma^* \rightarrow$

Pop 2 x's.

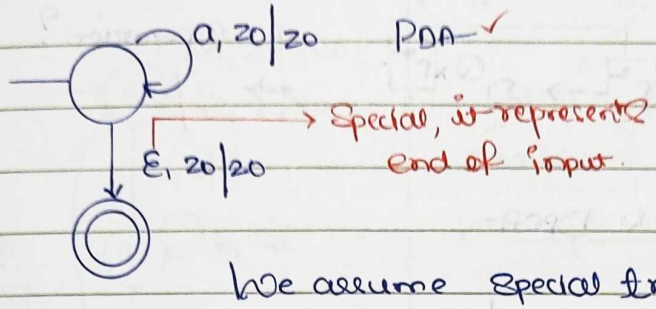
$\delta(q, a, xx) = (q', \varepsilon)$

present state. | current input symbol. | present top of stack | next state

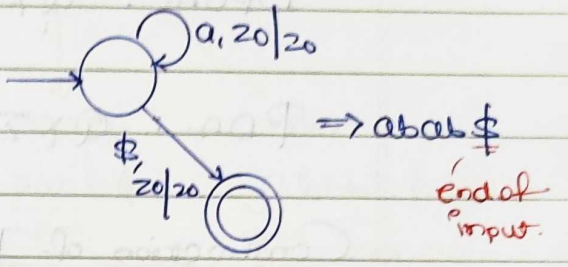


1. $\delta(q, \varepsilon, \varepsilon) = (q', a)$ — PDA ✓, DPDA ✗ → Without reading input, without looking at tos, push 'a'.
2. $\delta(q, \varepsilon, a) = (q', a)$ — DPDA ✗, PDA ✓ → without reading input, when tos=a, no operations performed on stack
3. $\delta(q, a, \varepsilon) = (q', a)$ — PDA ✓, DPDA ✗ → when i/p=a, without looking at stack push 'a'
4. $\delta(q, a, b) = (q', ab)$ — PDA ✓, DPDA ✓ → when i/p=a, tos=b, push a
5. $\delta(q, a, b) = (q', aaab)$ — PDA ✓, DPDA ✓ → when i/p=a, tos=b, push 3 a's

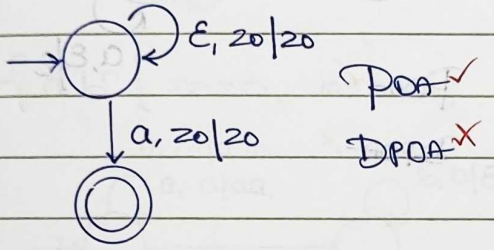
Identify DPDA or PDA

1. 

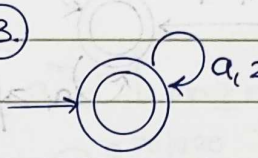
DPDA ✓
PDA ✓



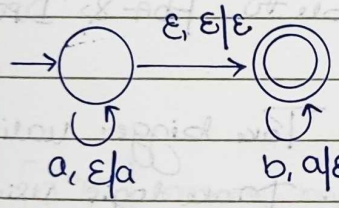
We assume special transitions required to represent end of i/p.

2. 

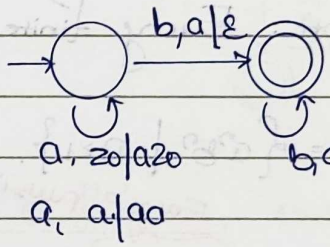
PDA ✓
DPDA ✗

3. 

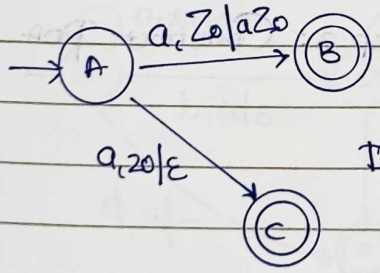
PDA ✓
DPDA ✓

4. 

PDA ✓
DPDA ✗

5. 

PDA ✓
DPDA ✓

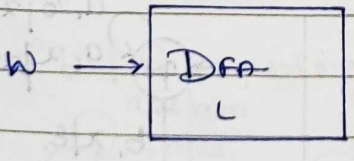
6. 

PDA ✓
DPDA ✗

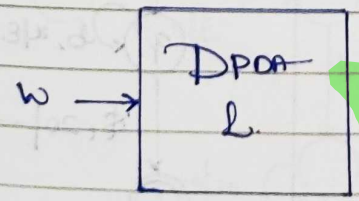
$Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$ — Fails

$Q \times \Sigma_\varepsilon \times \Gamma^* \rightarrow Q \times \Sigma^*$

$\delta(A, a, z_0) = (B, az_0)$ or (C, ε)



If $w \in L$, exactly 1 path have at final state
If $w \notin L$, exactly 1 path have at non final state.



If $w \in L$, exactly 1 path have at final state.
If $w \notin L$, exactly 1 path have at non final state or zero paths

Date 24/11/2023

Operation.

DPDA : $Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$ (1. 2. 3. 4. 5.)

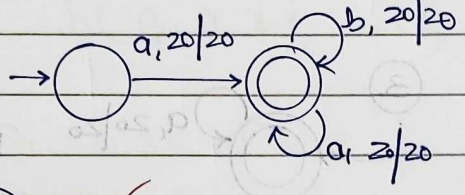
PDA : $Q \times \Sigma_\varepsilon \times \Gamma^* \rightarrow 2^{Q \times \Gamma^*}$ (1. 2. 3. 4. 5.)

How to find curative the operation?

→ 3/5.

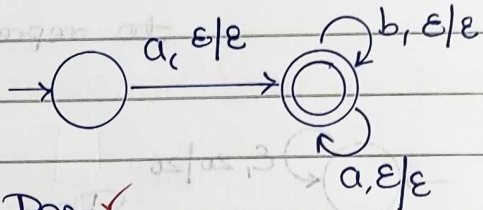
Construction of PDA & DPDA

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



DPDA ✓

PDA ✓



PDA ✓

DPDA ✗

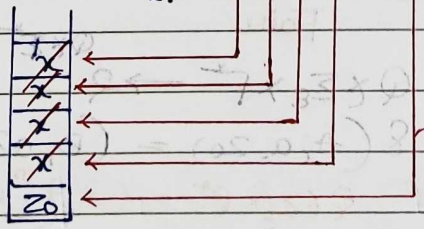
Note: Every finite automata is convertable to PDA & DPDA.

(2.) $L = \{a^n b^n \mid n \geq 1\}$.

Every 'a' push 'x'.

aaaa bbbb $

push x. every 'b' pop 'x'.

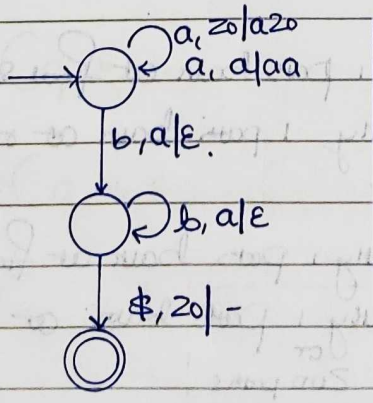


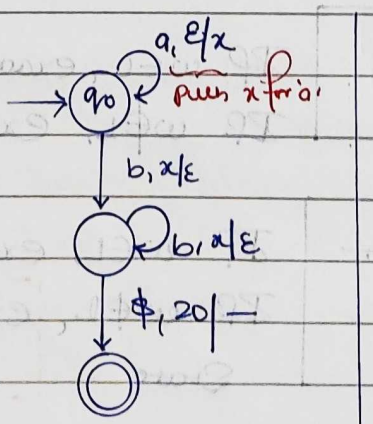
Accept (go to final).

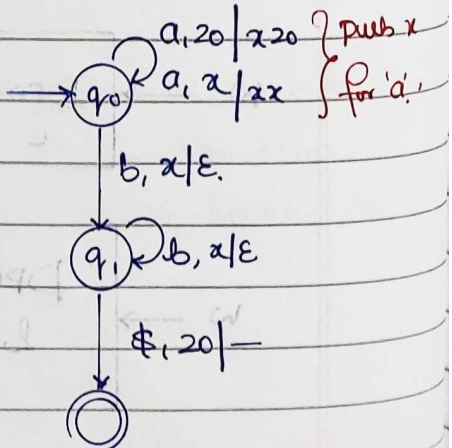
Step 1: Take bigger valid string

Step 2: Make logic using Stack

Step 3: Construct PDA.

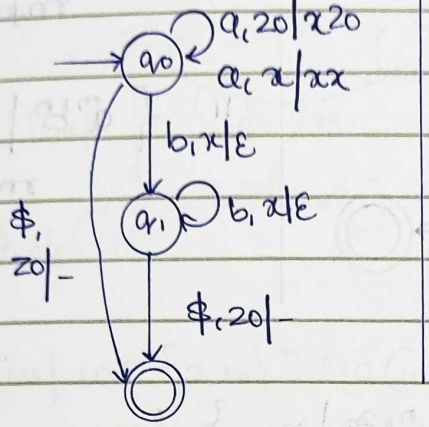


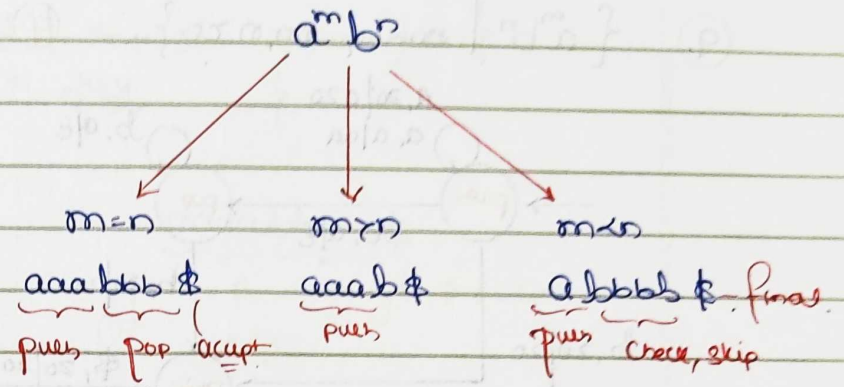




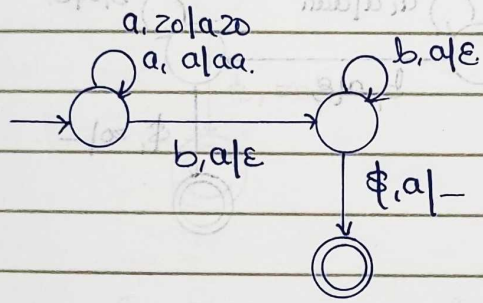
③ $L = \{a^m b^n \mid n \geq 0\}$

$= ② \cup \{\varepsilon\}$

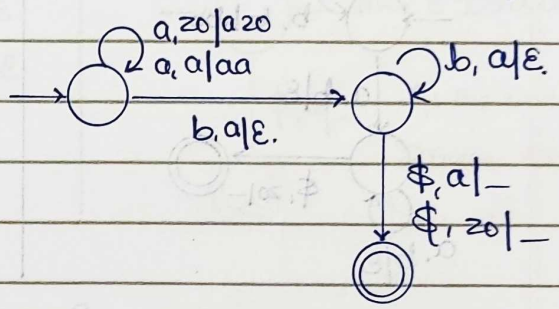




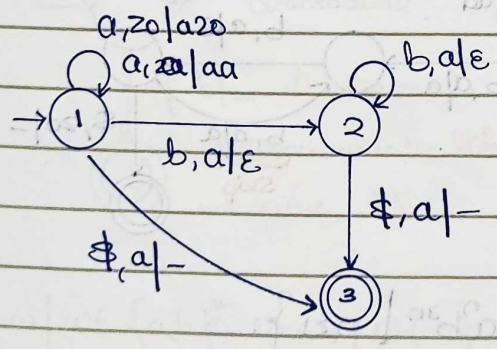
④ $\{a^m b^n \mid m > n,\ m, n \geq 1\}$.



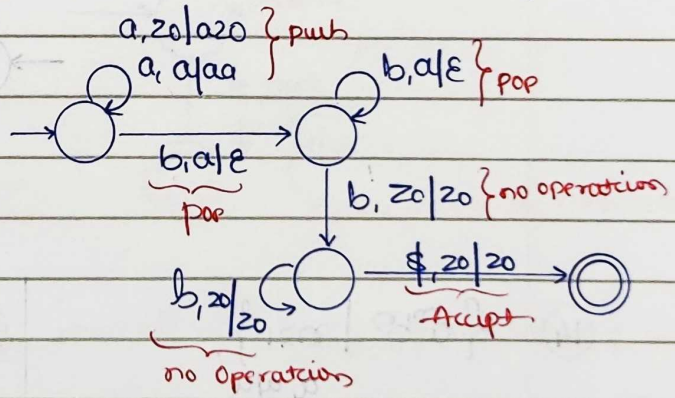
⑤ $\{a^m b^n \mid m \geq n,\ m, n \geq 1\}$.



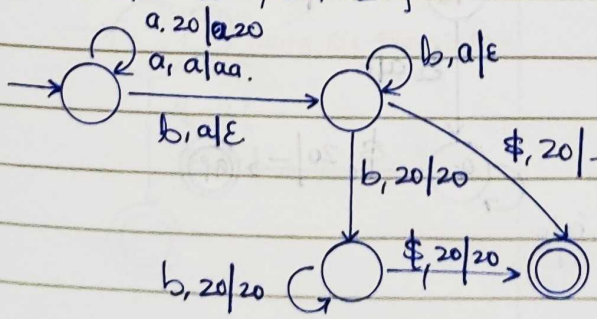
⑥ $\{a^m b^n \mid m > n,\ m, n \geq 0\}$.



