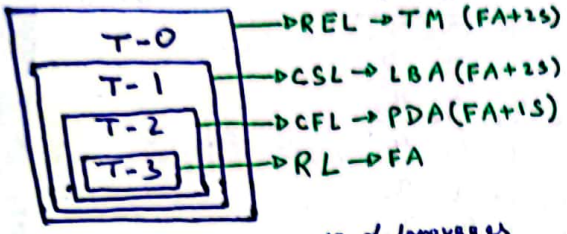
TOC

<< FA & Reg. Lang. >>

*) Chomsky hierarchy (4 types of formal lang.)



*) Expressive power = No. of languages accepted by Automata

$TM_4 > LBA_3 > PDA_2 > FA_1$

*) App.'s of Automata:
- Lexical Analyzer → FA
- Syntax " → PDA
- Semantic " → LBA

*) Alphabet: $\Sigma_\epsilon = \{a, b, c\}$
String: abbca
ϵ : special string of length '0'

Q: n-length string (distinct char.s) Substrings, Prefixes, Suffixes ?

- Substring:

0 length substring	1
1	n
2	n-1
3	n-2
. . .	
n	1

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

- Prefixes/Suffixes: For abc

Prefix = {ϵ, a, ab, abc} = 3+1
Suffix = {ϵ, c, bc, abc} = 3+1

$= n + 1$

*) Power of alphabet:

$\Sigma = \{0, 1\}$
$\Sigma^0 = \epsilon$
$\Sigma^3 = \{000, 001, 010, \ldots 111\}$
$\Sigma^* = (0+1)^*$
$\Sigma^+ = \Sigma^* - \Sigma^0$
Every Lang. $\subseteq \Sigma^*$

*) FA types

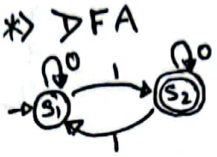
1. without o/p
   - DFA
   - NFA
2. with o/p
   - Moore
   - Mealey

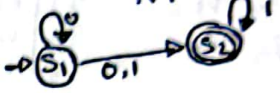
*) FA without o/p
5 tuple machine

$M = (Q, \Sigma, \delta, F, S)$

Q → Set of states (Finite non-empty)
Σ → i/p Alphabet
δ → Transition fx.n
F → Set of final states.
S → Start state

*) DFA vs NFA





DFA:
$Q = \{S_1, S_2\}$
$F = \{S_2\}$
$S = S_1$
$\delta = \delta(S_1, 0) = S_1$
$\delta(S_1, 1) = S_2$
$\delta(S_2, 0) = S_2$
$\delta(S_2, 1) = S_1$

NFA:
$Q = \{S_1, S_2\}$
$F = \{S_2\}$
$S = S_1$
$\delta = \delta(S_1, 0) = S_1, S_2$
$\delta(S_1, 1) = S_2$
$\delta(S_2, 0) = —$
$\delta(S_2, 1) = S_2$

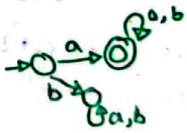
DFA ≤ NFA

*) DFA constr.n:

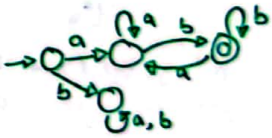
- (a+b)*

- (a+b)+

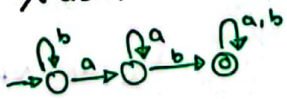
- a(a+b)*



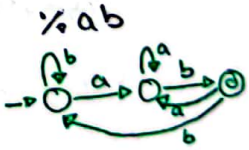
- a(a+b)* b



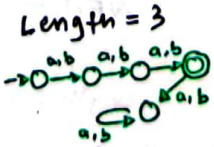
- %ab%



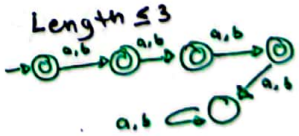
- %ab



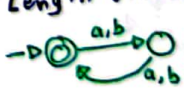
Length = 3



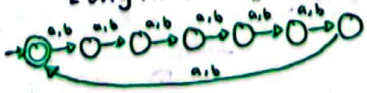
Length ≤ 3



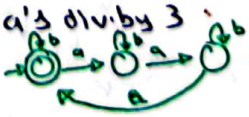
Length div. by 2



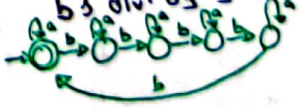
Length div. by 7



a's div. by 3

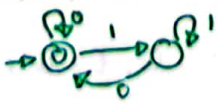


b's div. by 5



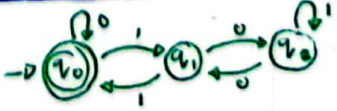
Note: •Length exactly 'n' requires → n+2 states
•Length div. by 'n' requires → n states

