Regular Languages & FA → without O/p → DFA, NFA, ε-NFA; with O/p → Mealy, Moore machine

⇒ FA Theory :-

1) FA is a 5-tuple machine.

$M(Q, \Sigma, \delta, q_0, F)$ — Q: finite & non-empty set of states; Σ: alphabet; δ: transition fu"; q_0: initial state; F: set of final state

$|Q| \geq 1$; $|\Sigma| \geq 1$; $|F| \geq 0$; $q_0 \in Q$; $F \subseteq Q$

2) If there's no final state → Empty lang accepted
$F = \emptyset \Rightarrow L = \emptyset$ (in both DFA & NFA)
$F = Q \Rightarrow L = \Sigma^*$ (only in DFA)

3) δ (Transition function)
DFA :- $\delta_{Q \times \Sigma} = Q$ (exactly 1 trans)
NFA :- $\delta_{Q \times \Sigma \cup \{\epsilon\}} = 2^Q$ (0, 1 or more trans)

4)

	DFA	NFA
1. Choice	✗	✓
0 trans → 2. Dead configuration (dead state not required)	✗	✓
3. ε-move	✗	✓

5) In DFA, no of trans $|\delta| = |Q| * |\Sigma|$

6) Every DFA is NFA & every NFA can be converted to DFA.

7) Extended transition fu" (δ^*)
$\delta^*(q_2, aab)$ means starting from q_2 set of all reachable states after reading aab
DFA: $\delta^*_{Q \times \Sigma^*} \to Q$ NFA: $\delta^*_{Q \times \Sigma^*} \to 2^Q$

8) For DFA :-
- $\delta^*(q, \epsilon) = q$
- $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$

9) Right invariance property (for both NFA & DFA)
$\delta^*(q, x) = \delta^*(q, y)$
$\Rightarrow \delta^*(q, xz) = \delta^*(q, yz)$

10) Lang accepted by a machine :-
DFA :- $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$
NFA :- $L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$
i.e. reaches atleast 1 final state.

11) FA theorems :-

ⓐ complementary machine theorem
If we complement machine (interchange final & non-final states) ⇔ it accepts $\bar{L}$.
$L(\bar{M}) = \bar{L(M)}$ (only for DFA)

ⓑ Finite lang → Regular lang (Regular ⊃ Finite)

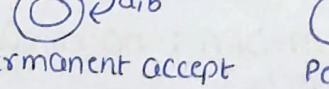
ⓒ Pumping Lemma (if L is CFL → L satisfies PL for CFL)
L is regular → L satisfies P.Lemma for RL.
L doesn't satisfies P.Lemma → L is not regular

ⓓ Myhill Nerode theorem
L is regular ⇔ No. of equivalence class is finite
No. of equi-class = No. of states in m-DFA

ⓔ Kleen's theorem
L is regular ⇔ FA exists
∃ DFA ; ∃ NFA with 1 final state
∃ RE ; ∃ RG ; ∃ RL RG, ∃ LL RG

⇒ FA Design :-

1) m-DFA is unique.
2) m-NFA is not unique but no. of states in m-NFA is unique.
3) DFA, NFA are not unique.
4) For an infinite lang
n(min FA or m-DFA) ≥ $|W_{min}| + 1$ (min string)
5) m-DFA design :-
- accept all minimal string in L
- fill up all missing arrows.

6) 

Permanent accept — Permanent reject (Trap state)

7)

m-DFA		Permanent Accept	Trap
Starting with		✓	✓
Ending with		✗	✗
Substring		✓	✗
Mod	(Many cycles)	✗	✗
atleast ≥		✓	✗
exactly =		✗	✓
atmost ≤		✗	✓

8) For negative thing take complement
9) If in finite lang maximal string is of length n
- m-DFA contains atleast (n+2) states (trap state must be present)
- m-NFA contains atleast (n+1) states

10) containing atleast 1 'a' & atleast 1 'b' only when both are atleast there's no trap else trap state present.

11) "no of a ÷ by 2 (mod2) & no. of b ÷ by 3 (mod3)" contains 2*3 = 6 states

12) Always try to get m-NFA from RE instead of converting from m-DFA.

NOTE: By default FA is NFA
minimal FA → m-NFA

13) If initial state is final state then ∈ is accepted.

14) If finite lang loop is not present

15) If there's AP, in m-DFA cycle is present at common difference.

16) FA fails to accept languages in which comparison exist blw symbols

17) For 1 symbol language, if there is no common difference blw strings generated FA does not exist.

---

* NO. of states in m-DFA :-

x mod n (on dividing by n gives x as remainder)

n is odd → n states

n is even → not power of 2: $n = 2^k * m$ (m odd) → k+m states

power of 2: $n = 2^k$ → k+1 states

* Constructing m-DFA :- no. divisible by 3
remainder → 0, 1, 2

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Minimize this table using partition algo.

1) m-DFA that accepts set of all strings containing a substring of length N requires (N+1) states.

2) m-DFA that accepts all strings of length exactly n → (n+2) states
- strings of length at least n → (n+1) state
- strings of length atmost n → (n+2) states
- length of string ÷ by n → n states
- length of string not ÷ by n → n states

3) Product automata :- final state depends on it
no. of a's exactly 4, and b's atmost 3
5 state ← excluding dead state → 4 state
= 5*4 = 20+1 = 21 states
Add 1 if either of them has dead state.

same for OR case. only final states are changed.

4) m-DFA that accepts set of all strings ending with particular n length string requires (n+1) states

5) nth bit is 'a' from LHS
→ (n+2) states in m-DFA (1 trap state)

6) nth bit is 'a' from RHS (make required states final & not useful as initial)
→ 2^n states in m-DFA
→ (n+1) states in NFA

NOTE: $L^c = \Sigma^* - L$
$L \cap L^c = \phi$; $L \cup L^c = \Sigma^*$

7) If L is regular, L^c is also regular

8) For starting conditions dead state required not for ending condition.

9) $L = \{a^{kn+k_2}\}$
if n ≥ 0 → case1: $k_2 < k_1$ ⇒ k_1 states in m-DFA & m-NFA
case2: $k_2 \geq k_1$ ⇒ (k_2+1) states
if n ≥ 1 or 2 or 3...
⇒ |Wmin| + 1 states

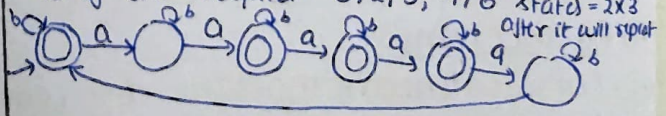
NOTE:- $L = \{a^m b^n \mid m,n \geq 0 \ \&\& \ m+n = 10\}$
L is regular because it's finite lang

→ TO check whether a lang is regular or not :-

L
- L is finite ⇓ Regular
- L is infinite
  - single alphabet → if AP series then Regular
  - more than 1 alpha → if no comparison then Regular

* NO. of a's ÷ by 2 or 3

NO. of a's accepted = 0, 2, 3, 4, 6 ← max no. of states = 2x3 after it will repeat



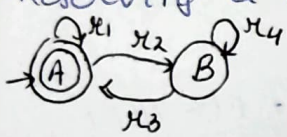
- If condition is on same symbol and it collectively forms an AP then no. of states in m-DFA = common difference (check for pattern, if not)
- If doesn't form AP then no. of states = m x n (cartesian product)

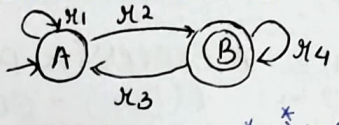
⇒ Regular Expression :- → it not unique for a language

1) Machine → RE → through guidelines
   ↳ M → L → RE

→ Through guidelines :-

- DOesn't matter if it's DFA or NFA.
- Delete trap state, non reachable state
- Delete any state through which there is no path to some accepting state
- collapse any final states in series to a single final state if together they behave like a permanent accept state.
- If multiple final states say A and B then RE for M is $R_A + R_B$.
- RE for any state is the longest path from the starting state to that state
- Delete states except starting & final & write whatever happens through them manually.
- Resolving a two state machine

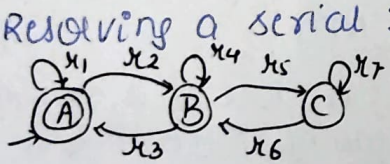




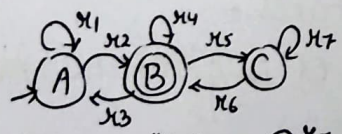
$RE_A = (R_1 + R_2 R_4^* R_3)^*$ or $RE_B = (R_1 + R_2 R_4^* R_3)^* R_2 R_4^*$