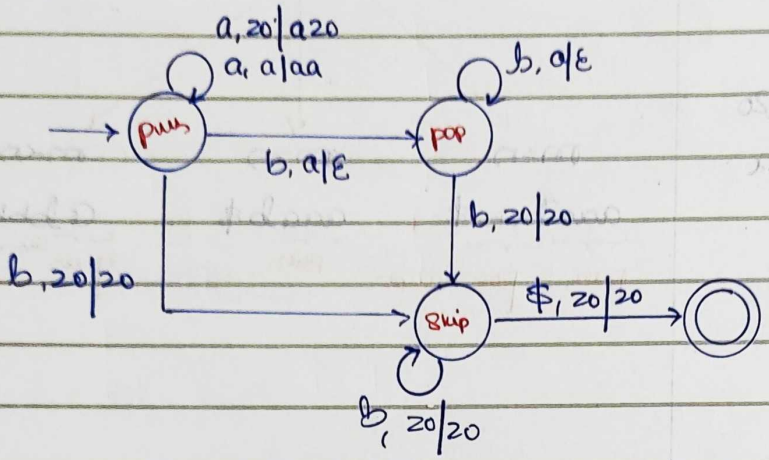
⑦ $\{a^m b^n \mid m < n,\ m, n \geq 1\}$.



⑧ $\{a^m b^n \mid m \leq n,\ m, n \geq 1\}$.



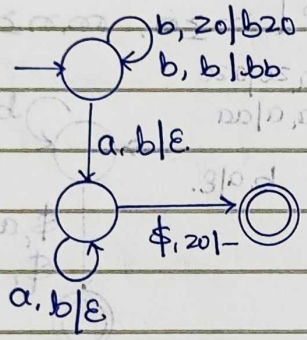
(9) $\{a^m b^n \mid m \neq n,\ m,n \geq 0\}$. - (7) $\cup \{b^+\}$



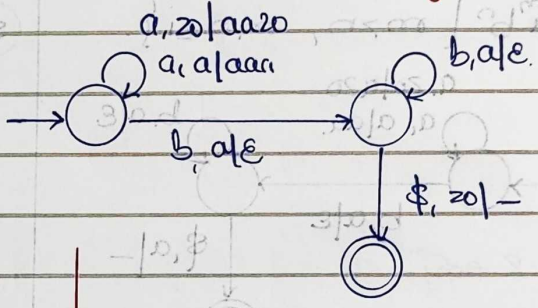
(10) $\{a^m b^n \mid m,n \geq 1,\ m \neq n\}$

(11) $\{a^m b^n \mid m,n \geq 0,\ m \neq n\}$

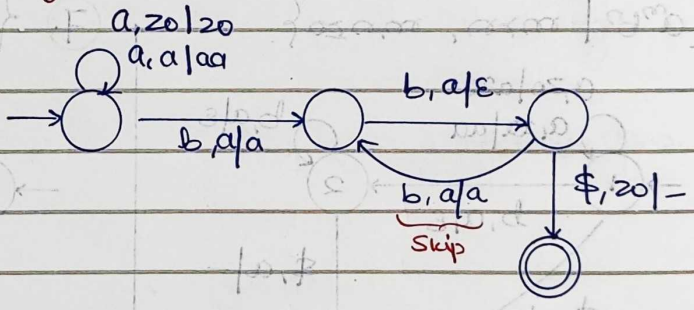
(12) $\{b^n a^n \mid n \geq 1\}$



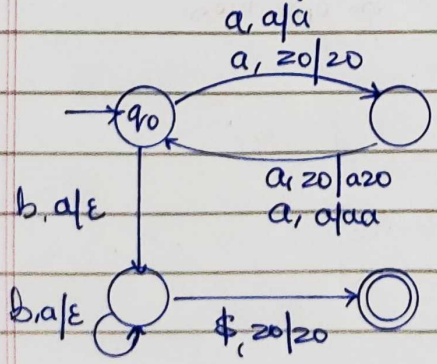
(13) $\{a^n b^{2n} \mid n \geq 1\}$ Logic 1:



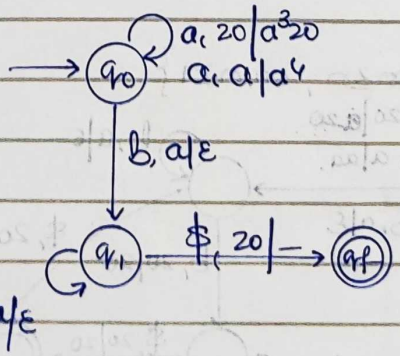
Logic 2:



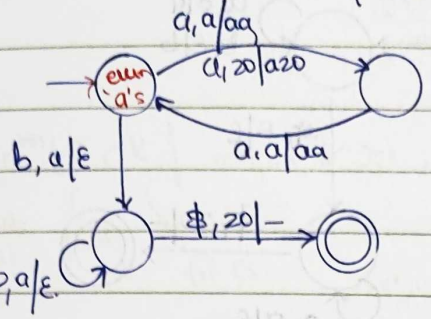
(14) $\{a^m b^n \mid n \geq 1\}$.



(15) $\{a^n b^{3n} \mid n \geq 1\}$.

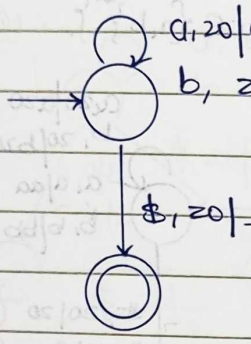


(16) $\{a^{2n}b^{2n} \mid n \geq 1\} = \{a^m b^n \mid m, n \geq 1,\ m = n = \text{even}\}$
$= \{a^2b^2, a^4b^4, \ldots\}$.



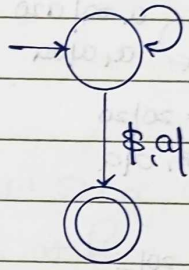
aaaabbbb $
push a's — every b, pop 'a'
for each a ✓
guarantee even 'a's

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



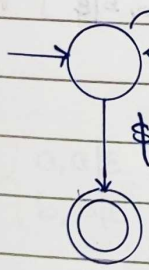
a, a/aa
b, b/bb
a, b/ε
b, a/ε.

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



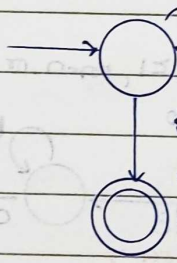
At the end of i/p String, Stack has a's.

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



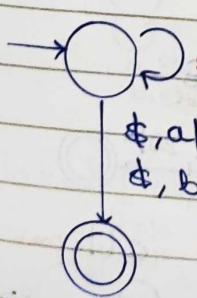
At the end, Stack will be left with b's.

(20) $\{w \mid w \in \{a,b\}^*, n_a(w) \leq n_b(w)\}$.
equal or more b's.

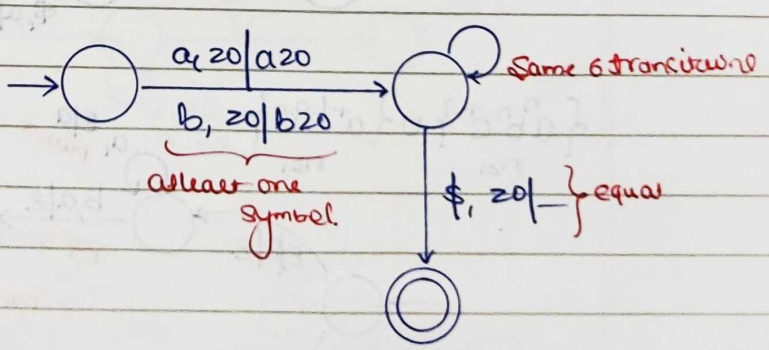


replace with 'a' if $n_a(w) \geq n_b(w)$

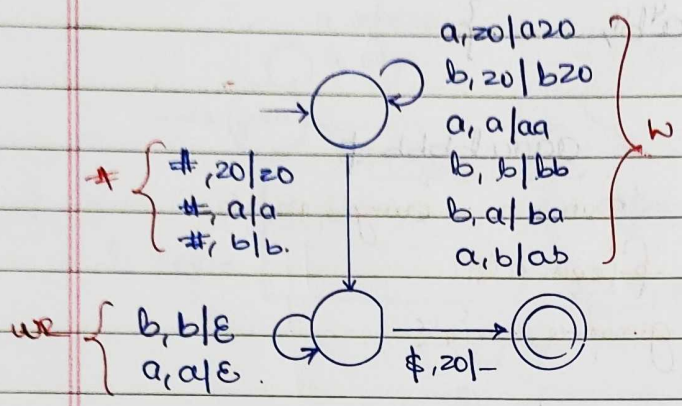
(21) $\{w \mid w \in \{a,b\}^*, n_a(w) \neq n_b(w)\}$.
a's more or b's more.



Note: $L = \{w \mid w \in \{a,b\}^+, n_a(w) = n_b(w)\}$

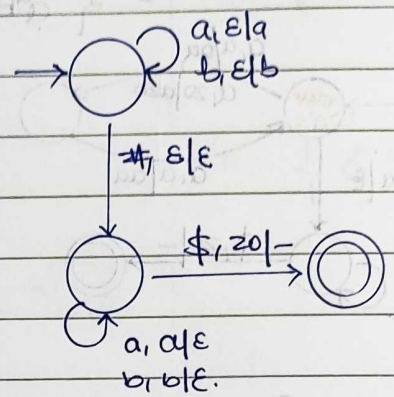


(23). $L = \{ w \# w^R \mid w \in \{a,b\}^* \}$.

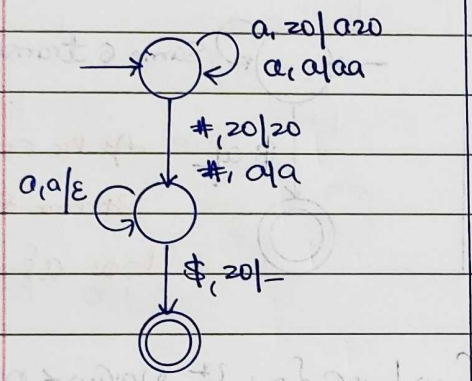


PDA ✓

DPDA ✗

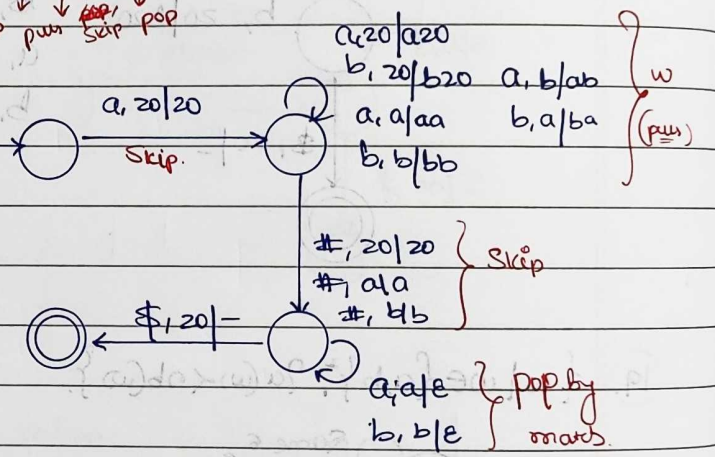


(24). $L = \{ a^n \# a^n \mid n \geq 0 \}$.

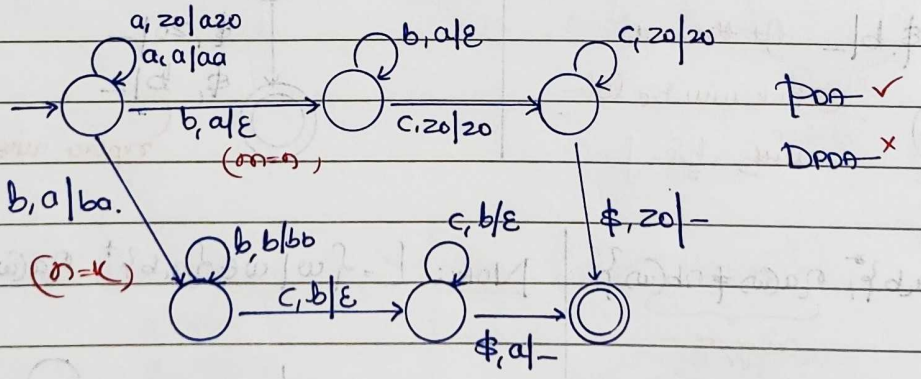


(25) $\{ a w \# w^R \mid w \in \{a,b\}^* \}$.

Skip push Skip pop



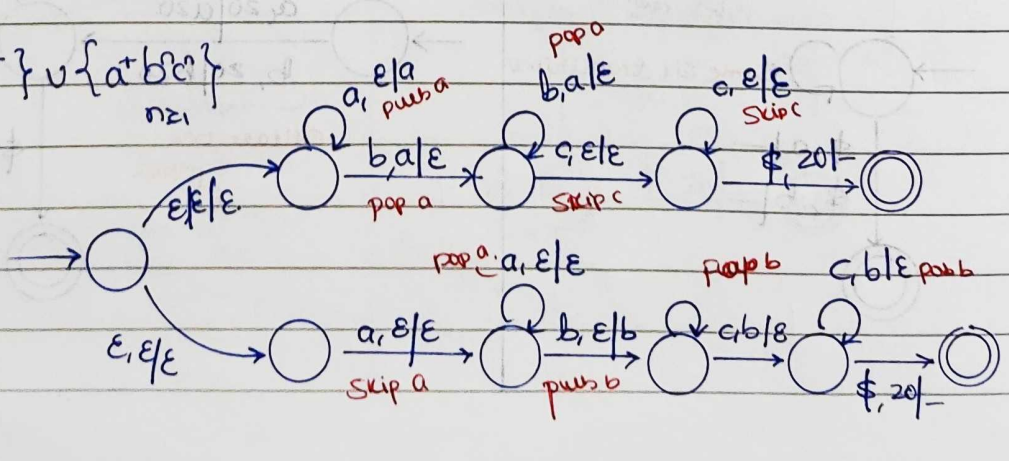
(26) $\{ a^m b^n c^k \mid m,n,k \geq 1, m=n \text{ or } n=k \}$.



PDA ✓

DPDA ✗

$\{ a^n b^n c^+ \}_{n \geq 1} \cup \{ a^+ b^n c^n \}_{n \geq 1}$



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

pop a_i skip b_i's. $a^n b^n c^i$ $n \geq 1$ (a_i's = b's + c's) $a^n b^i c^n$ $n \geq 1$.

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

push a_i (push b_i (pop b_i for c_i. #a's = #c's #b's = #c's.

without reading c_i pop b_i, then every c_i pop a_i.