Binary no.s div. by 2
0, 10, 100, 110, 1000 → ending with 0



Binary no.s div. by 3
ϵ = Binary => 0,1 ; Div. by 3 => remainder = 0, 1, 2

Remainder	0	1
→q_0*	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



Ternary no.s div. by 7

Ternary ⇒ $\Sigma = \{0,1,2\}$

div. by 7 ⇒ rem. = 0 to 6

	0	1	2
→q_0^*	q_0	q_1	q_2
q_1	q_3	q_4	q_5
q_2	q_6	q_0	q_1
q_3	q_2	q_3	q_4
q_4	q_5	q_6	q_0
q_5	q_1	q_2	q_3
q_6	q_4	q_5	q_6

Ternary no.s div. by 6 && Starts with 2

Ternary ⇒ $\Sigma = \{0,1,2\}$

div. by 6 ⇒ rem. = 0 to 5

Starts with 2 ⇒ →S —2→ q_2, S —0,1→ Dead

	0	1	2
→S	D	D	q_2
D	D	D	D
q_0^*	q_0	q_1	q_2
q_1	q_3	q_4	q_5
q_2	q_0	q_1	q_2
q_3	q_3	q_4	q_5
q_4	q_0	q_1	q_2
q_5	q_3	q_4	q_5

Base 4 no.s div. by 5 && starts with 2 & 3

Base 4 ⇒ $\Sigma = \{0,1,2,3\}$

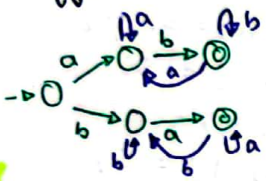
div. by 5 ⇒ rem. = 0 to 4

starts with 2&3 ⇒ →S —2→ q_2, S —3→ q_3, S —0,1→ Dead

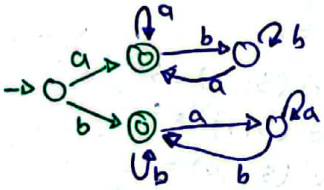
	0	1	2	3
→S	D	D	q_2	q_3
D	D	D	D	D
q_0^*	q_0	q_1	q_2	q_3
q_1	q_4	q_0	q_1	q_2
q_2	q_3	q_4	q_0	q_1
q_3	q_2	q_3	q_4	q_0
q_4	q_1	q_2	q_3	q_4

Note: $(Num)_m$ div. by n requires → n states

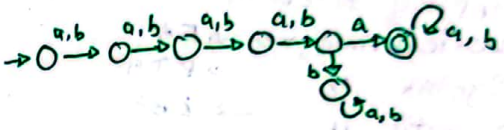
Starting & ending symbols different a%b or b%a

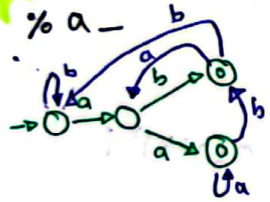


Starting-ending symbols same a or a%a or b%b or b

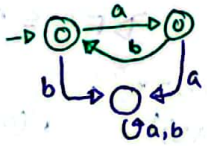


---- a %

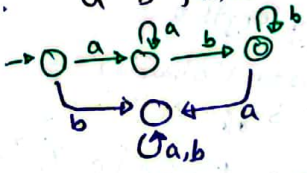




All prefixes of abababab...



$a^m b^n$; m,n > 0



m = n ✗ FA not possible

m ≠ n ✗

m > n ✗

m < n ✗

m+n = 10 ✓ (∵ Finite)

m−n = 10 ✗ (m = n+10)

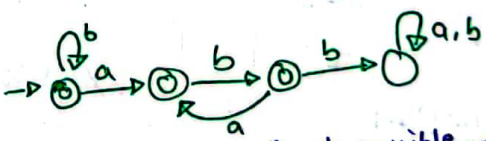
$a^n b^n$; n = 1 crore ✓ (∵ Finite)

Note: n^{th} symbol from LHS 'a' → n+2 states

n^{th} " " RHS 'a' → 2^n states

Not contain abb substring

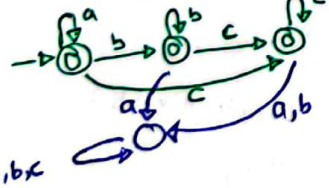
abb substr. ⇒ FA → complement



Note: NFA compl.n not possible, so do

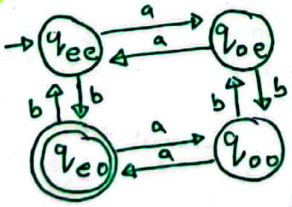
NFA —O(2^n)→ DFA —O(n)→ complm.n

$a^m b^n c^k$; m,n,k ≥ 0



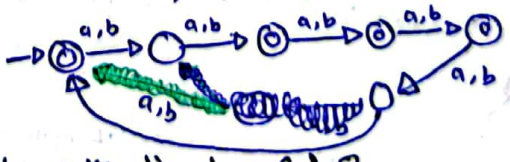
a,b,c

a's even & b's odd



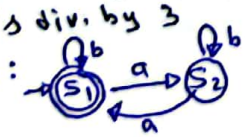
Length div. by 2 or 3 or 4

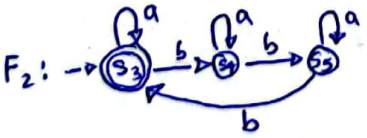
Pattern = 0, 2, 3, 4, 6, 8, 9, 10, 12, (2, 1, 1, 2, 2, 1, 1, 2)