$RE_B = R_1^* R_2 (R_4 + R_3 R_1^* R_2)^*$

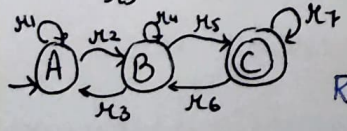
- Resolving a serial 3 state machine



$RE_A = (R_1 + R_2 X^* R_3)^*$ where $X = R_4 + R_5 R_7^* R_6$



$RE_B = R_1^* R_2 (R_4 + R_3 R_1^* R_2 + R_5 R_7^* R_6)^*$



$RE_C = RE_B \cdot R_5 R_7^*$

$RE_B = R_1^* R_2 (R_4 + R_3 R_1^* R_2 + R_5 R_7^* R_6)^*$

2) Properties of RE :-

(a) Commutative :- Let R, S, T be RE

$R + S = S + R$ Ex :- $a^* + b^* = b^* + a^*$

$R \cdot S \neq S \cdot R$ may or may not $a^* b^* \neq b^* a^*$

(b) Associative :- $R + (S + T) = (R + S) + T$

$R \cdot (S \cdot T) = (R \cdot S) \cdot T$

Ex :- $a + (b^* + c^*) = (a + b^*) + c^*$

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

(c) Distributive :-

$R(S + T) = RS + RT$; $(S + T)R = SR + TR$

$R + (S \cdot T) \neq (R + S)(R + T)$

(d) Identity :-

$R + \emptyset = R = \emptyset + R$

$R \cdot \epsilon = R = \epsilon \cdot R$

$R \cdot \emptyset = \emptyset = \emptyset \cdot R$

$R + \epsilon \neq R$

$R + \epsilon = R$ iff $\epsilon \in L(R)$

NOTE :-
$\epsilon^* = \epsilon$, $\epsilon^+ = \epsilon$
$\emptyset^* = \epsilon$, $\emptyset^+ = \emptyset$

(e) Properties of *, + & . :-

- $R_1 + R_1 = R_1$
- $R_1 \cdot R_1 \neq R_1$
- $R^* + R^* = R^*$
- $R^* \cdot R^* = R^*$
- $R^* \cdot R^+ = R^+$
- $(R^*)^* = (R^*)^+ = (R^+)^* = R^*$
- $(R^+)^+ = R^+$
- $\epsilon + R R^* = R^*$, $R R^* = R^+$
- $\epsilon + R + R R R^* = R^*$
- $(PQ)^* P = P(QP)^*$
- $(R^* + S^*)^* = (R^* S^*)^* = (R + S)^*$
- $= (R^* + S)^* = (R + S^*)^* = ((R + S)^*)^* =$
- $(S^* R^*)^* = (R S^* + S R^*)^* = ((R + S)^+)^*$
- $(R^* S^* + R^* S^*)^* = ((R + S)^*)^+ = (R^* S)^* R^*$
  $= S^* (R S^*)^*$

(Just check R & S are separately available inside ( )*

- $R + X = R$ where $L(X) \subseteq L(R)$

⇒ Regular Grammar :- all production must be either left linear or right linear

$V \rightarrow VT^* / T^*$ Left linear grammar

or

$V \rightarrow T^* V / T^*$ Right linear grammar

1) Solving Regular grammar :-

$G \rightarrow$ RLRG $\rightarrow$ FA $\rightarrow$ Lang

$G \rightarrow$ LLRG $\rightarrow$ Substitution
CFL

2) $\emptyset$: $S \rightarrow A$

$\{\epsilon\}$: $S \rightarrow \epsilon$

$\{a\}$: $S \rightarrow a$

$\{a, b\}$: $S \rightarrow a/b$

a^* : $S \rightarrow aS/\epsilon$
$\rightarrow Sa/\epsilon$
$\rightarrow a/Ss/\epsilon$

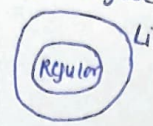
$(ab)^*$: $S \rightarrow abS/\epsilon$
$\rightarrow Sab/\epsilon$

- $S \rightarrow L$ to make L^* add SS/ϵ
  ↳ $S \rightarrow L/SS/\epsilon$
- add SS to make L^+
  $S \rightarrow L/SS$
- $(a+b)^*$
  $S \rightarrow aS/bS/\epsilon$
  $S \rightarrow Sa/Sb/\epsilon$
  $S \rightarrow a/b/SS/\epsilon$
- $(a+b)^+$
  $S \rightarrow aS/bS/a/b$

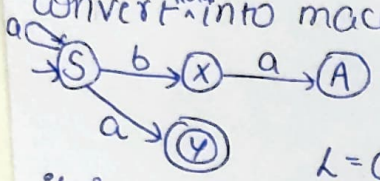
- $S \to xS / yS / z$
  $(x+y)^*S \Rightarrow (x+y)^*z$
- $S \to xS / Sy / \epsilon$
  x^*y^*
- $S \to \underbrace{xS / Sy}_{x^*Sy^*} / z$
  $\Rightarrow x^*zy^*$

$L = L_1 + L_2$
$S \to S_1 / S_2$
$L = L_1 . L_2$
$S \to S_1 . S_2$

NOTE: Linear grammar → at most 1 var at RHS
$S \to aS/Sb/a$
$S \to aSb/ab$
are linear but not regular



* $S \to aS/baA/a$
convert RLRG into machine & find lang.



(variable is a state)

$L = a^*a = a+$

if in production ϵ is there, S will also be final state.

* Two grammars are equivalent if they produces same language.
(Solve it by contradiction, find a string present in 1 grammar not in other).

NOTE: In NFA just ensure all valid strings are accepted.

⇒ ϵ NFA to NFA ;

- Start making NFA with equal no of states of ϵ-NFA
- Make initial state same of ϵNFA
- Make final same.
  From any state on reading ϵ if it reaches final, make it final.
- DO transitions.

⇒ ϵ NFA to DFA : (minimal no guarantee)

- Take initial state of ϵNFA and all states reached from initial on reading ϵ as single initial state of DFA.
- Proceed as converting NFA to DFA.
- Use dead state when no transition
- Make all states having final state of ϵNFA as final state of DFA.

⇒ NFA to DFA conversion

no guarantee for min-DFA

TC = O(1) Bc O(2^n) wc

(Subset construction also)

- Start with initial state of NFA & do transition for each symbol in Σ.
- If new state comes then proceed else stop
- NFA final states wherever present in DFA, make it final
- If there is no transition for a symbol in NFA, goto dead state in DFA. On dead state make self loop.

* If NFA contains n-states then equivalent DFA contains maximum 2^n states.
[1 to 2^n] states

NOTE:- Don't use any conversion algo in exam. Just verify the lang accepted by both is same in order to prove them equivalent.

NOTE: RE for set of all strings of even length
$= ((0+1)^2)^*$
odd length string $= ((0+1)^2)^*(0+1)$

* 2 consecutive 1's not allowed
$RE = (0+10)^*(1+\epsilon)$

* 2 consecutive 0's & 1's not allowed
$RE = (1+\epsilon)(01)^*(0+\epsilon)$

* Expressive power
E(DFA) = E(NFA) = E(ϵ-NFA)

NOTE: DFA ≡ NFA
DPDA ≢ NPDA
DTM ≡ NTM

Multitape TM = Single tape TM

* ϵ closure: of a state q is, starting from q by reading only ϵ, set of all reachable states.

→ Minimization of DFA (Partition Algo)

NOTE:- M-DFA is unique.

- Eliminate inaccessable states.
- Apply algo
  First make 2 groups = finals & non finals
  its zero equivalence
  $\pi_0 = (q_0, q_1, q_4)(q_3)$
  if q_0 on Σ ⇒ state in same grp in π_0
  q_1 on Σ ⇒ then keep in same grp else create new grp of q_1
  $\pi_1 = (q_0, q_4)(q_1)(q_3)$ ← same so stop
  $\pi_2 = (q_0, q_4)(q_1)(q_3)$ ←
  $q_0 = q_4$
  Remove q_4 and make $q_4 \to q_0$

NOTE: while doing π_i use π_{i-1}

⇒ FA with output :- (Transducer)

Mealy m/c — O/p at transaction

Moore m/c — O/p at state

1) 6 tuples machine $(Q, \Sigma, \delta, q_0, \Delta, \lambda)$
$\Delta \rightarrow$ O/p alpha. $\lambda \rightarrow$ O/p fu^n

2) Mealy: $\lambda: Q \times \Sigma \rightarrow \Delta$ Moore $\lambda: Q \rightarrow \Delta$

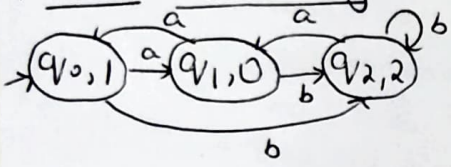
3) NO final state

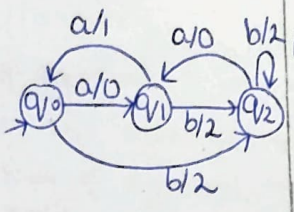
4) BOTH are deterministic (trans for each i/p from each state)

5) While converting mealy of n states & m O/p symbol to moore machine max no. of states possible = $n * m$

6) For a given n length i/p, O/p length
Mealy → n bits Moore → (n+1) bits

→ Moore to Mealy :-





→ Mealy to Moore :-

If initial state is splitted into many states, take any one as initial.

	a	b
→S1	S2 2	S1 2
S2	S2 1	S1 0

Make these as states

take any as initial

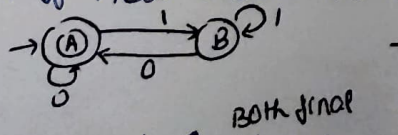
	O/p	a	b
→S10	0	S22	S12
S12	2	S22	S12
S21	1	S21	S10
S22	2	S21	S10

7) Minimal Moore & min-Mealy machine are unique

⇒ Equivalence problem :-

Two DFA are equal if they accept same language.