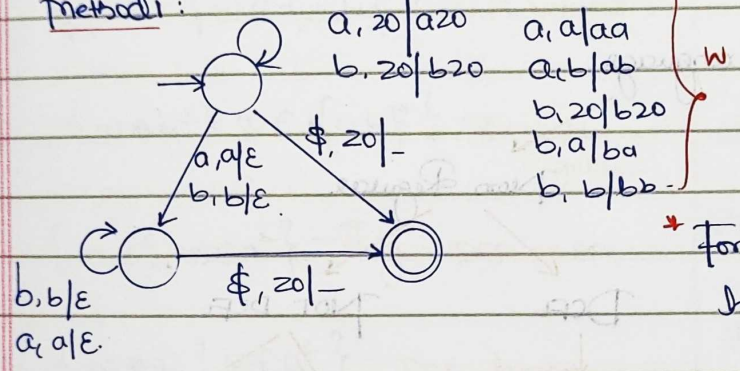
CFLs but not DCFLs

(29) $\{ww^R \mid w \in \{a,b\}^+\}$

When will w^R begin?
↳ Don't know

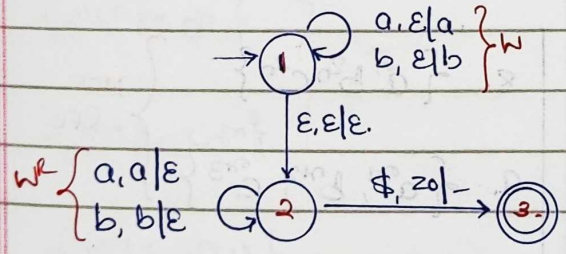
If w^R begins, what's the logic?
↳ present i/p = tos } previous i/p

Method1:

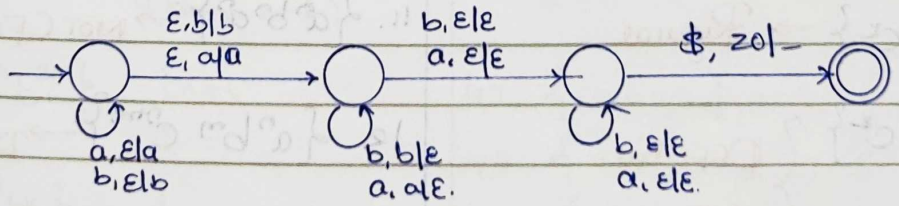


* for valid string, atleast 1 path that halts at final.

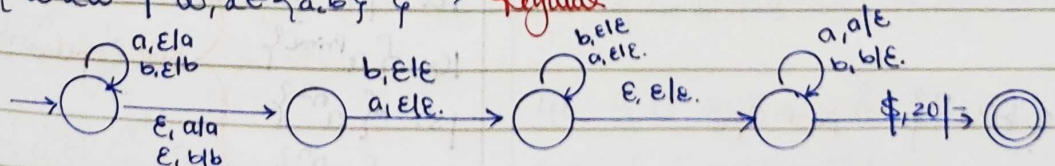
$\{ww^R \mid w \in \{a,b\}^+\}$ → CFL but not DCFL



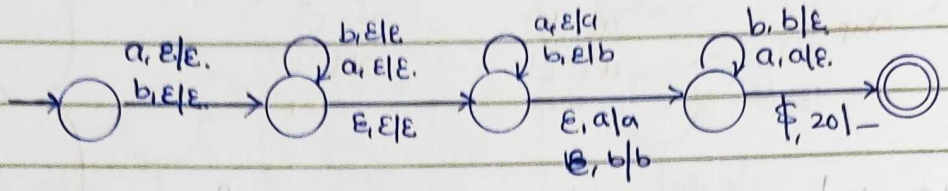
(30) $\{ww^R x \mid w,x \in \{a,b\}^+\}$. → CFL but not DCFL



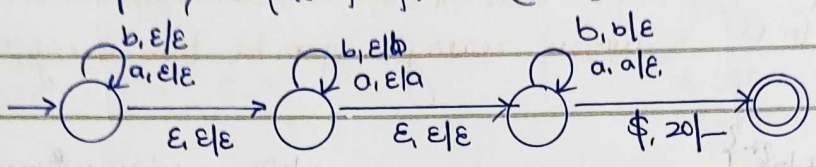
(31) $\{wxw^R \mid w,x \in \{a,b\}^+\}$ → Regular

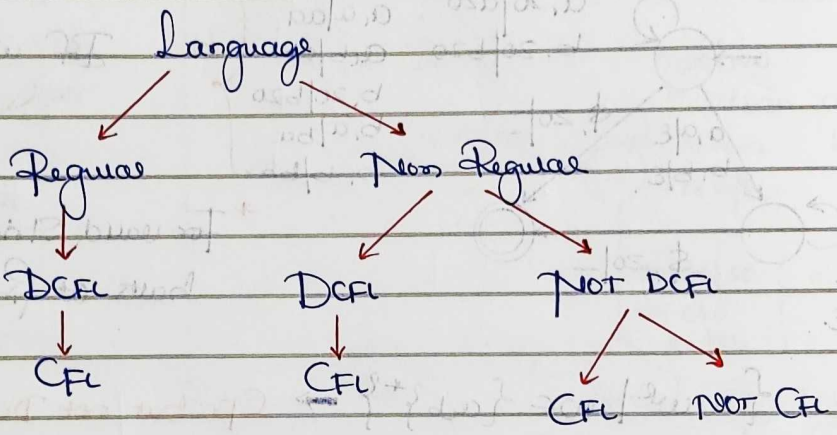


(32) $\{xww^R \mid w,x \in \{a,b\}^+\}$. ⟹ CFL but not DCFL



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





1. $\{a^m b^n\} \Rightarrow a^* b^* =$ Reg. = DCFL ⇒ CFL

2. $\{a^n b^n\}$ } DCFL but not regular

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

4. $\{a^m b^n c^k\}$ ⟹ Regular

5. $\{a^n b^n c^*\}$ } DCFL but not regular

6. $\{a b^n c^n\}$ }

7. $\{a^n b^n c^n\}$ ⟶ Not CFL

8. $\{a^n b^{2n} c^{3n}\}$ } NOT CFL

9. $\{a^{n+1}, b^{n+2}, c^{n+3}\}$ }

10. $\{a^{n+i}, b^{n+j}, c^{n+k}\}$ ⟹ Regular

11. $\{a^n b^n a^n\}$ ⟶ NOT CFL

12. $\{a^n b^m c^{n+m}\}$ ⟶ DCFL but ! Reg

13. $\{a^{n!}\}$ } Not Regular ⇒ Not CFL.

14. $\{a^{Prime}\}$ }

15. $\{a^{n^2}\}$ }

16. $\{a^{n^n}\}$ }

Jo ek Stack se operation ni ho skata wo CFL ni h.

Note: * If L is over 1 Symbol ⟹ FA = PDA
All not regular are not CFLs.

18. $\{a^{n^2}\} = a^* =$ Regular.

19. $\{ww \mid w \in \{a,b\}^*\}$ } not CFL

20. $\{w\#w \mid w \in \{a,b\}^*\}$ }

21. $\{ww^R \mid w \in \{a,b\}^+\}$ → CFL but not DCFL

22. $\{w\#w^R \mid w \in \{a,b\}^*\}$ — DCFL but not Regular.

23. $\{ww \mid w \in a^*\}$ — Reg ⇒ DCFL ⇒ CFL

24. $\{w\#w \mid w \in a^*\}$ — DCFL but not reg.

25. $\{a^n b^n c^k d^k\}$ } DCFL

26. $\{a^n b^k c^k d^n\}$ }

27. $\{a^n b^k c^n d^k\}$ — not CFL

28. $\{a^m b^n c^k d^l\}$ — DCFL
$(m+n = k+l)$

29. $\{a^{2n} b^{3n}\}$ — DCFL

30. $\{a^{n!} b^k\}$ } Not CFL

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

32. $\{a^n b^n c^{k^2}\}$ }

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

34. $\{wwx \mid w,x \in \{a,b\}^+\}$
Also: wxw, wxw^R, xww, xww^R, ww^Rx,
} Regular → DCFL → CFL

35. $\{wwx \mid w,x \in \{a,b\}^+\}$

36. $\{wxw \mid$ " " $\}$ } Not CFL

37. $\{xww \mid$ " $\}$ }

38. $\{ww^Rx \mid$ " $\}$ } CFL but not DCFL

39. $\{xww^R \mid$ " $\}$ }

40. $\{wxw^R \mid$ " " $\}$ — Regular

41. $\{ww^Rx^R \mid w,x \in \{a,b\}^*\}$ — CFL but not DCFL

42. $\{wwxx \mid w,x \in \{a,b\}^+\}$ } Not CFL.

43. $\{ww^Rw \mid$ " $\}$

44. $\{www \mid$ " $\}$

45. $\{w\#w\#w \mid$ " $\}$

46. $\{a^n \# a^n \# a^n\}$ } Not CFL.

47. $\{a^n \# b^n \# c^n\}$ }

48. $\{w\#w^R\#x\#x^R \mid w,x \in \{a,b\}^*\}$ — DCFL

49. $\{a^m\#b^n\#c^k\}$ } Regular.
50. $\{a^m\#a^n\#a^k\}$ }

51. $\{a^m b^n c^k \mid m=n \text{ or } m=k\}$
52. $\{a^m b^n c^k \mid m<n \text{ or } m>k\}$
53. $\{a^m b^n c^k \mid m \neq n \text{ or } m \leq k\}$
54. $\{a^m b^n c^k \mid m \neq n \text{ or } m \neq k\}$.
55. $\{a^m b^n c^k \mid \text{if } (m \neq n) \text{ then } m=k\}$

} CFL but not DCFL.

56. $\{a^m b^n \mid m \neq n \text{ or } m=n\} = a^* b^*$ — Regular.

57. $\{a^m b^n c^k \mid m=n \text{ and } m=k\} = a^n b^n c^n$
58. $\{a^m b^n c^k \mid m \neq n \text{ and } m \leq k\}$

} not CFL

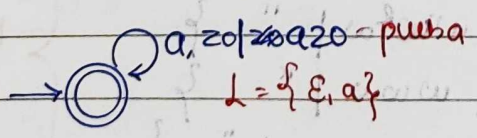
59. $\{a^m b^n c^k \mid m \neq n \text{ or } (m \leq k)\}$ — CFL but not DCFL

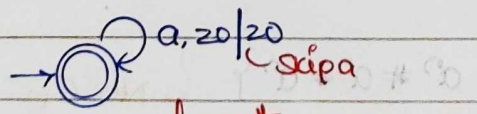
60. $\overline{\{a^n b^n\}}$ — DCFL

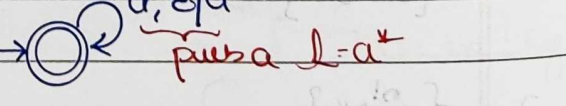
61. $\overline{\{a^n b^n c^n\}}$ — CFL but not DCFL

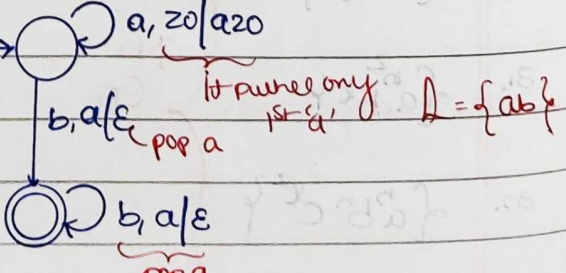
62. $\overline{\{ww \mid w \in \{a,b\}^*\}}$ — CFL but not DCFL.

PDA uses final State.

1. 

2. 

3. 

4. 

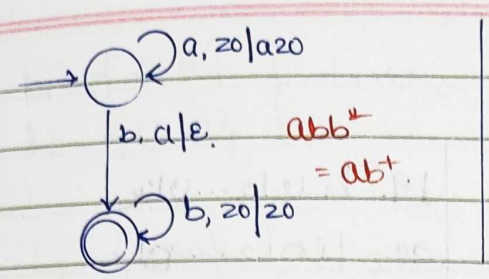
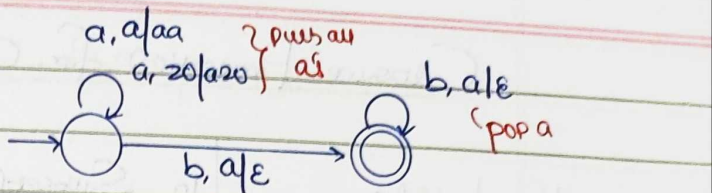
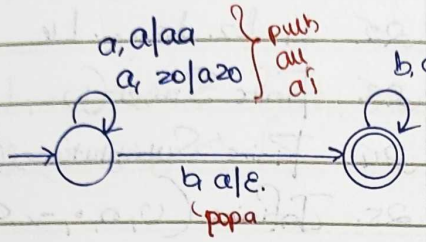
5. 

abb* = ab+

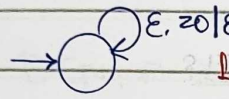
6. 

$\{a^m b^n \mid m \geq n, m, n \geq 1\}$
#ai ≥ #bi.

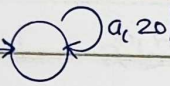
7. 

$\Rightarrow L \cup a^*$
$\{a^m b^n \mid m \geq n \geq 0\}$.

PDA uses empty Stack: (if String is valid, show that Stack is empty at the end of i/p.)

1. 

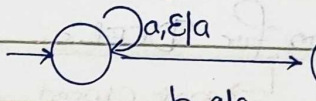
$L = \{\varepsilon\}$.

2. 

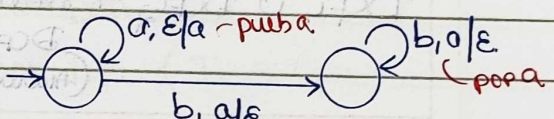
$L = a$

3. 

$L = \{\} = \phi$.

** 4. 

$L = \{a^n b^n\}$.
Assume Stack is empty initially.

** 5. 

Closure Properties for DCFLs (Remember Closed Operations).