- Start with initial state of both DFA
- Transition for each symbol. Both should be accepted or both should not
- If new state comes then repeat.



Both final

	0	1
(A,C)	(A,D)	(B,E)
(A,D)	(A,D)	(B,E)
(B,E)	(A,C)	(B,E)

Both non final

→ Finiteness problem :-

- Remove inaccessable states & states from which we can't reach final state
- If loop or cycle present → infinite lang.

→ Emptiness problem :-

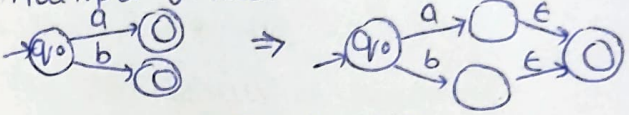
- Eliminate all inaccessable states
- If atleast 1 final state → Non-empty lang

NOTE: Palindrome lang over more than 1 symbol are not regular whereas over 1 symbol its regular.

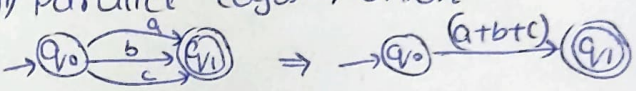
⇒ Finite Automata to RE :-

★ State elimination method :-

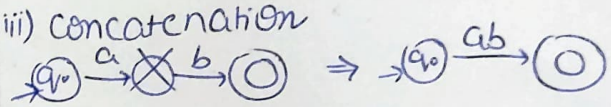
i) Multiple finals → single final



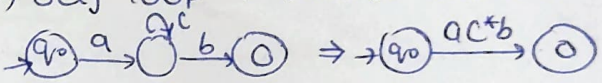
ii) Parallel edges ⇒ Union



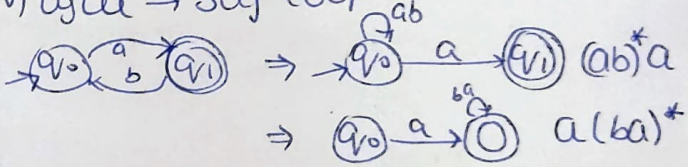
iii) Concatenation



iv) Self loop → Kleene closure



v) Cycle → Self loop



NOTE:- $a(b+a) + b(b+a) = (a+b)(b+a)$

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

$(a+b)c + d(a+b) \neq (a+b)(c+d)$

common is taken only from starting or ending.

★ Arden's Method :-

- write state eq^n for each state & solve
State eq^n of q means if q is final what it will accept.
- can't apply for ε-NFA.
- If P & Q are two RE & P does not contain ε then
$R = Q + RP \Rightarrow R = QP^*$
- If P contains ε then we have infinite number of solutions.

Standard RL and NRL

1. $\{a^n b\}$ RL
2. $\{a^n b^m \mid n, m \geq 1\}$ RL
3. $\{a^n b^m \mid n \leq 7, m \leq 1\}$ RL
4. $\{a^n b^m \mid n = 2 \text{ or } 4\}$ RL
5. $\{a^n b^m \mid n \geq 1, m \geq 2\}$ RL
6. $\{a^n b^m \mid n \times m \geq 2\}$ RL
7. $\{a^n b^m \mid n \times m \leq 2\}$ RL
8. $\{a^n b^m \mid n + m = 6\}$ RL
9. $\{a^n b^m \mid n + m \geq 6\}$ RL
10. $\{a^n b^m \mid n + m \leq 6\}$ RL
11. $\{a^n b^m \mid n - m = 5\}$ NRL
12. $\{a^n b^m \mid n = m + 5\}$ NRL
13. $\{a^n b^m \mid 100 - n = m\}$ RL
14. $\{a^n b^m \mid n = 2m\}$ NRL
15. $\{a^n b^m \mid n > m\}$ NRL
16. $\{a^n b^m \mid n \geq 2m\}$ NRL
17. $\{a^n b^m \mid n/m = 5\}$ NRL
18. $\{a^n b^m \mid n^2 = m\}$ NRL
19. $\{a^p \mid p \to \text{Prime}\}$ NRL
20. $\{a^n b^n c^m \mid n \leq 7, m \geq 2\}$ RL
21. $\{a^n b^n c^m \mid n \geq 1, m \leq 2\}$ NRL
22. $\{a^n b^n c^m \mid n \geq 1, m \geq 1\}$ NRL
23. $\{a^n b^n c^m \mid n = m\}$ NRL
24. $\{a^n b^m c^p \mid n = m + p\}$ NRL
25. $\{a^n b^m c^p \mid n = m \times p\}$ NRL
26. $\{a^n b^m c^p \mid n = 2m + p\}$ NRL
27. $\{a^n b^n c^n \mid n \geq 1\}$ NRL
28. $\{a^p \mid p \text{ is prime}, |p| \leq 30\}$ RL
29. $\{a^n b^n c^n \mid n \leq 10^7\}$ RL
30. $\{a^n a^n \mid n \geq 1\}$ RL

- $L = \{a^n b^m \mid (n+m) \text{ is even}\}$ RL $(aa)^*(bb)^* + 1(aa)^*(bb)^*$
- $L = \{1, 2, 4, 8, \ldots\}$
  written in binary :- RL 0^*10^*
  written in unary :- NRL

31. $\{w.w \mid w \in (a,b)^*\}$ NRL
32. $\{w\, w^R \mid w \in (a,b)^*\}$ NRL
33. $\{w(w^R)^* \mid w \in (a,b)^*\}$ RL $= (a+b)^*$
34. $\{w \# w^R \mid w \in (a,b)^*\}$ NRL
35. $\{w \mid w \in (a,b)^*\}$ RL
36. $\{w x w^R \mid w, x \in (a,b)^*\}$ RL
37. $\{w x w^R \mid w, x \in (a,b)^+\}$ RL $a\Sigma^+a + b\Sigma^+b$
38. $\{x\, w w^R \mid w, x \in (a,b)^*\}$ RL
39. $\{x\, w w^R \mid w, x \in (a,b)^+\}$ NRL
40. $\{w w^R x \mid w, x \in (a,b)^*\}$ RL $= (a+b)^*$
41. $\{w w^R x \mid w, x \in (a,b)^*\}$ NRL
42. $\{x \subseteq y \mid x, y \in (a,b)^*\}$ RL
43. $\{w x w \mid w, x \in (a,b)^+\}$ NRL
44. $\{w x w \mid w, x \in (a,b)^*\}$ RL
45. $\{w x w \mid w \in (a,b)^* \; x \in (a,b)\}$ NRL
46. $\{w x w \mid w, x \in (a,b)^+, |w| \leq 3\}$ RL
47. $\{a^n b^n c^m d^m \mid n, m \geq 1\}$ NRL
48. $\{a^n b^m c^p d^q \mid n, m, p, q \geq 1\}$ RL
49. $\{a^n b^m c^p d^q \mid n, m \geq 3, pq \leq 10\}$ RL
50. $\{a^n c\, b^n \mid n \geq 1\}$ NRL
51. $\{a^{2n} c\, b^{3m} \mid n, m \geq 1\}$ RL
52. $\{awa \mid w \in (a,b)^*\}$ RL
53. $\{waw \mid w \in (a,b)^*\}$ NRL
54. $\{ww w^R \mid w \in (a,b)^*\}$ NRL
55. $\{ab b^n \mid n \geq 1\}$ RL
56. $\{w \mid w \in (a,b)^*, |w| = \text{even}, n_a(w) = \text{odd}\}$ RL
57. $\{a^n b^n \mid n \geq 1, n \neq 99\}$ NRL
58. $\{a^n b^m \mid n < m < 2^n\}$ NRL
59. $\{a^n b^m \mid n > m > 10\}$ NRL
60. $\{a^n b^m \mid n < m < 10\}$ RL
61. $\{a^n b^{2m} \mid 3 < m < 4, n \geq 1\}$ RL
62. $\{a^m b^n c^p \mid m + n + p = 10\}$ RL
63. $\{w_1 \cdot w_2 \mid w_1, w_2 \in (a,b)^*, |w_1| = |w_2|\}$ RL — even length string
64. $\{w \mid n_a(w) \neq n_b(w)\}$ NRL
65. $\{w \mid w \in (a,b)^*\}$ RL — Check condition at each point while scanning it for every prefix of w

- $L = \{a^p \mid p \text{ is prime}\}$ RL $(\epsilon + aa^+)$
- $L = \{a^{mn} \mid m > n \;\&\; n \geq 1\}$ RL (aa^+)
- $L =$ odd / even length palindrome string RL $a(aa)^*, (aa)^*$ on single symbol.
- $L = \{xy \mid x, y \in \{a,b\}^*, \#a(x) = \#b(y)\}$ RL — complement lang
- $L = \{x \& y \mid x, y \in (a,b)^*, \#a(x) = \#b(y)\}$ NRL

• $\{w \mid w \in (ab)^*, n_a(w) \bmod 3 > n_b(w) \bmod 3\}$ RL

• $\{w \mid w \in (a^*b^*), n_a(w) \bmod 3 = n_b(w) \bmod 3\}$ RL

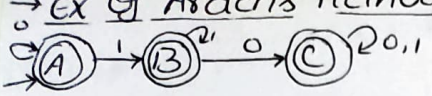
• $\{ww^R \mid w \in (a+b)^*\}$ RL

• $\{w \# w^R \mid w \in (a+b)^*\}$ NRL

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

Scanned with CamScanner

→ Ex of Arden's Method:-



$A = A\cdot 0 + \epsilon$
$B = A\cdot 1 + B\cdot 1$
$C = B\cdot 0 + C\cdot(0+1)$

∵ A, B & C are final states

$RE = A + B + C$

$A = \epsilon\cdot 0^* = 0^*$; $B = A\cdot 1\cdot 1^* = 0^*1^+$

$C = B\cdot 0\cdot(0+1)^* = 0^*1^+0(0+1)^*$

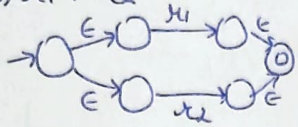
$RE = 0^* + 0^*1^+ + 0^*1^+0(0+1)^* = \Sigma^*$ ← DFA with all state final

⇒ RE TO FA (ϵ-NFA)

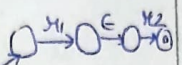
1) ∅ → ○

2) ϵ → ◎

3) $R_1 + R_2$

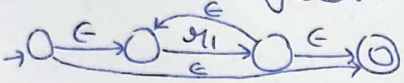


4) $R_1 \cdot R_2$



* If R_1 & R_2 are regular then (R_1+R_2), R^* and $R_1\cdot R_2$ are regular.

5) R_1^*



NOTE:- $R = a^*b^*c^*d^*$

No. of states in m-DFA = no. of letters + 1 = 4+1 = 5

⇒ $L = \{x \mid x \in (0,1)^*, n_{(110)}(x) = n_{(011)}(x)\}$ RL

If these are reverse then regular.

- $n_{011}(x) > n_{110}(x)$ RL
- $n_{000} \geq n_{111}(x)$ NRL

⇒ $L = \{xwywR \mid w,x,y \in (a,b)^+\}$ NRL

$L = \{wxwRy\}$ NRL

$L = \{wxywR\}$ RL

⇒ $L = \{wxwy \mid x,y,w \in (a,b)^* \text{ or } (a,b)^+\}$ RL if after after or before before

$L = \{xwywy\}$

⇒ wxywR RL if after before or before after

xwwRy

→ m-DFA that accepts binary nos ÷ by 5 & start with 1

Make modulo 5 machine & add a new starting state & dead state

	0	1
→ S	D	q_{01}
D	D	D
q_0	q_{00}	q_{01}
q_1	q_{02}	q_{03}
q_2	q_{04}	q_{00}
q_3	q_{01}	q_{02}
q_4	q_{03}	q_{04}

→ ÷ by 5 & ends with 1 condition at end, no dead state required

⇒ 10 states

5, 15, 25, 35, ...

⇒ Pumping Lema:

- used to prove a lang non regular
- L is regular → satisfies pumping lema for regular lang
- does not satisfy → L is not regular
- Based on Pigeon hole principle & uses proof by contradiction.
- A FA without loop/cycle with n states can accept max (n-1) length strings.
- Pumping lema is for infinite lang.

* TO prove L is NON regular

- Assume L is regular. Then there exists FA for L and n is no. of state
- select string $w \in L$ such that $|w| \geq n$ Then loop or cycle exist.
- Divide $w = xyz$ such that ○ —x→ ◎ —y→ ◎ —z→ ○ $|y| \geq 1$ and $|xy| \leq n$
- Then $xy^iz \in L \ \forall\ i = \{0,1,2,\ldots\}$ otherwise assumption is wrong.

NOTE: y is choosen from first n symbol of string w.

* For any regular lang L, there exists an integer $n \geq 1$ such that for all $w \in L$ with $|w| \geq n$ there exists $w = xyz$ and
  - $|xy| \leq n$
  - $|y| \geq 1$
  - $\forall i \geq 0$; $xy^iz \in L$

  n → Pumping length

* Pumping len is not unique but min-pumping len is unique for a RL
* Every string w of length ≥ Min PL $\in L$ must have at least 1 or more bits pumpable.

- Every no ≥ #states in mDFA will work as pumping length(n)
- Pumping len is property of lang not DFA.
- $\forall_{\text{Reg lang}}\ \exists_{n\geq 1}\ \forall_{w\in L,\ |w|\geq n}\ \exists_{x,y,z}\ (w = xyz \wedge |y|\geq 1 \wedge |xy|\leq n \wedge xy^iz \in L\ \forall i\geq 0)$
- Min Pumping len(MPL) ≤ #states in mDFA
- MPL > $|w_{min}|$
- If n is pumping len then any no ≥ n is pumping length.
- MPL(Σ^*) = 1
- MPL(∅) = 1
- MPL(finite lang) = $|w_{max}| + 1$
- MPL ≥ 1
- If n is MPL, Pumping len = m ; m ≥ n
- MPL(ϵ) = 1
- If $L = (L_1 \cup L_2)$ Look it on lang L dont focus only on L_1 or L_2

⇒ Myhill Nerode theorem :-

1) L is Regular language ⟺ Finite no. of equivalence classes
2) NO. of equivalence class = NO. of states in m-DFA
3) $E_i \cap E_j = \phi \quad \forall i, j \; \& \; i \neq j$
4) $E_1 \cup E_2 \cup \ldots = \Sigma^*$
5) TO find equiv classes, find Regular expression for each state.
6) If L is not regular ⟺ infinite no. of equ. classes.
7) 2 different lang can have same no. of equi classes, also same set of eq. classes

⇒ closure properties of RL :-

1) **Union** :- Regular ∪ Regular = Regular
   Regular ∪ Irregular = May or May not RL
   - If $L_1 \cup L_2$ is RL then L_1 & L_2 may or not RL
   - If $L_1 \cup L_2$ is RL & L_1 is finite then L_2 is RL
2) **Intersection** :- RL ∩ RL = RL
   - If $L_1 \cap L_2$ is RL then L_1 & L_2 may or not RL.
3) **complementation** :
   $(RL)^c = RL$ $\quad$ $(NRL)^c = NRL$
4) **Difference** : $RL_1 - RL_2 = RL$
   - $\Sigma^* - L = L^c$
   - $L - \Sigma^* = \phi$
   - $L_1 - L_2 = L_1 \cap L_2^c$
   - $L_1 - L_2 = L_1 - (L_1 \cap L_2)$
5) **Subset** :
   subset of RL may or may not be RL
   subset of NRL
6) **concatination** : $RL_1 \cdot RL_2 = RL$
   - If $L_1 . L_2$ is RL & L_1 is RL then L_2 may or may not be RL.
7) **Reversal operator** :
   L is RL ⟷ L^R is RL
   L is NRL ⟷ L^R is NRL

* Reversing a DFA :-
reverse each edge direction & interchange initial and final states.
If there are more than 1 final states connect them using ∈ to a new initial state.