1. $L_1 \cup L_2$
2. $L_1 \cap L_2$
3. $\overline{L}$
4. $L_1 - L_2$
5. $L_1 \cdot L_2$
6. L^{Rev}
7. L^*
8. L^+
9. $L_1 \Delta L_2$
10. Subset(L)
11. Prefix(L)
12. Suffix(L)
13. Substring(L)
14. $f(L)$
15. $h(L)$
16. ε-free(L)
17. $h^{-1}(L)$
18. L_1 / L_2
19. $L_1 \cup L_2 \cup L_3 \ldots \cup L_k$
20. $L_1 \cap L_2 \cap L_3 \ldots L_k$
21. $L_1 - L_2 - L_3 \ldots L_k$
22. $L_1 \cdot L_2 \cdot L_3 \ldots L_k$
23. Finite Subset(L)
24. Finite Substitution(L)
25. If finite $\{\cup, \cap, -, \cdot, \subseteq, +\}$.

Closure Properties for CFLs

1. $L_1 \cup L_2$	10. Subset(L)	19. $L_1 \cup L_2 \ldots \cup L_k$
2. $L_1 \cap L_2$	11. Prefix(L)	20. $L_1 \cap L_2 \ldots \cap L_k$
3. $\bar{L}$	12. Suffix(L)	21. $L_1 - L_2 - L_3 \ldots - L_k$
4. $L_1 - L_2$	13. Substring(L)	22. $L_1 . L_2 . L_3 \ldots L_k$
5. $L_1 . L_2$	14. $f(L)$	23. Finite Subset(L)
6. L^{Rev}	15. $h(L)$	24. Finite Substitution(L)
7. L^+	16. ε-free $h(L)$	25. Infinite $(\cup, \cap, ^\circ, -, \subseteq, f)$
8. L^+	17. $h^{-1}(L)$	
9. $L_1 \Delta L_2$	18. L_1 / L_2	

1. Union for DCFLs

→ not closed

$DCFL_1 \cup DCFL_2$ → need not be DCFL (Always CFL)

Example 1:

$DCFL_1 \cup DCFL_2 \Rightarrow$ Not DCFL possible

$\{a^n b^n c^*\} \cup \{a^* b^n c^n\} = \{a^m b^n c^k \mid m=n \text{ or } n=k\}$. (not DCFL)

Example 2:

$DCFL_1 \cup DCFL_2 \rightarrow$ DCFL possible

$a^* \cup a^* \Rightarrow a^*$ DCFL.

Union for CFLs

→ Closed

$CFL_1 \cup CFL_2 \Rightarrow$ Always CFL

Algorithm:

$CFL_1 \Rightarrow$ Write CFG, with S_1 as start

$CFL_2 \Rightarrow$ " " " S_2 "

↓

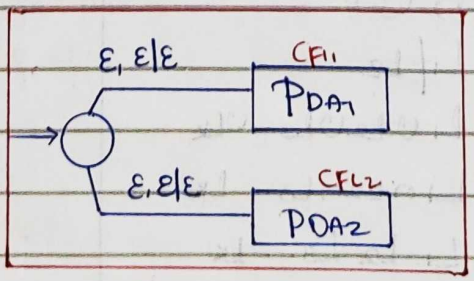
Add new start S

and append $S \rightarrow S_1 | S_2$

$S_1 \rightarrow S_1 | S_2$

CFG1

CFG2



$CFL_1 \cup CFL_2$

1. $L_1 = \{a^n b^n\}$, $L_2 = \{b^n a^n\}$ $\Big\}$ $L_1 \cup L_2 = \{a^n b^n\} \cup \{b^n a^n\}$ = DCFL.

2. $L_1 = \{a^n b\}$, $L_2 = (a+b)^*$ $\Big\}$ $L_1 \cup L_2 = (a+b)^*$ → Regular.

3. $L_1 = a^* b^* c^*$, $L_2 = a^n b^n c^*$ $\Big\}$ $L_1 \cup L_2 = a^* b^* c^*$ → Regular.

(2)

Intersection for DCFLs	Intersection for CFLs
→ not closed	→ not closed

$DCFL_1 \cap DCFL_2$ → May or may not be DCFL (may or may not be CFL)

$L_1 = a^n b^n c^*$, $L_2 = a^* b^n c^n$ $\Big\}$ $= L_1 \cap L_2 = \{a^n b^n c^n\}$ (not DCFL, not CFL (It is CSL))

$CFL_1 \cap CFL_2$ → May or may not be CFL.

1. $L_1 = \phi$, $L_2 = \{a^n b^n\}$ → $L_1 \cap L_2 = \phi$
2. $L_1 = \Sigma^*$, $L_2 = \{a^n b^n\}$ → $L_1 \cap L_2 = L_2 = \{a^n b^n\}$
3. $L_1 = \{ab^n c^n\}$, $L_2 = \{a^n b^n c\}$ = $L_1 \cap L_2 = \{abc\}$.

(3)

Compliment for DCFLs	Compliment for CFLs
→ Closed	→ not closed
$\overline{DCFL}$ = Always DCFL	$\overline{CFL}$ = either CFL or not CFL
L is DCFL ↓ Construct DPDA ↓ f ↔ nf ↓ Modified DPDA ↓ $\bar{L}$ is DCFL	$L = \overline{\{a^n b^n c^n\}}$ is CFL ⇓ $\bar{L} = \{a^n b^n c^n\}$ is not CFL (it is CSL)

$\overline{\{a^n b^n\}} = (a+b)^* - \{a^n b^n\}$.

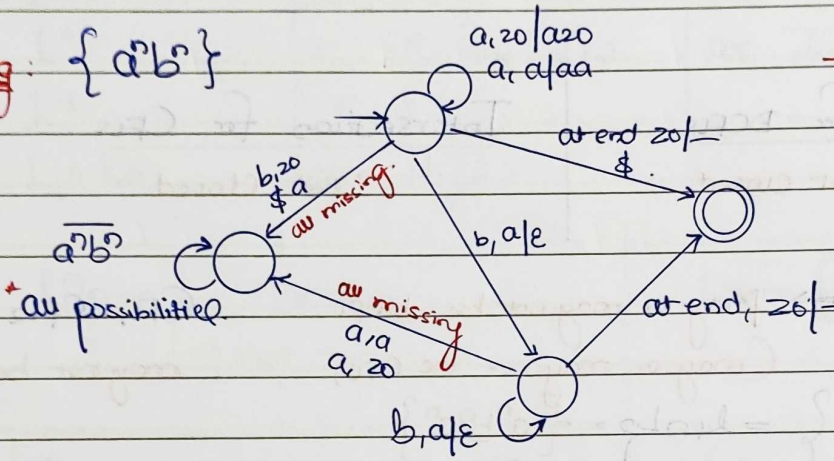
$= \Sigma^* ba \Sigma^* \cup \{a^m b^n \mid m \neq n\}$

if the string is not in a^*b^* — If the string is in a^*b^*.

$L = a^*b^* = \{a^m b^n \mid m = n \text{ or } m \neq n\}$

$\hookrightarrow \overline{L} = \Sigma^* ba \Sigma^*$

eg: $\{a^n b^n\}$



final ↔ non final
Interchange to get Compliment

$L = \overline{\{a^n b^n c^n\}} = (a+b+c)^* - \{a^n b^n c^n\}$

$= (a+b+c)^* - \{a^m b^n c^k \mid m = n = k\}$

$=$ CFL but not DCFL

1. $\overline{\{a^n b^n\}}$ → DCFL
2. $\overline{\{a^n b^n c^n\}}$ → CFL but not DCFL
3. $\overline{\{a^* b^*\}}$ → Regular
4. $\overline{\{a^m b^n c^k \mid m < n < k\}}$ → CFL but not DCFL

	DCFLs	CFLs
∪	×	✓
∩	×	×
$\overline{L}$	✓	×

Note: I. Compliment of a CFL is either CFL or not CFL

II. Compliment of not CFL is may or may not be CFL

III. Compliment for CFLs is not closed.

(4) Difference

→ Not closed for DCFLs

→ Not closed for CFLs

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

I. $DCFL_1 - DCFL_2 \rightarrow DCFL_1 \cap \overline{DCFL_2} \Rightarrow DCFL_1 \cap DCFL_3$ → need not be DCFL

II. $CFL_1 - CFL_2 \rightarrow CFL_1 \cap CFL_2$ → need not be CFL

L_1 is DCFL
L_2 is CFL

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

1. $L_1 \cup L_2$ is always CFL
2. $L_1 \cap L_2$ is need not be CFL
3. $\bar{L}_1 \cup \bar{L}_2$ is need not be CFL
4. $\bar{L}_1 \cup L_1$ is Σ^* is regular
5. $\bar{L}_2 \cup L_2$ is Σ^* is regular.

(5) Concatenation:

→ Not closed for DCFLs
→ Closed for CFLs

$DCFL_1 \cdot DCFL_2$ → need not be DCFL (~~Always CFL~~)

$CFL_1 \cdot CFL_2$ → ~~Always~~ CFL

$L_1 = \{a^n b^n\}$
$L_2 = \{a^n b^{2n}\}$

$L_1 \cdot L_2 = \{a^{n_1} b^{n_1} a^{n_2} b^{2n_2}\}$ — not DCFL

aaab
push.

L_1 is CFL = $CFG_1(S_1)$
L_2 is CFL = $CFG_2(S_2)$

Add $S \rightarrow S_1 S_2$ $\Rightarrow$ $S \rightarrow S_1 S_2$ CFG_1, CFG_2.

I. $\{a^n b^n a^k b^k\}$ → DCFL
II. $\{a^n b^n a^k b^{2k}\}$ → CFL but not DCFL
III. $\{a^n b^n a^k b^{2k} \mid n \geq 1\}$ → DCFL

always comes

(6) Reversal

→ Not closed for DCFLs
→ Closed for CFLs

I. $(DCFL)^{Rev}$ → need not be DCFL

$L = \{a^n b^n a^k b^{2k} \mid \substack{n \geq 1 \\ k \geq 0}\}$ "is DCFL".

$L^{Rev} = \{b^{2k} a^k b^n a^n \mid n \geq 1, k \geq 0\}$ is not DCFL

L is CFL

$L = a^n b^n$

$S \Rightarrow aSb \mid \varepsilon$

↓ every rule, recursive

$S \rightarrow bSa \mid \varepsilon$

$L = b^n a^n$ CFL

7. Kleene Star: → Not Closed for DCFLs
→ Closed for CFLs

8. Kleene plus: → Not closed for DCFLs
→ Closed for CFLs

L is DCFL → L^+ need not be DCFL

$L = \{a\}$

$S \rightarrow a$

add new start X and Add $X \rightarrow XS \mid \varepsilon$

$X \rightarrow XS \mid \varepsilon$

$S \rightarrow a$

$L = a^+$ is CFL

9. $L_1 \Delta L_2 = (L_1 - L_2) \cup (L_2 - L_1)$
$= (L_1 \cup L_2) - (L_1 \cap L_2)$
→ Not closed for DCFLs & CFLs

10. Subset of DCFL is need not be DCFL.
Subset of CFL need not be CFL.

$L = (a+b)^*$

$\phi \subseteq L$	$a^* \subseteq L$	$a^n b^n \subseteq L$	$ww^R \subseteq L$	$a^n b^n a^n \subseteq L$
Regular.	Reg.	not Reg	not reg	not regular
		DCFL	not DCFL	not CFL
			~~not~~ CFL	not DCFL.

Push Down Automata (Continued)

M T W T F S S
Page No.:
Date: 25/11/23
YOUVA

Regular Closures for DCFLs:

1. DCFL ∪ Regular
2. DCFL ∩ Regular
3. DCFL − Regular
4. Regular − DCFL

} Always DCFL (need not be regular)

Note:
* DCFL ∩ Regular → Always DCFL
* Set of all DCFLs ∩ Set of all regulars → Set of all regulars

Regular Closures for CFLs:

1. CFL ∪ Regular
2. CFL ∩ Regular
3. CFL − Regular

} Always CFL

4. Regular − CFL → need not be CFL.

I. DCFL | ∪ ∩ − / . | Regular = DCFL

II. CFL | ∪ ∩ − / | Regular → CFL.

1. $L = \{a^n b^n\} \Rightarrow$ prefix $(L) = \{a^m b^n \mid m \geq n\}$
   suffix $(L) = \{a^m b^n \mid m \leq n\}$
   substring $(L) = \{a^* b^*\}$.

2. $L = \{a^n b^n\}$, $f(a) = \{0\}$, $f(b) = \{11\} \Rightarrow f(L) = 0^n 11^n = \{0^n, 1^{2n}\}$.

3. $L = \{ww^R \mid w \in \{a,b\}^*\} \Rightarrow$ Prefix $(L) = (a+b)^*$
   Suffix $(L) = (a+b)^*$
   Substring $(L) = (a+b)^*$.

Prefix (DCFL) is DCFL

DCFL → DPDA → mod DPDA → Pref (DCFL)

from initial to final, & can come from any state.

Note: Language L satisfies prefix property if every w in L is not prefix of any other string in L.

Q1. $L = a^*$, does this satisfy prefix property?

→ No

Q2. $L = \{a^n b^n \mid n \geq 1\}$

→ Satisfies prefix property

* If any DCFL satisfies prefix property then we can make DPDA with empty stack.

DPDA ≅ DPDA with final state

≠ DPDA with empty stack

DPDA with empty stack ≺ DPDA

→ Set of all DCFLs

≅

Set of all languages accepted by DPDA

≅

Set of all languages accepted by DPDA with final state

≅

Set of all languages generated by LR(1) CFGs

PDA	DPDA
With F.S ≅ With E.S.	With F.S ≠ With E.S (less powerful)
	With F.S: Same as DPDA

CFG ≅ CFL ≅ PDA ≅ PDA with F.S ≅ PDA with E.S.

DCFL ≅ DPDA ≅ DPDA with F.S ≅ LR(k) CFG

$(k \geq 1)$

Inherently Ambiguous CFL:

1. a^* — Regular but not IAL
2. $a^n b^n$ — DCFL but not IAL
3. $\{ww^R \mid w \in \{a,b\}^*\}$ — CFL, not DCFL, and not IAL
4. $\{a^m b^n c^k \mid m=n \text{ or } n=k\}$ — CFL but not DCFL, it is IAL
   - there is no Unambiguous CFG in the world.

Ambiguous CFG

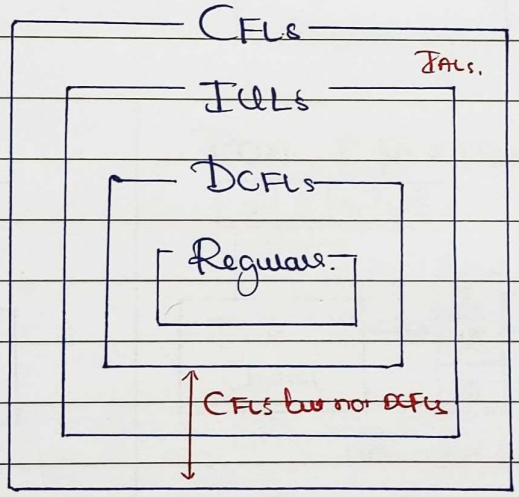
$\exists w, >1$ PT

IAL

L is IAL if every CFG that ~~represents~~ generates L is ambiguous if
L has no Unambiguous CFG

IUL

L is IUL if some Unambiguous CFG exist for L.



* Every Regular is IUL
* Every DCFL is IUL
* CFL is either IUL or IAL.

$\{ww^R \mid w \in \{a,b\}^*\}$.

- → CFL but not DCFL
- → Unambiguous CFL

We can show Unambiguous CFG

$S \to asa \mid bsb \mid \varepsilon$

Unambiguous CFG

$\{a^n b^n c^k \mid n=n \text{ or } n=k\}$

- → CFL but not DCFL
- → Ambiguous CFL

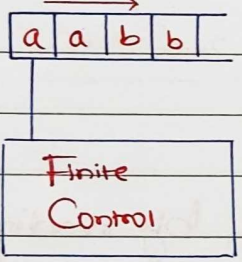
* There is no Unambiguous CFG for this language

If L is Unambiguous CFL → Show atleast one Unambiguous CFG
If L is Ambiguous CFL → we don't have Unambiguous CFG
Every CFG is ambiguous

1. $L = a^* b^*$ → Reg } Ind
2. $L = a^n b^n$ → DCFL }
3. $L = \{ a^m b^n c^k \mid m=n \text{ or } m=k \}$ → CFL but not DCFL, Ind
4. $L = \{ a^n b^n c^k \}$ → DCFL } Ind
5. $L = \{ w \# w^R \mid w \in \{a,b\}^* \}$ → DCFL }
6. $L = \{ ww^R \mid w \in (a+b)^* \}$ → CFL, but not DCFL }

TURNING MACHINE RECURSIVELY ENUMERABLE

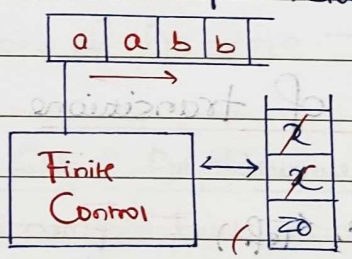
Regulars	CFLs	Recursively Enumerable Languages (REs)
(I) FA → DFA, NFA	(I) PDA	(I) TM
(II) Regular Grammar → LLG, RLG	(II) CFG	(II) Unrestricted Grammars
(III) Regular Expressions		
* These problems can be solved without using memory.	* Can be solved using 1 stack with non determinism	* Can be solved without any restrictions



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

$\delta_{DFA} : Q \times \Sigma \to Q$

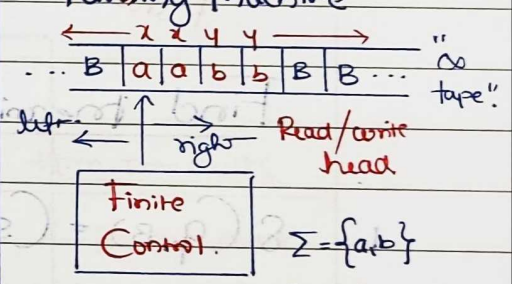
$\delta_{NFA} : Q \times \Sigma_\varepsilon \to 2^Q$



$PDA = (Q, \Sigma, \delta, q_0, F, Z_0, \Gamma)$

$\delta_{PDA} = Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon^* \to 2^{Q \times \Gamma^*}$

$\delta_{DPDA} : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$



$TM = (Q, \Sigma, \delta, q_0, F, B, \Gamma)$

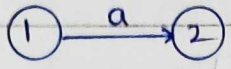
$\delta_{DTM} : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

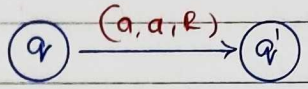
$\delta_{NTM} : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$

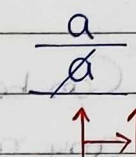
B → Blank Symbol.

* PDA that uses finite stack ≅ FA ≅ Regular Language

* FA + R/W tape + Bidirectional + Infinite tape (head) ≅ Turing m/c

FA	PDA	TM
To remember one symbol 'a':  1. Change State.	1. Change State (or) 2. push/pop into stack (or) 3. Change State and push or pop into stack	1. Change State (or) 2. Write with new Symbol on tape (or) 3. Change State & write

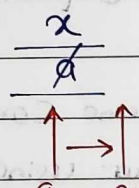




Read 'a'.

$\delta(q, a) = (q', a, R)$

present state, Read, new state, write, direction



Read 'a' but write with 'x'

$\delta(q, a) = (q', x, R)$

Find meaning of transitions

1. $\delta(q, B) = (S, a, \leftarrow \text{left})$ → From state q, by reading B, goes to S, by writing with a and moves left.
2. $\delta(q, a) = (q, a, \rightarrow \text{right})$ → Read 'a' move right.
3. $\delta(q, a) = (q', B, L)$ → i/p = a, Replacing a with B left direction.

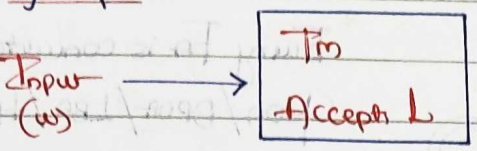
Turing Machine

* It represents REL (Recursively Enumerable language)
* It can accept REL

OR

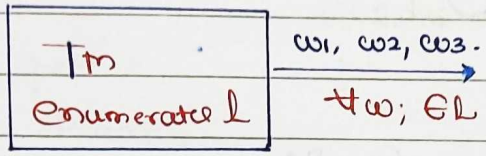
* It can enumerated REL

Acceptor:



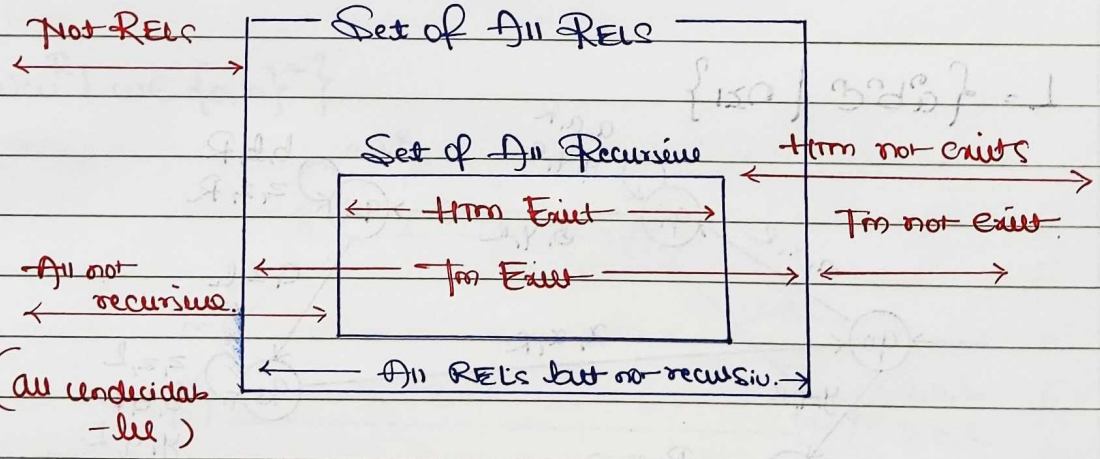
If $w \in L$, TM halts at final state.
If $w \notin L$, either halts at non final or never halts.

Enumerator



TM ⇒ logic exist for valid strings
Accepts REL
(logic may or may not exist for invalid string)

Halting TM ← HTM ⇒ logic exist for valid and invalid
Accepts Recursive language (Decidable)
valid → halts at final
invalid → halts at non final.

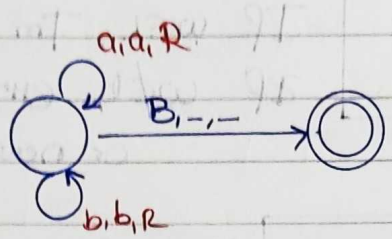


1. Recursive Language (decidable language)
2. REL (Semi decidable language)
3. REL but not recursive
4. Not REL
5. Undecidable language.

TM	HTM
✓	✓
✓	don't know.
✓	×
×	×
don't know	×

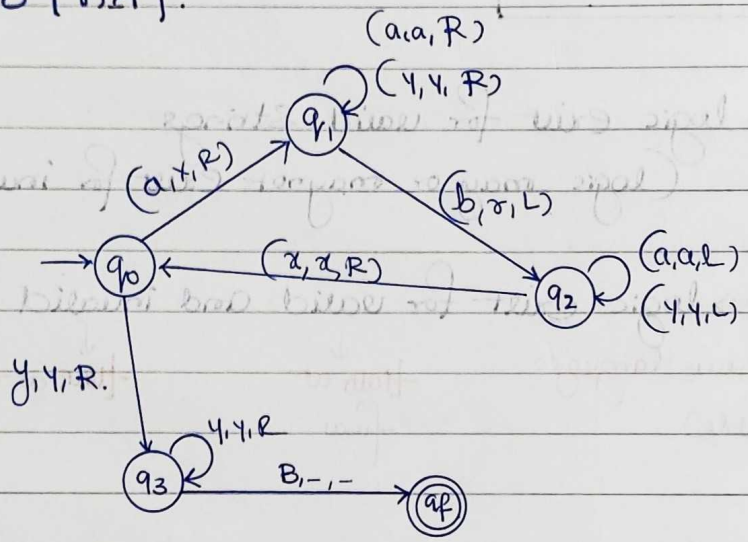
Construction of TM:

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

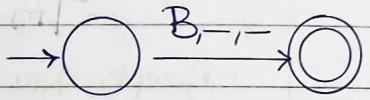


* Every FA is convertable to TM/ PDA/DPDA/LBA/HTM

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

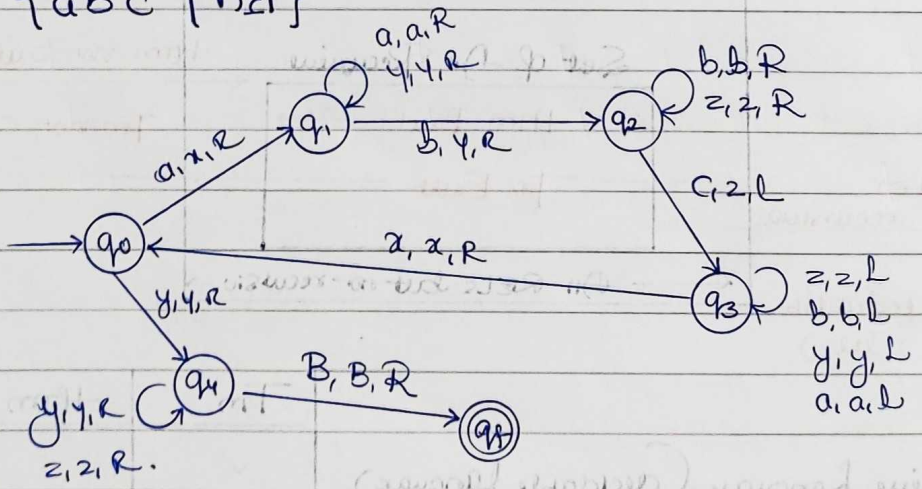


3. $L = \{\varepsilon\}$

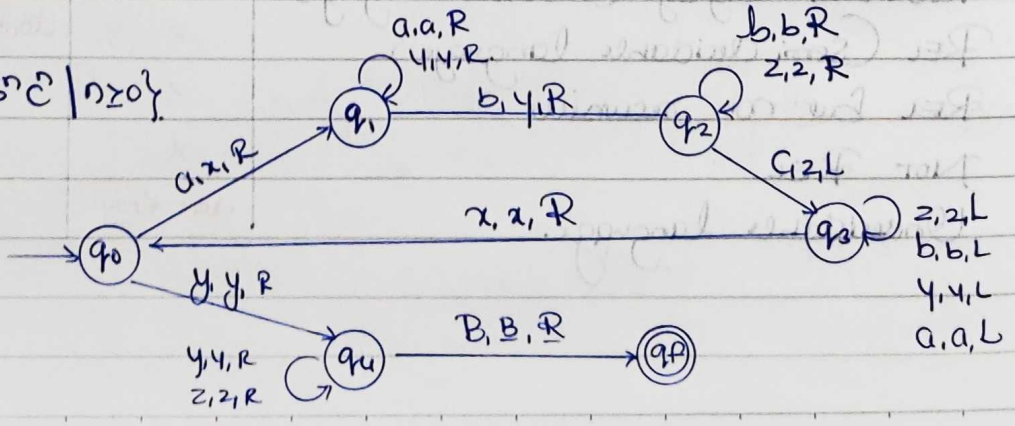


TM accepts only ε

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



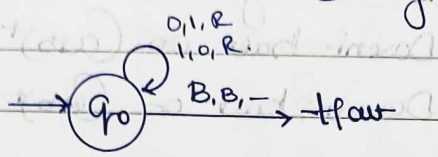
5. $L = \{a^n b^n c^n \mid n \geq 0\}$



Tm.