DFA $\xrightarrow{\text{Reverse}}$ DFA/NFA/∈-NFA

8) **Kleene closure** :-
   L is RL → L^* is RL
   L is NRL → L^* may or not be RL
9) **Quotient Operation** :
   $L = L_1/L_2 = \{x \mid \forall xy \in L_1 \; \& \& \; \exists y \in L_2\}$
   - cancellation done from RHS only if unable to cancel, ans is ∅.

→ {a, ab} {ϵ, b, aa} = {a, ab} try 1 by 1 num to every demo.

→ $a/a = \epsilon$ → $a^*/b^* = a^*$ (for ϵ independent)

- $RL/RL = RL$
- $RL/NRL = RL$

→ $\frac{\epsilon}{a} = \phi$, $\frac{\epsilon}{\phi} = \phi$, $\frac{\phi}{\text{kuch bhi}} = \phi$, $\frac{\text{kuch}}{\phi} = \phi$

→ $\Sigma^* = \frac{\Sigma^*}{\Sigma^*} = \frac{\Sigma^+}{\Sigma^+} = \frac{\Sigma^*}{\Sigma^+} = \frac{\Sigma^+}{\Sigma^*}$

→ If L_2 has ϵ then L_1/L_2 contains atleast L_1.

→ If L_1 is not empty then $\frac{\Sigma^*}{L_1} = \Sigma^*$

→ If L_1 is not empty then $\frac{L_1}{\Sigma^*}$ = all prefixes of L_1

10) **Prefix, suffix and substrings** :
   - Make all states final (except dead), it will accept all prefixes of L.
   - Make all states initial, it will accept all suffices of L.
   - Make all states initial as well as final, it will accept all substrings of L.
   - L is RL ⇒ Prefix(L), Suffix(L) & Substring(L) r also RL.
11) **Substitution** :
   mapping from Σ to $P(\Delta^*)$ (RL)
   - $s(ab) = s(a) \cdot s(b)$
   - $s(a+b) = s(a) + s(b)$
   - $s(a^*) = [s(a)]^*$
   - $s(\epsilon) = \epsilon$ $\quad$ - $s(\phi) = \phi$
   - RL is closed under substitution

NOTE :- Δ → symbol
Δ^* → string , $P(\Delta^*)$ → RL

NOTE : Suffix(L) = Reverse(prefix(Reverse(L)))

12) Homomorphism :- RL closed.
- mapping of $\Sigma \to \Delta^*$
- $h(\epsilon) = \epsilon$
- $h(\phi) = \phi$
- $h(a+b) = h(a) + h(b)$
- $h(ab) = h(a) \cdot h(b)$
- $h(a^*) = [h(a)]^*$

* L is RL → h(L) & $h^{-1}(L)$ are RL

13) Inverse Homomorphism :-
$h^{-1}(w) = \{x \mid h(x) = w\}$
- If can't find for entire lang, find for strings which are known.

14) If R is regular then $\frac{1}{2}R$, $\frac{1}{3}R$, ... are also regular.

closure Table

	REG	DCFL	CFL	CSL	REC	RE
U KICC						
Union	✓	X	✓	✓	✓	✓
Intersection	✓	X	X	✓	✓	✓
Complement	✓	✓	X	✓	✓	X
Concatination	✓	X	✓	✓	✓	✓
Kleene closure	✓	X	✓	✓	✓	✓
All HIS R						
Reverse	✓	X	✓	✓	✓	✓
Homomorphism	✓	X	✓	X	X	✓
Inverse Homomorphism	✓	✓	✓	✓	✓	✓
Substitution	✓	X	✓	✓	X	✓
All						
L ∪ RL	✓	✓	✓	✓	✓	✓
L ∩ RL	✓	✓	✓	✓	✓	✓
L − RL	✓	✓	✓	✓	✓	✓
Preety Queens C HMM						
Prefix	✓	✓	✓			
Quotient (suffix, substring)	✓	X	✓			
Cycle	✓	X	✓			
Min, Max	✓	X	X			
Half	✓	X	X			

- init is prefix

* subset, superset, infinite ∪, infinite ∩, infinite − (infinite concatination) are not closed for any language.
* All are closed under finite subset.
* U KICC HIS R Preety Queens C HMM

* DCFL ∪ DCFL is X so Push both upword CFL ∪ CFL = CFL
Thus, union of two DCFL is CFL.
* If operation not present in table then push up
Ex: DCFL · RL = DCFL · DCFL is cross
CFL · CFL = CFL
* convert secondary opr to primary
- $L_1 - L_2 = L_1 \cap \overline{L_2}$
- $L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1) = (L_1 \cap L_2^c) \cup (L_2 \cap L_1^c)$
- $L_1 \uparrow L_2 = (L_1 \cap L_2)^c = L_1^c \cup L_2^c$
- $L_1 \downarrow L_2 = (L_1 \cup L_2)^c = L_1^c \cap L_2^c$

Ex:- DCFL − CFL = DCFL ∩ $\overline{CFL}$
= DCFL ∩ $\overline{CSL}$ = DCFL ∩ CSL
= CSL ∩ CSL = CSL

* simplify expressions :
$\phi \cup L = L$; $\phi \cap L = \phi$; $L \cap \overline{L} = \phi$
$\Sigma^* \cup L = \Sigma^*$; $\Sigma^* \cap L = L$; $L \cup \overline{L} = \Sigma^*$
$L_1 \cup (L_1 \cap L_2) = L_1$; $L_1 \cap (L_1 \cup L_2) = L_1$
* Always simplify before applying closure properties.
* when actual lang is given, instead of type only, do operation & check type of language.
* contra positive :-
L_1 & L_2 are RL → $L_1 \cup L_2$ is RL
$L_1 \cup L_2$ is NOT RL → at least 1 is NRL

→ closure properties of finite lang.
- Fin ∪ Fin = Fin
- Fin ∩ Fin = Fin
- Fin · Fin = Fin
- $(Fin)^c$ = Infinite
- subset of fin = fin
- superset of fin = fin or ∞
- $(Fin)^*$ = inf
- $(Fin)^R$ = Fin
- h(Fin) = Fin
- $h^{-1}(Fin)$ = Fin or inf

→ closure properties of ∞ lang.
- ∞ ∪ ∞ = ∞
- ∞ ∩ ∞ = fin or ∞
- ∞ · ∞ = ∞
- h(∞) = fin or ∞
- subset of ∞ = ∞ or fin
- superset of ∞ = ∞
- $(\infty)^c$ = Fin or ∞
- $(\infty)^*$ = ∞
- $(\infty)^R$ = (∞)
- $h^{-1}(\infty)$ = ∞

→ Deterministic finite automata:-

- L1/L2 FA is L1 automata only but there may be change in final states.
- To know state q_i is final or not, starting from q_i if it accepts any string of L_2 then q_i is final. Check for each state.

Grammar

1) G(V, T, P, S)
"Varanasi thermal Power station"
V: variable (non empty & finite) capital letter
T: terminal (finite) small letter
P: Production (non empty & finite)
S: start state $S \in V$

2) ϵ is neither variable nor terminal.

3) Sentential form: $(V+T)^*$

4) Sentence: T^*

5) Two grammars are equivalent if they represent same language.

6) Ex: Derivation $S \to aSb \to aaSbb \to aabb$
Sentential form: S, aSb, aaSbb, aabb
Sentence: aabb

→ Types of grammars:-

Type 0 (REL)	Type 1 (CSL)	Type 2 (CFL)	Type 3 (RL)
unrestricted grammar	context sensitive gr	context free grammar	Regular grammar
$(V+T)^+ \to (V+T)^*$	$(V+T)^+ \to (V+T)^+$ $\alpha \quad \beta$ $\lvert\alpha\rvert \le \lvert\beta\rvert$	$V \to (V+T)^*$	$V \to VT^*/T^*$ Left linear or $V \to T^*V/T^*$ Right linear

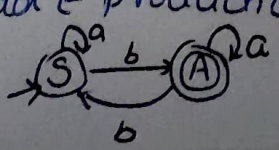
★ $T_3 \subset T_2 \subset T_1 \subset T_0$ (for ϵ free grammar)

Regular Grammar

1) For every left linear grammar, right linear grammar also possible & vice versa

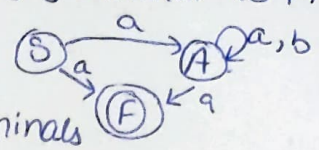
2) Reg lang ↔ Reg grammar ↔ LLG ↔ RLG

→ FA to grammar: (RLG)
V: {states} T: Σ
P: transitions S: initial state
Add ∈ production to final state

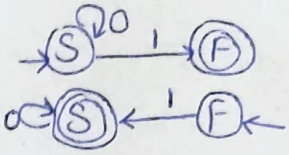
Ex:


$S \to aS/bA$
$A \to aA/bS/\epsilon$

3) Similarly does RLG to FA.
$S \to aA/a$
$A \to aA/bA/a$
Final state for terminals



→ LLG to FA:
Reverse grammar → construct FA → Reverse FA
Ex: $S \to S0/1$
$S \to 0S/1$



→ FA to LLG:
Reverse FA → construct RLG → Reverse grammar

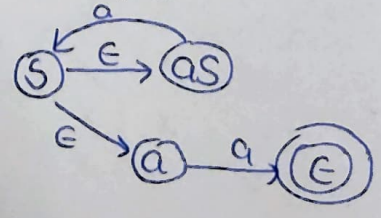
→ Left most derivation: If left most variable is replaced first.
→ Right most derivation: If right most variable is replaced first.
- If LMD possible ↔ RMD possible

* If for atleast 1 string more than 1 derivation tree possible then grammar is ambiguous.
* checking for ambiguity & removing is undecidable. (no algo for it)
* Inherently ambiguous grammar: from which ambiguity can't be removed
* $S \to aS/a$
There is recursion, it will generate infinite number of strings.