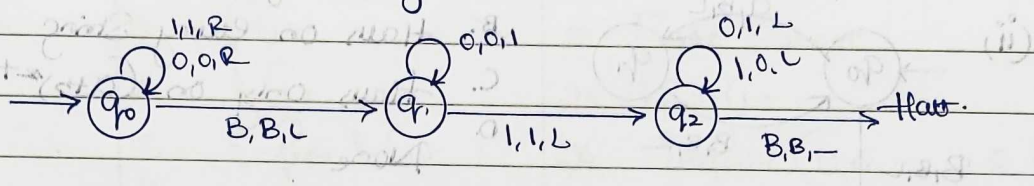
- Acceptance
  - Halts at final → Accepted
  - Halts at non final → Not Accepted.
- Functionality
  - 1's compliment
  - 2's compliment
  - ⋮
- Halting & non-halting
  - Behaviour of Tm/c

6. 1's Compliment of Binary.

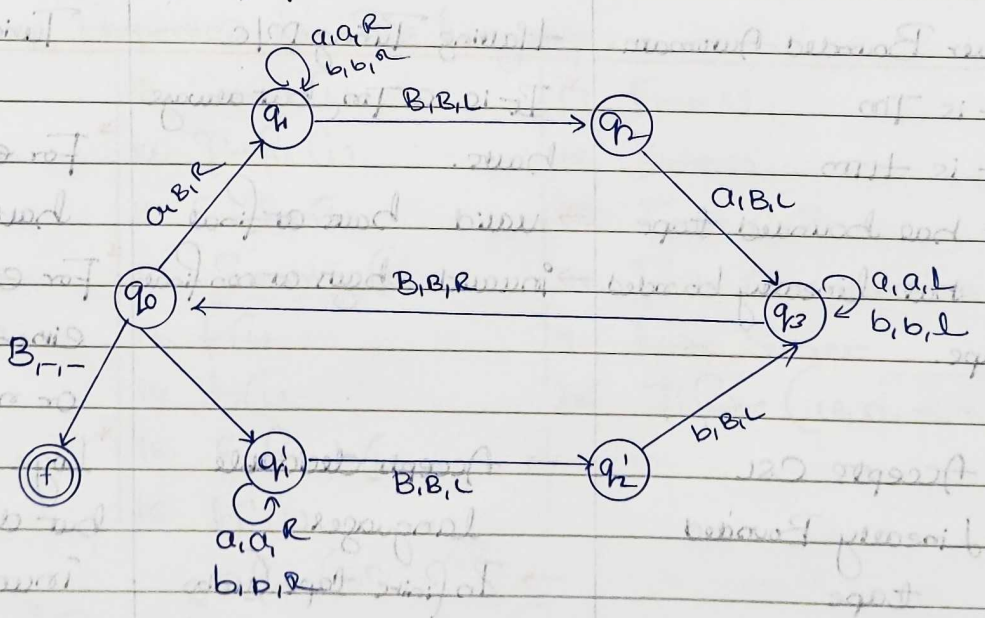


* We do not need final state. There is no concept of valid and invalid input.

7. 2's Compliment of Binary.



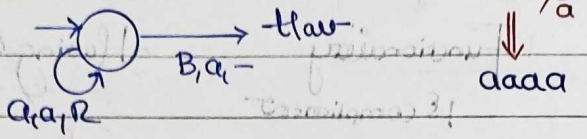
8. $\{ww^R \mid w \in \{a,b\}^*\}$.



9. $\{w\#w^R \mid w \in \{a,b\}^*\}$

10. $\{w\#w \mid w \in \{a,b\}^*\}$.

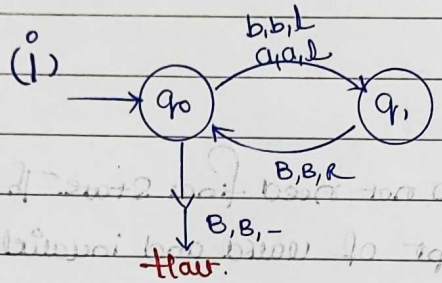
(11.) Unary addition



Q1. What is the language accepted by Tm? What is the functionality?

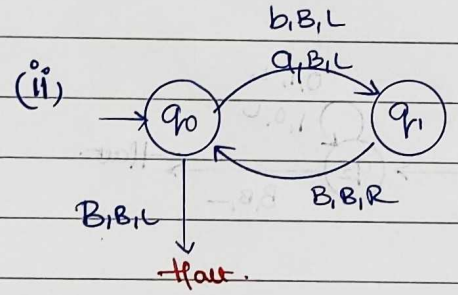
Ans

(i)



A Halts only on ε
B Halts only on (a+b)+
C Doesn't halt on (a+b)+
D Doesn't halt on (a+b)+

(ii)



A. Halts only on ε
B. Halts on every string
C. Halts only on (a+b)*+
D. None

Linear Bounded Automata	Halting Turing M/c	Turing M/c
* It is TM * It is HTM * It has bounded tape * It has linearly bounded tape. → Accepts CSL → Linearly Bounded tape → Always halts	* It is a TM, but always halts. → valid halts at final → invalid halts at non final → Accepts decidable languages → Infinite tape (unbounded). → Always halts	* For every valid string halts at final * For every invalid string, either halts at non final or never halts * Logic exist for valid but don't know about invalid. → Accepts REL Infinite tape (unbounded)

CSLs	Recursives	RELs
* Every regular * Every CFL. * $a^n b^n c^n$ * a^{Prime} * $\{ww \mid w \in \{a,b\}^*\}$ * $a^{n!}$	* Every CSL * $\{\langle r_1, r_2 \rangle \mid r_1, r_2$ are regular expression $L(r_1) = L(r_2)\}$	* Every CSL * Every Recursive * Set of all regular languages.

Closure Properties for Recursive Languages

1. $L_1 \cup L_2$	~~9.~~ Subset(L)	17. Finite ∪
2. $L_1 \cap L_2$	10. Prefix(L)	18. Finite ∩
3. $\bar{L}$	11. Suffix(L)	19. Finite Diff
4. $L_1 - L_2$	12. Substring(L)	20. Finite Concatenation
5. $L_1 . L_2$	~~13.~~ f(L)	21. Finite Subset
6. L^{Rev}	~~14.~~ h(L)	22. $f_{finite}(L)$
7. L^*	15. ε-free h(L)	~~23.~~ Infinite $(\cup, \cap, -, \cdot, \subseteq, f)$.
8. L^+	16 $h^{-1}(L)$	

Closure Properties for RE language.

1. $L_1 \cup L_2$	~~9.~~ Subset(L)	17. Finite ∪
2. $L_1 \cap L_2$	10. Prefix(L)	18. Finite ∩
~~3.~~ $\bar{L}$	11. Suffix(L)	~~19.~~ Finite diff.
~~4.~~ $L_1 - L_2$	12. Substring(L)	20. Finite Concatenation
~~5.~~ $L_1 . L_2$	13. f(L)	21. Finite Subset
6. L^{Rev}	14. h(L)	~~22.~~ Infinite $(\cup, \cap, -, \cdot, \subseteq, f)$
7. L^*	15. ε-free h(L)	
8. L^+	16. $h^{-1}(L)$	

& Infinite $(\cup, \cap, -, \cdot, \subseteq, f)$

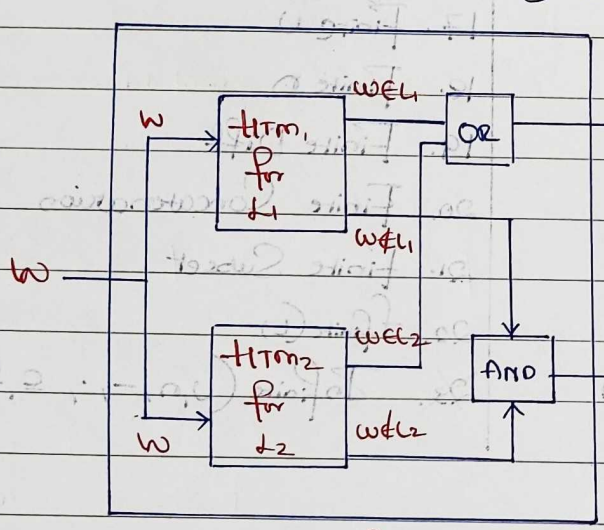
Note: * Subset are not closed for Regular languages, DCFLs, CFLs, CSLs, Recursive, RELs.

* h^{-1} and finite subset are closed for regular, DCFLs, CFLs, Recursive, RELs.

1. Regulars → $\subseteq$, Infinite $(\cup, \cap, -, \circ, \subseteq, f)$
2. DCFLs → $\overline{}$, prefix, h^{-1}, finite subset
3. CFLs → $\cap, \cdot, =, \subseteq, /, F_{EQ}, I_{ALL}$
4. Recursive → $\subseteq, f, h, I_{nf}(\cup, \cap, -, \cdot, \subseteq, f)$
5. RELs → $\cdot, -, \subseteq$, Finite diff, $I_{nf}(all)$

(1.) Union

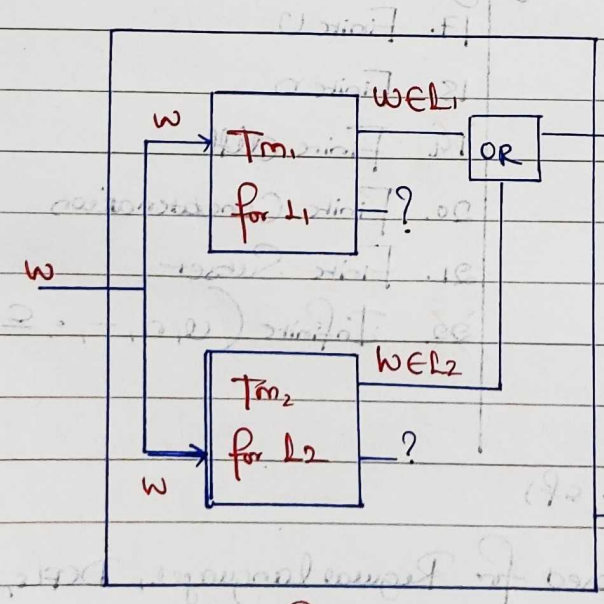
- Closed for recursive languages (decidable): $Rec_1 \cup Rec_2 = Rec$
- Closed for RE language (semi-decideable): $REL_1 \cup REL_2 = REL$



If any HTM halts at final then new HTM halts at final.

If both HTMs halt at non-final then new HTM halts at non-final.

HTM for $L_1 \cup L_2$

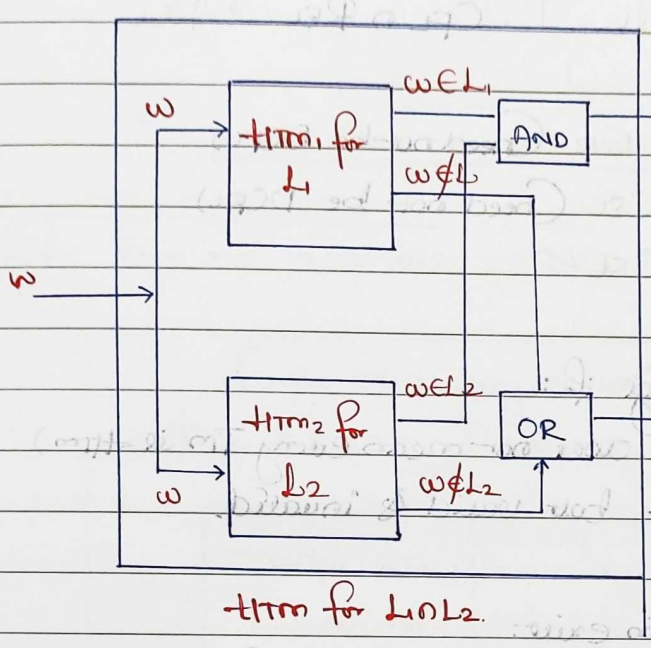


If any TM halts at final then new TM halts at final.

TM for $L_1 \cup L_2$.

(2) Intersection:

- Closed for Recursive languages: $Rec_1 \cap Rec_2 = Rec$
- Closed for RE languages: $REL_1 \cap REL_2 = REL$

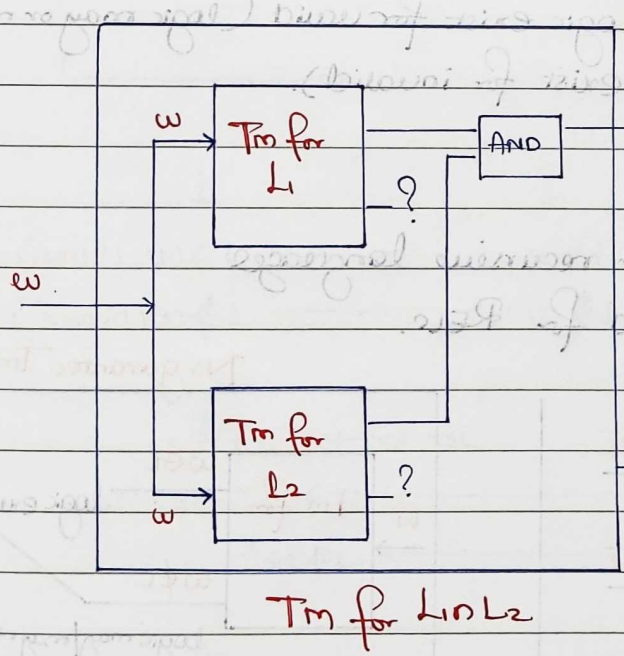


$w \in L_1 \cap L_2$

If both HTM halt at final then new HTM halts at final.

$w \notin (L_1 \cap L_2)$

If any HTM halts at non final then new HTM halts at non final.



$w \in (L_1 \cap L_2)$

If both TM halts at final then new TM halts at final

Q1. A_1, A_2 are Recursive
B_1, B_2 are REL's.

1. $A_1 \cup A_2$ is Recursive
2. $B_1 \cup B_2$ is REL
3. $A_1 \cup B_1$ is REL
4. $A_1 \cap A_2$ is Recursive
5. $B_1 \cap B_2$ is REL
6. $A_1 \cap B_1$ is REL

Note:

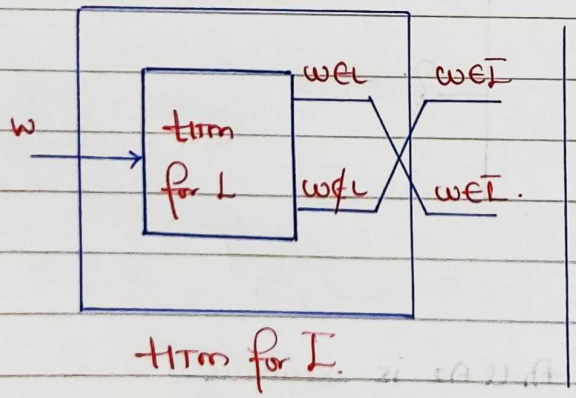
Reg ∩ DCFL → DCFL	CFL ∩ CSL → CSL
Reg ∩ CFL → CFL	CSL ∩ Recursive → Recursive
DCFL ∩ CFL → CSL	Rec ∩ REL → REL
$CFL_1 \cap CFL_2$ → CSL	CFL ∩ REL → REL

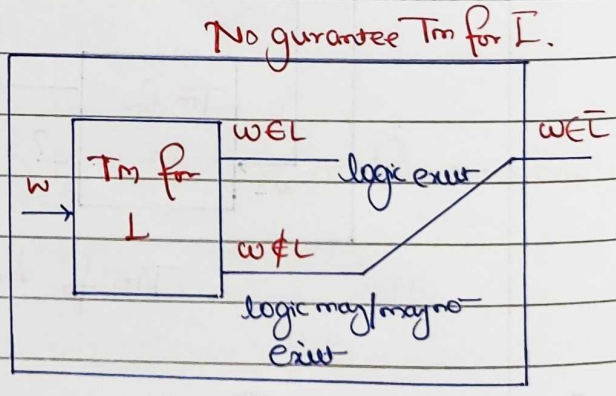
* $DCFL_1 \cap DCFL_2 \Rightarrow$ CSL (need not be DCFL)
* $CFL_1 \cap CFL_2 \Rightarrow$ CSL (need not be DCFL)
* $DCFL_1 \cap CFL \Rightarrow$ CSL

→ L is recursive language if:
1. HTM exist (it does not mean every TM is HTM)
2. Logic exist for both valid & invalid.

→ L is REL if: 1. TM exist.
2. Logic exist for valid (logic may or may not exist for invalid).

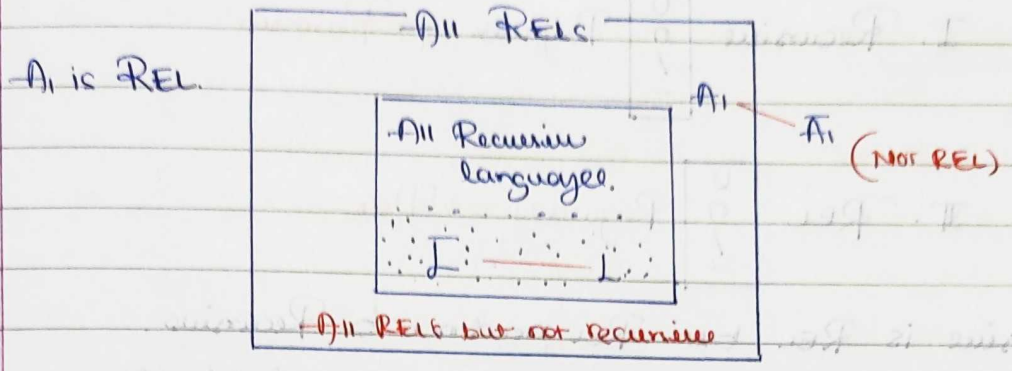
③ Complement
- → closed for recursive languages
- → not closed for RELs.



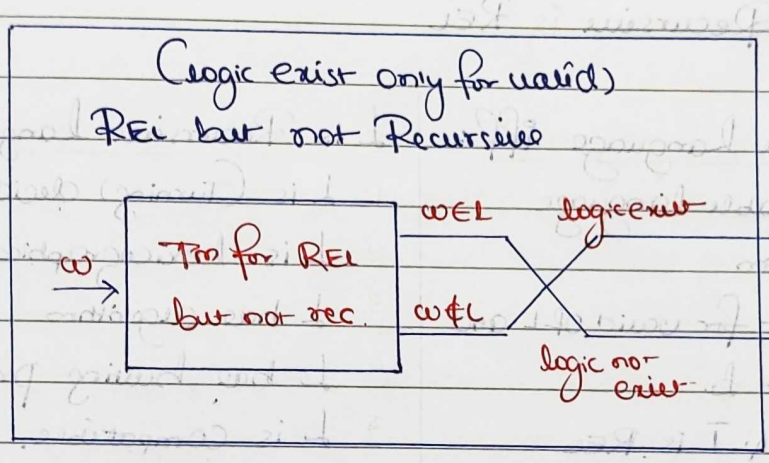


I. Recursive language ⇒ Recursive language

II. REL → Never be "REL but not rec."
REL → Need not be REL
→ Either recursive lang or "Not-REL"



III. REL but not recursive $\Rightarrow$ NOT REL.



L	$\bar{L}$
(valid) $w \in L$	$w \notin L$ (invalid of $\bar{L}$)
(invalid) $w \notin L$	$w \notin L$ (valid of I)

REL but not rec	$\overline{\text{REL but not rec}}$
✓ $w \in L$	$w \notin L$ ✓
× $w \notin L$	$w \in L$ × TM not exist.

Recursive	$\overline{\text{Recursive}}$
✓ $w \in L$	$w \notin L$ ✓
✓ $w \notin L$	$w \in L$ ✓

REL	$\overline{\text{REL}}$
✓ $w \in L$	$w \notin L$ ✓
(don't know) ? $w \notin L$	$w \in L$ (Don't know) ?
	TM may/may not exist.

Note: I. Recursive $\left[\begin{matrix}\cup\\ \cap\\ /\\ -\end{matrix}\right]$ Regular $\Rightarrow$ Recursive

II. REL $\left[\begin{matrix}\cup\\ \cap\\ /\\ -\end{matrix}\right]$ Regular $\Rightarrow$ REL

* Recursive is REL but REL need not be Recursive.
* Decideable is REL but REL need not be decideable
  * Recursive $\cong$ Decideable
  * Recursive is REL

→ **L is Recursive language iff:**

1. L is decideable language
2. L has TM
3. Logic exist for valid of L and
4. invalid of L.
5. L is REL & $\bar{L}$ is REL
6. L has Tm & $\bar{L}$ has Tm.

L is Recursive language if:

1. L is (Turing) decideable
2. L is Lexicographically Enumerable
3. L has algorithm
4. L has having program
5. L is compatible.

→ **L is REL iff:**

1. L has Tm
2. Logic exist for valid members of L
3. L is semi decideable
4. L is enumerable
5. L has program
6. L is turing recognisable
7. L is turning acceptable
8. L is Recognizable (acceptable)
9. L has Unrestricted grammar.

L is undecidable iff:

1. L is not decidable,
2. L is not recursive & no TM.

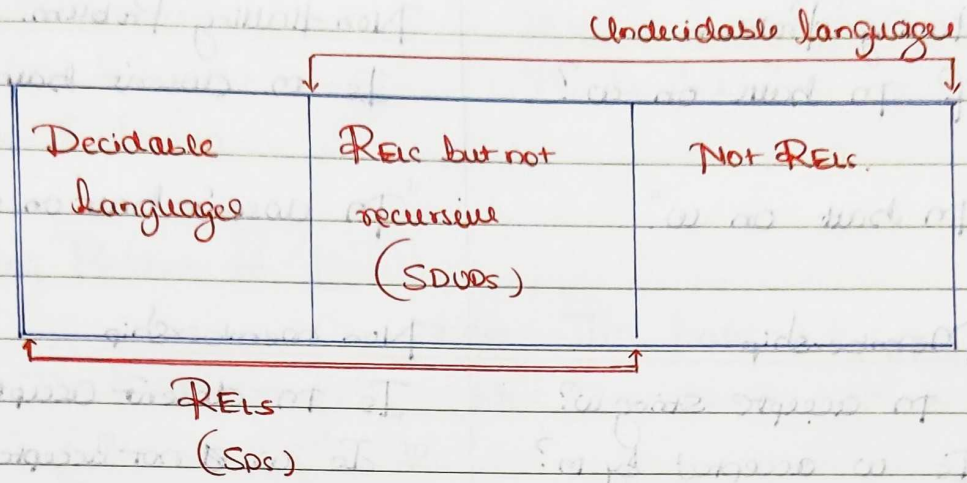
L is REL but not Recursive if:

1. L has Tm but not recursive
2. L has Tm but L has no TM
3. L has Tm but $\bar{L}$ has no Tm
4. L is semi-decidable & undecidable
5. Only valid of L has logic.
6. L is REL & $\bar{L}$ is not REL

L is not REL iff:

1. L is not semi decidable
2. L has no Tm
3. Valid strings of L has no logic

DECIDABILITY



	D	SDUD	NR
Members	valid → logic exist invalid → logic exist	valid → logic exist invalid → logic not exist.	valid → logic not exist.
Machine	HTM exist.	TM exist but HTM not exist	TM not exist.
Language.	Decidable language.	SDUD (REL but not rec)	NOT REL (NOT SD)
Program (Computer)	Halting program exist (Algorithm exist)	Program exist but no algorithm.	Program not exist.

Decision Properties Table (Decision Problems)

	Problems	FA / Regular lang / Reg Exp. / RG.	DPDA / DCFLs	PDA / CFLs / CFGs	HTM / Recursive lang.	TM / RELs / UGs.
1.	Halting	D	D	D	D	UD
2.	Membership	D	D	D	D	UD
3.	Emptiness	D	D	D	UD	UD
4.	Finiteness	D	D	D	UD	UD
5.	Totality	D	D	UD	UD	UD
6.	Equivalence	D	D	UD	UD	UD
7.	Disjoint	D	UD	UD	UD	UD
8.	Set Containment	D	UD	UD	UD	UD

1.	**Halting Problem** If m halts on w? Or "m halts on w".	**Non-Halting Problem.** Is m doesn't halt on w? Or "m doesn't halt on w".
2.	**Membership** Is m accepts string w? Or Is w accepted by m? Or Is $w \in L(m)$? Or "m accepts w".	**Non membership** Is m doesn't accept w? Or Is w is not accepted by m? Or Is $w \notin L(m)$? Or "m doesn't accept w"
3.	**Emptiness** Is m accepts ϕ? Or Is m accepts nothing? Or Is $L(m) = \phi$?	**Non Emptiness** Is m not accepts ϕ? Or Is m accepts something? Or Is $L(m) \neq \phi$?
4.	**Finiteness** Is m accepts finite language? Is $L(m)$ = finite language?	**Non-Finiteness** Is $L(m)$ ≠ finite language? Is $L(m)$ = Infinite language?
5.	**Totality.** Is m accepts everything? Is $L(m) = \Sigma^*$?	**Non Totality.** Is m not accepts everything? Is m not accepting atleast one string? Is $L(m) \neq \Sigma^*$.
6.	**Equivalence** Is $m_1 \cong m_2$? Is $L(m_1) = L(m_2)$?	**Non Equivalence** Is $m_1 \not\cong m_2$? Is $L(m_1) \neq L(m_2)$?
7.	**Disjointness** Is $L(m_1) \cap L(m_2) = \phi$?	**Non Disjointness** Is $L(m_1) \cap L(m_2) \neq \phi$?

8. **Set Containment**
Is $L(m_1) \subseteq L(m_2)$?
Is $L_1 \subseteq L_2$?

Non Set Containment
Is $L_1 \not\subseteq L_2$?

Halting Problem for Tm.

Is Tm halts on ε? → Yes: Tm halts on ε. (logic exist)
→ No: Tm doesn't halt on ε. (logic not exist)

Undecidable — SDUD

Not Halting: Is Tm doesn't halt on w? → Yes: logic not exist
No

Undecidable — NOT RE

Membership for FA/RegL/RegExp/RG

FA → , w → | Is $w \in L(FA)$? | → Yes: $w \in L(FA)$: FA halts at final (logic exist)
→ No: $w \notin L(FA)$: FA halts at non final (logic exist)

Similarly, membership problem is decidable for FA/DPDA/PDA/LBA/TM

Membership for Tm: ⟹ SDUD

Tm → , w → | Is Tm accepts w? | → Yes: $w \in L(Tm)$ given Tm halts at final. (logic exist)
→ No: $w \notin L(Tm)$ given Tm either halts at non final or never halts (logic not exist)

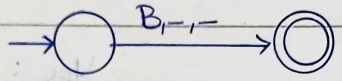
Non membership for Tm → Not RE

Problems:

1. Is FA accepts ε? → Membership for FA → Decidable
2. Is PDA accepts ε? → D
3. Is Tm accepts ε? → SDUD

4. Is $FA_1 \cong FA_2$? → SDUD (↗ Some String ≥1 PT (logic exist))
5. Is CFG Ambiguous? → Yes: CFG is Ambiguous
   → No: CFG is Unambiguous (↳ every string, 1PT (logic not exist))
6. Is grammar CFG? → Decidable
7. Is CFG unambiguous → Not RE

(I.) Can we design Tm that accepts ε?



We can design Tm that accepts ε.