→ Regularity Problem:- means checking whether lang generated by given CFG is regular or not.
- Its undecidable problem.

→ RLG to ∈ NFA:

$S \to aS/a$



$aS \xrightarrow{a} S$
$a \xrightarrow{a} \epsilon$
Taking out a from it

* ∈ is final state

→ Simplification of CFG's-

Step1: Remove NULL productions

S→AaB	Remove A→∈ &	S→AaB/aB/Na/a
A→aA/∈	Put A=∈ in	A→aA/a
B→bB/∈	get base conditions	B→bB/b

One by one do for all NULL production

Step2: Remove Unit productions (A→B) (A: variable, B: variable)

S→A	S→A
A→B	A→a
B→a	B→a

Step3: Remove useless variables:
- Remove variables which are not deriving any strings.
  Ex: B→bB it does not generate string as its recursion w/o termination
- Remove variables which r not required (not reachable from start symbol)

* Grammar (CFG, L(G)) → simplified grammar (CFG', L(G'))
- Both lang are same if ∈ ∉ L(G)
- If CFG is ∈ free then CFG ≡ CFG'

→ Normal form of CFG's-

Applying condition on RHS of CFG.

Chomsky (CNF) Normal form: V→VV/T

Greibach (GNF) Normal form: V→TV*

⇒ CFG → CNF:
First simplify then convert.

S→AS/ASS	S→AS/CS
A→SA/aa	C→AA
	A→SA/BB
	B→a

* TO generate n length string from CNF grammar, total no. of production required is (2n-1) (No. of steps in derivation)

⇒ CFG → GNF:
First simplify then convert.

S→(AB/aB)	S→aAB/bBB/bB/aB
A→aA/bB/b	A→aA/bB/b
B→b	B→b

* TO generate n length string from GNF grammar, total no. of production required is 'n'

* For every ∈ free CFG, equivalent CNF and GNF can be constructed

---

* $L = \{ WW \mid W \in (a,b)^* \}$

S→C/AB/BA
C→LLC/L
A→LAL/a
B→LBL/b
L→a/b

NOTE: $\overline{WW}$ is CFL but WW is not.

* CFG ↔ CFL ↔ PDA

* $a^i b^j c^k \mid i=j \&\& j=k$ — 2 stacks needed
$i=j$ or $j=k$ — 1 stack needed

⇒ Pushdown automata (PDA)

1) PDA = FA + 1 stack (infinite size)
2) PDA $M = (Q, \Sigma, \delta, q_0, F, K, Z_0)$ 7tuples
   K: stack alphabets
   Z_0: initial top of stack. $Z_0 \in K$
3) Operations allowed on stack: PUSH, POP, SKIP
4) PUSH: $\delta(S, a, Z_0) = (S_2, Z_0X)$
   POP: $\delta(S, a, X) = (S_2, \in)$
   SKIP: $\delta(S, a, Z_0) = (S_2, Z_0)$
   (S: state, i/p, top of stack, next state, present top of stack)
5) PDA is not unique.
6) In PDA construction, cover only valid things, invalid things are rejected by default.
7) PDA as lang recognizer
   - DPDA: without guess
   - NPDA: with guess (By default)
8) NPDA is powerful than DPDA.
9) Acceptance method of PDA

Empty stack	Final state
At end if stack empty string is accepted (no final state)	At end of i/p if final state is reached → accepted

10) If size of stack of PDA is restricted it will accept only regular lang.

11) NO. of languages accepted by empty stack & final state method is same in NPDA

12) NO. of lang. accepted by final state is more than empty stack in DPDA. i.e DPDA by final state is more powerful than DPDA by empty stack

13) $\delta : Q \times \Sigma \cup \{\in\} \times \Gamma \to Q \times \Gamma^*$

NOTE: FA has finite memory (no. of states)

14) Drawback of PDA: can't perform more than 1 comparison.

15) Expressive power:

FA < DPDA < NPDA < TM

DFA = NFA = ENFA ; NTM = DTM

16) If L is regular ⇒ DCFL ⇒ NCFL

17) $L = \{a^n b^{2^n} \mid n \geq 1\}$
$L = \{a^n b^{n^2}\}$
$L = \{a^{n!} b^{n!}\}$
$L = \{a^p b^p \mid p \text{ is prime}\}$
} NO common difference in these infinite series Thus not CFL.

18) DCFL is closed under complementation in DPDA inter change final & non-final states.

19) Odd length palindrome $W\overset{\text{Marker}}{Z}W^R$ → DCFL
even length palindrome WW^R → CFL

20) $L = \{WW \mid W \in (a,b)^*\}$ → CSL
$L = \{WxW^R \mid W, x \in (a,b)^*\} = (a+b)^*$ → Regular
$L = \{WxW^R \mid W, x \in (a,b)^+\} = a\Sigma^+a + b\Sigma^+b$ → RL

21) $a, \in /a$ It means if i/p is a, without seeing top of stack, push it. It is NPDA. Hence for every DPDA, NPDA possible.

22) Odd length palindrome

with Marker	without Marker
$WC W^R$	$W(a+b) W^R$
(DPDA) ⇒ DCFL	(NPDA) → CFL

* $L = \{a^m b^n C^{m+n} \mid m, n \geq 1\}$ ⇒ CFL

CFG: $S \to aSc/B$
$B \to bBc/bc$

23) $L = \{a^p \mid p \text{ is prime}\}$ ⇒ CSL

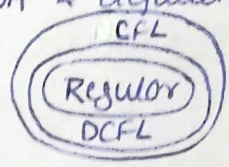
24) $L = \{a^n b^{2n}\}$ (DCFL) $\cup$ $\{a^n b^{3n}\}$ (DCFL)
L is not DCFL, it's CFL.

25) $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$ ⇒ CFL (not DCFL)

26) $L = \{a^n c\, b^{2n}\} \cup \{a^n d\, b^{3n}\}$ ⇒ DCFL

27) $L = \{ca^n b^{2n}\} \cup \{da^n b^{3n}\}$ ⇒ DCFL

NOTE: Default PDA is NPDA & default CFL is NCFL.



28) $L_1 = \{WW \mid W \in (a+b)^*\}$
$L_2 = \{WCW \mid W \in (a,b)^*\}$
$L_3 = \{a^n b^n c^n\}$
L_1, L_2 & L_3 are not CFL but $\bar{L_1}, \bar{L_2}, \bar{L_3}$ are CFL

NOTE: For every non deterministic nature, TC = $O(2^n)$

→ CFL detection:

- Infinite lang with more than one comparison is not CFL.
- Finite lang is regular.
- All palindrome lang are CFL.
- If lang is over single symbol common difference ⇒ Regular, CFL
  NO. n diff ⇒ Non-Regular, Non CFL

→ CFGs to PDA:

Step1: Push starting symbol on stack
Step2: Replace all non terminals.
Step3: POP all terminals
Step4: Goto final state

EX: $S \to aAB$
$A \to aA/b$
$B \to bB/b$

Step1: $\delta(q_0, \in, Z_0) = (q_1, Z_0S)$
Step2: $\delta(q_1, \in, S) = (q_1, aAB)$
$\delta(q_1, \in, A) = (q_1, aA)$ or (q_1, b)
$\delta(q_1, \in, B) = (q_1, bB)$ or (q_1, b)
Step3: $\delta(q_1, a, a) = (q_1, \in)$
$\delta(q_1, b, b) = (q_1, \in)$
Step4: $\delta(q_1, \in, Z_0) = (q_f, Z_0)$

* CFGs → PDA also always results in NPDA (DPDA may or may not exist)

1) $L=\{a^n b^n c^m \mid n \le 1000\}$ RL, CFL

2) $L=\{a^n b^n c^m \mid n \neq m\}$ NON-CFL

3) $L=\{a^n b^m c^n \mid n>m, n,m \le 1000\}$ RL, CFL

4) $L=\{a^n b^m \mid n-m=4\}$ CFL

5) $L=\{a^n b^m \mid n/m=4\}$ CFL

6) $L=\{a^n b^m \mid n=2m+1\}$ CFL

7) $L=\{a^n b^m \mid n \neq m\}$ CFL

8) $L=\{a^n b^m \mid n \neq 2m\}$ CFL

9) $L=\{a^n b^m \mid n=m^2\}$ NON-CFL

10) $L=\{a^{n!} b^{n!} \mid n \ge 1\}$ NON-CFL

11) $L=\{a^n b^m \mid n \le m^2\}$ NON-CFL

12) $L=\{a^n b^m c^{n+m} \mid n,m \ge 1\}$ CFL

13) $L=\{a^n b^{n+m} c^{n+m} \mid n,m \ge 1\}$ NON CFL

14) $L=\{a^n b^n b^m c^n c^m \mid n,m \ge 1\}$ NON-CFL

15) $L=\{a^{m^2} b^{n^3} c^{k^3} \mid n,m,k \ge 1\}$ NON-CFL

16) $L=\{a^{3k} b^{5l} c^{25m} \mid k,l,m \ge 1\}$ Regular, CFL

17) $L=\{a^i b^j c^k \mid j=i+k\}$ CFL

18) $L=\{a^i b^j c^k \mid i>j$ or $j<k\}$ CFL

19) $L=\{a^i b^j c^k \mid i>j>k\}$ NON CFL

20) $L=\{a^i b^j c^k \mid j=\max(i,k)\}$ NON-CFL

21) $L=\{a^i b^j c^k \mid j=i^2+k^2\}$ NON CFL

22) $L=\{a^i b^j c^k d^l \mid i=l$ & $j=k\}$ NON CFL

23) $L=\{a^i b^j c^k d^l \mid i=k$ and $j=l\}$ NON CFL

24) $L=\{a^i b^j c^k d^l \mid i=k$ or $j=l\}$ CFL

25) $L=\{a^i b^j c^k d^l \mid i=2k$ or $j \neq 5l\}$ CFL

26) $L=\{a^i b^j c^k d^l \mid i+j=k+l\}$ CFL

27) $L=\{a^i b^j c^k d^l \mid i=4l$ & $j=3k\}$ CFL

28) $L=\{a^i b^j \mid (i+j) \bmod 5 = 0\}$ Regular, CFL

29) $L=\{a^{2^n} \mid n \ge 1\}$ non regular, NON CFL

30) $L=\{a^{n^2} \mid n \ge 1\}$ " " " "

31) $L=\{1^{2n+1} \mid n \ge 1\}$ Regular, CFL

32) $L=\{a^p \mid p$ is prime$\}$ NON-CFL

33) $L=\{a^k \mid k$ is odd no$\}$ Regular, CFL

34) $L=\{wxw \mid w \in (a,b)^*\}$ NON-CFL

35) $L=\{wxw \mid w,x \in (a,b)^*\}$ Regular, CFL

36) $L=\{ww^R x \mid w,x \in (a,b)^+\}$ CFL

38) $L=\{ww^R w \mid w \in (a,b)^*\}$ NON-CFL

39) $L=\{wwR ww^R \mid w \in (a,b)^*\}$ NON-CFL

40) $L=\{wwww \mid w \in (a,b)\}$ Regular, CFL

41) $L=\{x \mid x \in (a,b,c)^*\}$ NON CFL
$n_a(x)=n_b(x)=n_c(x)$

42) $L=\{x \mid x \in (a,b)^*\}$ CFL
$n_b(x)=n_a(x)+n_c(x)$

43) $L=\{x \mid x \in (a,b,c)^*\}$ NON-CFL
$n_a(x)=n_b^2(x)+n_c^2(x)$

44) $L=\{x \mid x \in (a,b)^*\}$ Reg, CFL
$n_a(x) \bmod 5 = 0$ & $n_b(x) \bmod 4 = 0$

45) $L=\{a^n b^{2n} c^{3n} \mid n \ge 1\}$ NON CFL

46) $L=\{a^n b^n c a^m b^m \mid n,m \ge 0\}$ CFL

47) $L=\{a^n b^m c^k \mid n \neq m$ or $m \neq k\}$ CFL

48) Set of all balanced paranthesis CFL

49) Set of all lexical error produced by compiler. Reg, CFL

50) $L=\{a^i b^j c^k d^l \mid i,j,k,l \ge 1, i=j, k=l\}$ CFL

51) $L=\{a^i \mid i$ is power of 2$\}$ NON CFL

52) $L=\{2^i$ in binary$\} = 10^*$ Reg, CFL

53) $L=\{a^m b^n \mid m,n \ge 0, 5m+3n=24\}$ Reg, CFL

54) $L=\{a^m b^n \mid m,n \ge 0, 5m-3n=24\}$ CFL

★ 55) $L=\{a^n b^n \mid n \ge 0, n \neq 13\}$ DCFL

★ 56) $L=\{wwR\}$ CFL

57) $L=\{wwR\}$ CFL

⇒ Decidable problems of CFGs:

1) Membership problem:
String is generated from given grammar or not.

- CYK algo $TC=O(n^3)$ Dynamic Programming

*CYK algo: applicable only for CHOMSKY form CNF

Is 'baaba' member of grammar

$S \to AB/BC$
$A \to BA/a$
$B \to CC/b$
$C \to AB/a$

	b	a	a	b	a
	B	A,C	A,C	B	A,C
	C				

For R3C1 = R1C1 * Diagonal1 + R2C1 * Diagonal2

Match the RHS of production

- If last box contains start symbol then string is member else not.

NOTE: $n \rightarrow$ length of string

- Substring: consecutive symbols
  No of substrings = $\frac{n(n+1)}{2} + 1$
- Subsequence: may or may not be consecutive
  No of subsequences = 2^n

2) Finiteness problem:

Also:
- Simplify grammar, construct CNF.
- construct CNF graph.
- If cycle/loop present then infinite lang else finite lang.

* If lang is finite, we can define range for each non-terminal.
* Rank is the length of longest path starting from that non terminal in CNF graph.
* If rank = r, max length of string generated from that non terminal = 2^r

3) Emptiness problem:

- Eliminate unreachable productions & which does not generate any strings.
- If no production left $\Rightarrow$ it accepts empty language.

---

* DCFL is closed under complementation just interchange nonfinal & final states of DPDA.

→ Prefix: CFL's & DCFL's are closed

$S \rightarrow aSb \mid ab$

For CFG of prefix of L, write prefix of each production.

$S \rightarrow aSb \mid ab \mid \epsilon \mid a \mid aS$

→ Suffix: CFL is closed, DCFL not

$S \rightarrow aSb \mid ab$

If L is CFL, for CFG of suffix of L, write suffix of each production.

$S \rightarrow aSb \mid ab \mid \epsilon \mid b \mid Sb$

→ Substring: CFL is closed, DCFL not

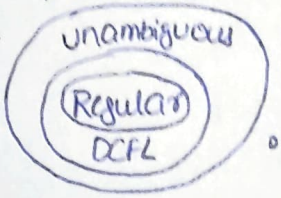
$S \rightarrow aSb \mid ab$

$S \rightarrow aSb \mid ab \mid \underbrace{a \mid S \mid b \mid aS \mid Sb \mid \epsilon}$ write substring of each production

---

→ Decidable problem: means also exist with polynomial or less time

→ Non decidable prblm: means also doesn't exist or exist with exponential or more time.



- Every DCFL is unambiguous
- Every regular lang is unambiguous
- Regular grammar can be ambiguous.

NOTE: Language generated by LL(1) & LR(1) parser is DCFL.

→ If any lang has prefix property then only it is accepted by empty stack method of DPDA.

→ Prefix Property: proper prefixes does not belong to language.

Ex: $L = \{a^n b^n \mid n \geq 1\} = \{ab, a^2b^2, \ldots\}$
ϵ, a, aa, aab

Ex: $L = \{a^n b^n \mid n \geq 0\}$ NOT have prefix property
$= \{\epsilon, ab, \ldots\}$
ϵ, a

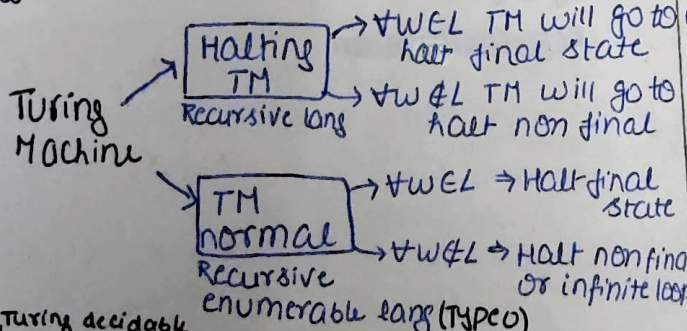
NOTE: If left recursion is present in grammar then it should be eliminated to convert to GNF.