(II). Can we verify arbitrary Tm that accepts ε? → Yes: logic ✓ / No: logic ×
↓ Undecidable & Semi decidable

Languages:

1. { FA | FA accepts ε } → D
2. { ⟨FA, w⟩ | FA accepts w } } Decidable
3. { FA # w | FA accepts w }
4. { Tm | Tm accepts ε } } SDUD
5. { Tm | Tm accepts w }
6. { Tm | Tm accepts some String } → SDUD
7. { Tm | Tm accepts only w } → Yes: Tm is accepting only w.
   ↓ SDUD → No: L(Tm) ≠ {w} Tm not only accepts w.
8. { Tm | Tm accepts 3 length String } } SDUD
9. { Tm | Tm reached state q }
10. { Tm | Tm reached state q within 5 steps } → Decidability.
11. $L = a^n b^n$
12. Finite language
13. Regular language } Decidable
14. DCFL
15. CFL
16. CSL

17. Recursive language → D
18. REL → D or SDUD
19. REL but not Recursive → SDUD
20. Not REL → NR
21. Not Recursive → NR or SDUD
22. Not Regular language → D or SDUD or NR

23. Not CFL → D/SDUD/NR
24. $a^n b^n$ → D
25. Set of (all) Finite languages } SDUD
26. Set of (all) regular languages }
27. Set of DCFLs, CFLs, CSLs, Decidable languages, RELs, SDUDs } SDUD
28. Set of not RELs } NR
29. Set of infinite languages
30. $\{L \mid L \text{ is decidable}\} \cup \{L \mid L \text{ is NR}\}$

Reducibility: $A \leq B$

A is reducable to B

* Every instance of A is mapped into some instance of B
* B is equal or harder than A
* B is atleast as hard as A
  - → $A \leq_p B$ (A is polynomially reducable to B)
  - → $A \leq_m B$ (A is mapping to one reducable to B)

Let $A \leq B$:

1. If A is undecidable then B is undecidable
2. If B is decidable then A is decidable
3. If B is undecidable then A is decidable/undecidable
4. If A is decidable then B is decidable/undecidable
5. If B is REL than A is REL
6. If A is not REL then B is not REL
7. If A takes 100 days then B takes min 100 days
8. If B takes 100 days then A takes max 100 days

If $A \leq B$ and $C \leq B$:

1. If A is decidable then B is ? and C is ?
2. If B is decidable then A is decidable and C is decidable
3. If C is decidable then A is ? and B is ?

→ If $A \leq B$ and $B \leq A$ then $A = B$

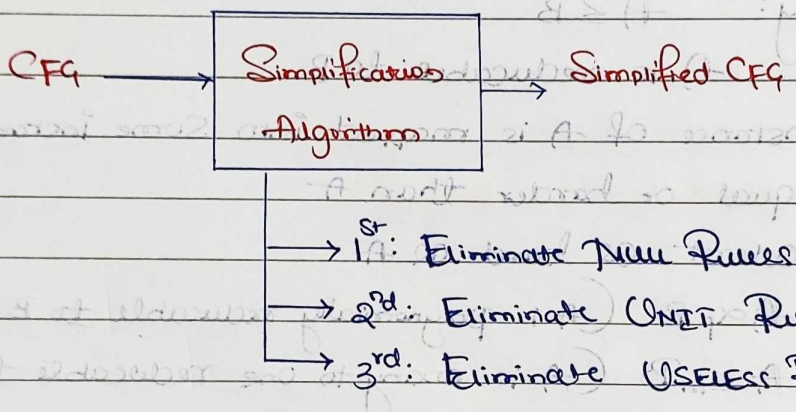
Let $A \leq B \leq C$ then:

1. If C is decidable then A is decidable and B is decidable
2. If A is decidable then B is undecidable and C is undecidable

→ L is SDUD → $\bar{L}$ is NR

→ L is Set of all SDUDL → $\bar{L}$ = Set of all decidables ∪ Set of all NRs

Simplification of CFG



eg:

$S \to AB \mid \varepsilon$
$A \to aA \mid bB \mid c$
$B \to Ba \mid \varepsilon \mid bS$

Step1: $S \to \varepsilon$

$S \to AB$
$A \to aA \mid bB \mid c$
$B \to Ba \mid \varepsilon \mid bs \mid b$

Step2: $B \to \varepsilon$

$S \to AB \mid A$
$A \to aA \mid bB \mid c \mid b$
$B \to Ba \mid bS \mid b \mid a$

eg2:

$S \to Sa \mid Ab \mid Bc \mid AB$
$A \to Ae \mid \varepsilon$
$B \to Bb \mid \varepsilon$

Step1: $A \to \varepsilon$ delete

$S \to Sa \mid Ab \mid Bc \mid AB \mid b \mid B$
$A \to Ae \mid e$
$B \to Bb \mid \varepsilon$

Step2: Delete $B \to \varepsilon$

$S \to Sa \mid Ab \mid Bc \mid AB \mid b \mid B \mid c \mid A \mid \varepsilon$
$A \to Ae \mid e$
$B \to Bb \mid b$

Step3: Delete $S \to \varepsilon$

$S \to Sa \mid Ab \mid Bc \mid AB \mid b \mid B \mid C \mid A \mid a$
$A \to Ae \mid e$
$B \to Bb \mid b$

$X \to \varepsilon$ delete

In whole grammar, add all new productions by putting X with ε

eg3:

$S \to aA \mid$
$A \to bB \mid aS \mid B$
$B \to aA \mid S$

Step 1: Delete $S \to A$

$S \to aA \mid bB \mid aS \mid B$
$A \to bB \mid aS \mid B$
$B \to aA \mid S$

Step 2: Delete $S \to B$

$S \to aA \mid bB \mid aS \mid \cancel{aA} \mid \cancel{S}$
$A \to bB \mid aS \mid B$
$B \to aA \mid S$

Step 3: Delete $A \to B$

$S \to aA \mid bB \mid aS$
$A \to bB \mid aS \mid aA \mid S$
$B \to aA \mid S$

Step 4: Delete $A \to S$

$S \to aA \mid bB \mid aS$
$A \to bB \mid aS \mid aA$
$B \to aA \mid S$

Step 5:

$S \to aA \mid bB \mid aS$
$A \to bB \mid aS \mid aA$
$B \to aA \mid bB \mid aS$

* In given G, if we delete $X \to Y \Rightarrow$ Replace Y with all Y productions.

Eliminate USELESS Productions:

Step 1: Find the nonterminals which derive nothing and eliminate it.

Step 2: Find the unreachable nonterminals and delete them.

$S \to \cancel{aA} \mid b \implies S \to b$

$S \to aA \mid Bb \mid CDe$
$A \to a \mid Sa$
$B \to bB$
$D \to bD \mid eE.$
$F \to g$
$H \to eA$

Step 1: Identify nonterminals derive nothing:

S A B D E F H
↓ ↓ ↓ ↓ ↓ ↓ ↓
aa a X X X g ea

Delete all productions associated with B, D, E.

Step 2: Delete all unreachable symbols

$S \to aH$
$A \to a \mid Sa$

CFG → [Elimination of NULL, UNIT & USELESS RULES] → Simplified CFG

CFG → [Elimination of USELESS RULES] → Reduced CFG.

Normal Forms:

- CNF CFG (Chomsky normal form)
- GNF CFG (Greibach normal form)

CNF CFG	GNF CFG
$V \to VV \mid T$	$V \to TV^*$
↓ exactly 2 non terminal. ↘ only 1 terminal	Example:
Example:	$S \to a \mid aSSA$
$S \to SS \mid AS \mid a$	$A \to bA \mid CAAASA$.
$A \to b \mid SA$	

	CNF	GNF
1. $S \to a$	✓	✓
2. $S \to ab$	x	x
3. $S \to SS \mid a$	✓	x
4. $S \to aS \mid bSS \mid c$	x	✓

Conversion of CFG to CNF CFG:

① $S \to ab \Rightarrow S \to C_1C_2$, $C_1 \to a$, $C_2 \to b$ or $S \to AB$, $A \to a$, $B \to b$

② $S \to abc \Rightarrow S \to C_1C_2C_3$

$S \to C_1X$

$X \to C_2C_3$

$C_1 \to a$

$C_2 \to b$

$C_3 \to c$

③ $S \to aS \mid bCA \mid bB$

$A \to abc \mid e$

$B \to Aa \mid b$

$S \to C_1S \mid C_2X \mid C_2B$

$X \to C_3A$

$A \to C_1Y \mid e$

$Y \to C_2C_3$

$B \to AC_1 \mid b$

$C_1 \to a$

$C_2 \to b$

$C_3 \to c$

$X \to$ ⌣

if in CNF ✓

if not ⇒ make every symbol as non terminal then divide into 2 length

CFG ⟹ Simplified ⟹ CNF CFG

$X \longrightarrow \square$ ...

if terminal ⇒ make non-terminal2,
if non-terminal, substitute with productions

Conversion of CFG to GNF CFG?

CFG ⟹ Simplified CFG ⟹ non left Recursive CFG ⟹ GNF CFG

1. $S \to ab \to S \to aC_1$, $C_1 \to b$

2. $S \to abc \to S \to aC_1C_2$, $C_1 \to b$, $C_2 \to c$

3. $S \to Aab \mid baB \Rightarrow S \to fC_1C_2 \mid bC_1B$
   $A \to f$ × Useless.
   $B \to gb$ $B \to gC_3$
   $C_1 \to a$
   $C_2 \to b$
   $C_3 \to c$

→ How long does the derivation takes place for deriving n length string using GNF CFG?

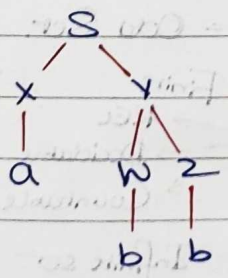
1. $n = 1$, $w = a$

   S
   ↓
   a = 1 Step

2. $n = 2$, $w = aa$

   S
   ↓
   AB = 3 Steps
   ↓
   aB
   ↓
   ab

3. $n = 3$, $w = abb$



S
↓
XY
↓
aY
↓
aWZ
↓
abZ
↓
abb

= 5 Steps
= 5 non leaf nodes
= 5 Substitutions.

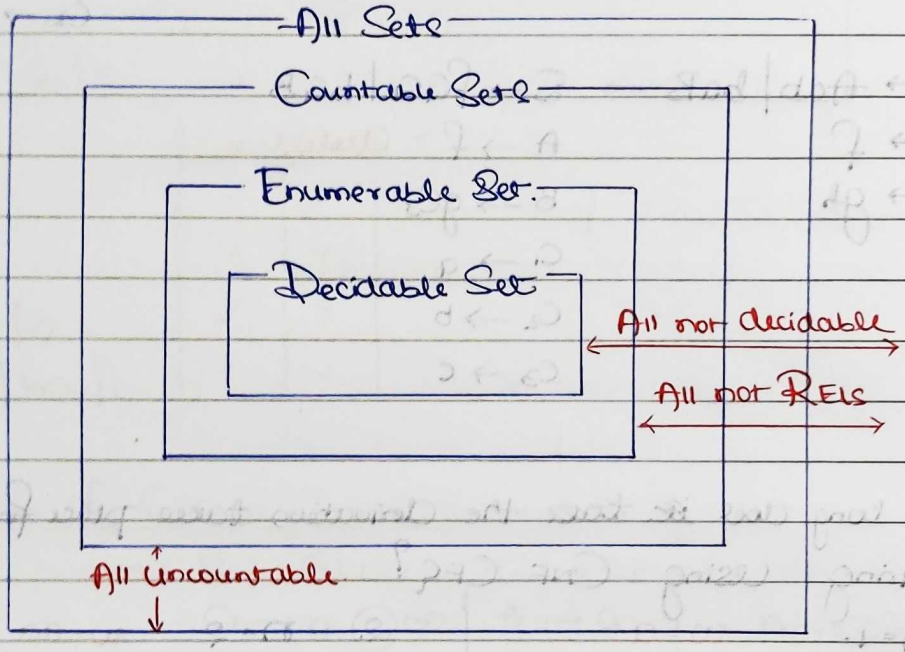
* n length String → (2n-1) steps using GNF CFG

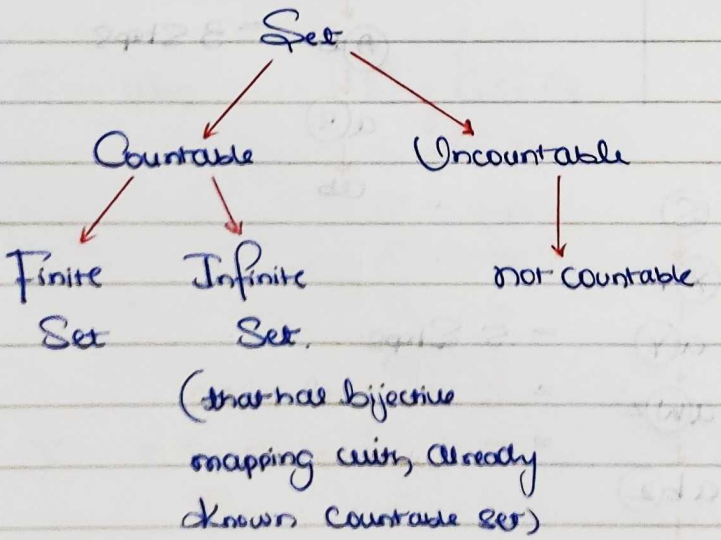
→ How long the derivation take place for deriving n length string using GNF CFG?

1. $n=1 \rightarrow$ # steps = 1
2. $n=2 \rightarrow$ # steps = 2
3. $n=3 \rightarrow$ # steps = 3

$n=1$	$n=2$	$n=3$
S	S	S
↓	↓	↓
a	aX	aXY
	↓	↓
	aa	aaY
		↓
		aaa
$w=a$	$w=aa$	$w=aaa$

* For n length String ↓ 'n' steps using GNF CFG





N = natural Set
Z = integer set
2N = Even Set
2N+1 = Odd Set.

Σ → Finite, REL, Decidable, Countable {Σ = Set of finite no. of symbols}

Σ^* → Infinite set, Regular set, Decidable, Countable, REL {Σ^* = Set of all Strings}

2^{Σ^*} → Infinite Set, not countable {2^{Σ^*} Set of all languages}

		Regular Set.	DCFLs	CFLs	Recurs. Lang.	RELs
1.	$L_1 \cup L_2$	Regular	CFL	CFL	Recursive	REL
2.	$L_1 \cap L_2$	Regular	CSL	CSL	Recursive set	REL
3.	$\overline{L}$	Regular	DCFL	CSL	Recursive	either Recursive or not REL
4.	$L_1 - L_2$	Regular	CSL	CSL	Recursive	Any.

Rice Theorem: Every non trivial property is undecidable.

non trivial Property ← $L = \{m \mid m \text{ is } T_m, m \text{ accepts Regular}\}$ → not REL

$\overline{L} = \{m \mid m \text{ is } T_m, m \text{ accepts not regular}\}$ → not REL

Trivial ← $L = \{T_m \mid T_m \text{ accepts REL}\}$.

$= \text{Set of all } T_m$'s

$\overline{L} = \{T_m \mid T_m \text{ accepts not REL}\}$

$= \phi$