$S \rightarrow Sa \mid b$

$S \rightarrow bS'$
$S' \rightarrow aS' \mid a$ } GNF

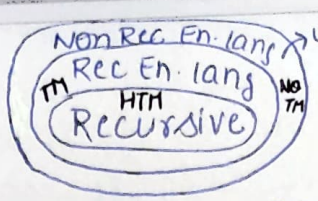
Turing Machine

1) TM = FA + 2 stacks = PDA + 1 stack
2) $M = (Q, \Sigma, \delta, q_0, F, \tau, B)$ 7 tuples
   τ: tape alphabets.
   B: initial tape symbol (Blank)
3) Tape is 2 way, infinite length.
4) In TM, i/p is present in tape.
5) In tape we can read, write, move left, right or stay on same place
   → Turnaround capability
6) $\delta: Q \times \tau \to Q \times \tau \times \{L, R, S\}$ DTM
7) Actions:-
   $a/Y, R$ → Replace a by Y & go right in tape
   $a/Y, L$ → " " " " " left " "
   $a/Y, S$ → " " " " " stay at same place
8) $\delta: Q \times \tau \to P(Q \times \tau \times \{L, R, S\})$ NTM
9) DTM ≡ NTM both have equal power and are interconvertable.
10) Tape pointer points to starting symbol of i/p string.
11) We need to take care of all necessary things, all invalid things will be rejected by default.
12) String is ∈ means all blanks in tape
13) Types of TM :-
   - Language Recogniser
   - Transducer (output generator) calculate
   - Enumerator (Lang generator) printing
14) On same state, for same tape symbol different behaviour. → NTM
15)



16) Recursive lang is decidable (Algorithm) (Turing decidable language)
    Rec enumerable lang is undecidable (Turing recognizable language) (program but not algo)
17) TM = program
    HTM = algorithm

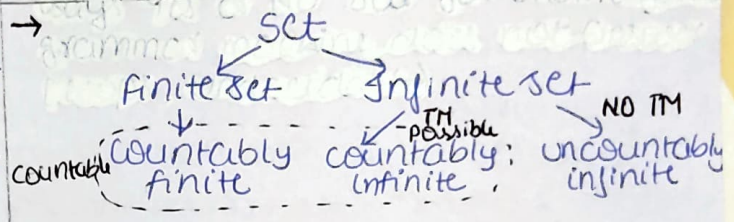
#



FA ⊂ DPDA ⊂ NPDA ⊂ LBA ⊂ HTM ⊂ TM
RL DCFL CFL CSL Rec REL

18) Halts at final state → string accepted
    halts at non " " → string rejected
    infinite loop → don't know.
19) Set of all empty grammars (no termination) is recursive finite grammar. (HTM)
20) Set of ambiguous grammar → REL (TM)

→



1) Countable set: one to one correspondance with natural nos set
   $|S| \leq N$; $N = \{1, 2, 3, \dots\}$
2) Uncountable set: $|S| > N$
   Ex: set of real nos.
3) Set of natural nos / integers / rational no → countable
   Set of real no / complex no / irrational no → uncountable.
4) Cantor's theorem :- If set S is countable then P(S) is uncountable
5) Subset of countable is countable
6) Subset of uncountable may be countable or uncountable.
7) {countable} × {countable} = {coun}
8) {unc} × {unc} = {unc}
9) {unc} × {coun} = {unc}
10) {coun} ∪ {coun} = {coun}
11) Union of infinite countable set is coun.
    $C \cup C \cup \dots = C$
12) $unc \cup unc \cup \dots = unc$
13) Σ^* → countably infinite
    $P(\Sigma^*) = 2^{\Sigma^*}$ → uncountable
    ↖ set of all lang

14) Countable: Σ^*, Set of all regular lang, Set of all CFL, Set of all CSL, Set of all recursive lang, Set of all REL, L(Any 1 lang of RL, CFL, CSL, recursive, REL, non-REL), Set of all algorithms

15) Uncountable: 2^{Σ^*}, Set of all non RL, Set of all non CFL, Set of all non CSL, Set of all non recursive lang, Set of all non REL, set of all undecidable prblms, Set of all prblms computer can't solve.

NOTE: Recursive lang is closed under complementation, REL is not.

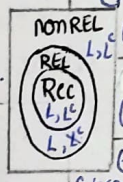
- complement of REL is NON-REL.

* For a given lang L and it's complement L^c, the following are possibilities:-

(i) Both L and L^c are recursive lang.

(ii) L is REL but not recursive then L^c is non REL.

(iii) Both L and L^c can be non REL.



⇒ Language:-

Recursive → HTM → Yes / No } Decidable (valid & invalid both have logic)

REL → TH → Yes / loop } semidecidable (undecidable) (Only valid has logic)

non-REL → TH loop } undecidable (no logic)

* Decidable: (L is recursive)
- HTM exists, Algo exists
- If HTM possible for L ⇒ HTM possible for L^c

* Semidecidable: (L is REL)
- TM exists, Program exists, no algo
- L has TM then L^c has no TM.

* Undecidable: (L is non REL)
- L has no TM (no logic)

* If we can write atleast 1 unambiguous grammar for a lang then lang is unambiguous.

Reduction:-

Problem A is reducible to problem B means B is at least as hard as A

$A \propto B$

* $A \propto B$

Undec → Undec.
Undec ↚ Undec
Dec ← Dec
Dec ↛ Dec
Recur. ← Recursive
REL ← REL
Non REL → Non REL
Non Recur → Non Recurs.

Agr x ka superset hai to $x \leftarrow x$
Agr nhi hai to $x \rightarrow x$

Decidable / Undecidable

Problem	Reg	DCFL	CFL	CSL	Rec	REL
Membership prblm (does 'w' ∈ L?)	✓	✓	✓	✓	✓	✗
Emptiness prblm (Is $L = \phi$?)	✓	✓	✓	✗	✗	✗
Finiteness prblm (Is L finite or not?)	✓	✓	✓	✗	✗	✗
Equivalence prblm (Given 2 automata accept same lang or not?)	✓	✓	✗	✗	✗	✗
Regularity prblm (L is regular?)	✓	✓	✗	✗	✗	✗
completeness (Is $L = \Sigma^*$)	✓	✓	✗	✗	✗	✗
Cofiniteness prblm (Is $\bar{L}$ finite?)	✓	✓	✗	✗	✗	✗
Ambiguity (Is lang unambiguous)	✓	✓	✗	✗	✗	✗
Intersection of 2 same type lang is of same type	✓	✗ (not closed under ∩)	✗	✓	✓	✓
Disjoint prblm (Is $L_1 \cap L_2 = \phi$?)	✓	✗	✗	✗	✗	✗
Subset prblm (Is $L_1 \subseteq L_2$?)	✓	✗	✗	✗	✗	✗
Is $\bar{L}$ & L of same type?	✓	✓	✗	✓	✓	✗

ME FER CC$_f$ A ID SL
मैं फिर सीधी फेर आया idiot साला

* For every given ambiguous regular grammar, equivalent unambiguous regular grammar can be written.
(construct m-DFA and find RG)

→ Turing machine un/decidable problems:

1) Halting problm: By reading string 'w' TM halts or not?
- Decidable for Recursive lang (HTM)
- Undecidable for REL (TM)
- Decidable for FA/DPDA/NPDA/LBA

NOTE: If not mentioned, by default grammar is Type 0 grammer (unrestricted grammar) (REL).

2) State counting problm: check whether TM has 100 states or not?
- Decidable

3) State entry problem: By reading string w, TM halts in a particular state?
- Undecidable

4) Membership problem: whether TM accepts given string or not?
- Decidable for HTM (Rec)
- Undecidable for TM (REL)

5) By reading string w, does m halts after (atleast/atmost) k-steps?
Decidable as steps is fix.

6) whether TM accepts given string w within 100 steps?
Decidable (finite number of steps)

7) Emptiness problm: TM halts on blank tape?
- Undecidable

8) Post correspondance problm (PCP) } Undecidable
9) Modified PCP

* L = {<M> / M is a TM that accepts string of length 2014} und, REL
(may loop for string of length > 2014)
* L = {<M> / M takes atleast 2016 steps on some input} Dec, Recursive