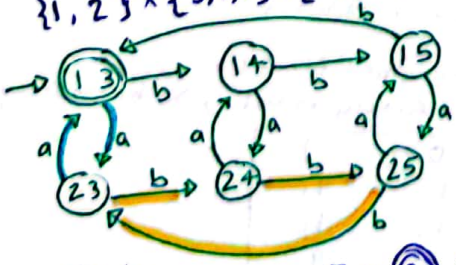
⇒ 6 states

a's div. by 2 && b's div. by 3





{1,2} × {3,4,5} = {13, 14, 15, 23, 24, 25}



F_1: (S1) (S2)

F_2: (S3) (S4) (S5)

F_1 = 13, 14, 15

F_2 = 13, 23

F_1 && F_2 ⇒ $F_1 \cap F_2$ = 13 ↓ Final

Length div. by 6 & 8

LCM(6,8) = 24 states

Length div. by 6 or 8

Pattern = 0, 6, 8, 12, 16, 18, 24, 30, 32, 36, 40, 42, 48 (6, 2, 4, 4, 2, 6, 6, 2, 4, 4, 2, 6)

⇒ 24 states

Length div. by 2 & 3 & 4

LCM(2,3,4) = 12

Length div. by 6 & 18

LCM(6,18) = 18

Length div. by 6 or 18

Pattern = 0, 6, 12, 18, 24, 30, 36, 42 (6 6 6 6 6 6)

⇒ 6 states

Length div. by 2 or 3

Pattern = 0, 2, 3, 4, 6, 8, 9, 10, 12 (2 1 1 2 2 1 1 2)

⇒ 6 states

Note:) Conclusions

1. Subproblems on different symbols ⇒ Blindly Multiply

Ex: a's div. by 2 (&&/or) b's div. by 3 (&&/or) c's not div by 6

⇒ 36 states.

2. Subproblems on same symbols or length

i) && ⇒ LCM | Ex: Len. div. by 6 && 8 | Ex: Len. div. by 2 && 3 && 4

ii) OR ⇒ Pattern | Ex: Len. div. by 2 or 3 | Ex: Len. div. by 6 or 8

Q: States = 3 = {x, y, z}
x always initial
Σ = {0, 1}
Possible no. of DFAs? = $2^3 \times 3^6$

	0	1
→x	3	3
y	3	3
z	3	3

2^3 ← Possibilities of final state; 3, 3 ← Possibilities

Q: ..same... but x is initial & final
= 3^6

Q: ... same... x is initial & final but there maybe some other finals also
= $2^2 \times 3^6$

	0	1
→x*	3	3
y	3	3
z	3	3

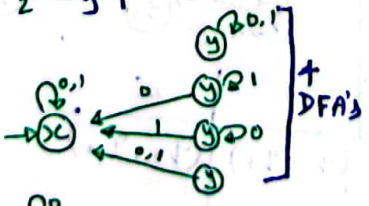
2^2 ← y, z

Q: States = 2 = {x, y}
x always initial
Σ = {0, 1, 2}
DFA's ? = $2^2 \times 2^6$

	0	1	2
→x	2	2	2
y	2	2	2

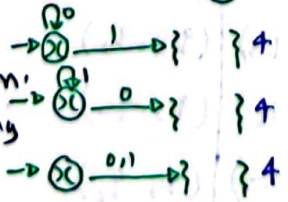
2^2 ← x, y

Q: States = 2 = {x, y}
x initial always
Σ = {0, 1}
DFA's = ? if



i) It accepts empty Lan:

ii) It accepts everything



1. 16 DFA (no final) 16✓ 16✗
2. 16 DFA (x final) 16✗ 4✓
3. 16 DFA (y final) 4✓ 16✗
4. 16 DFA (x & y final) 16✗ 16✓

i) 20 DFA's
ii) 20 DFA's

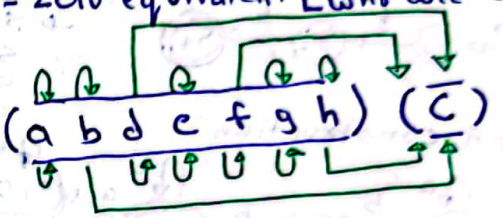
*) DFA to minimal DFA

- Minimization only applicable to DFA's
- Partition Algo used [O(n lg n)]

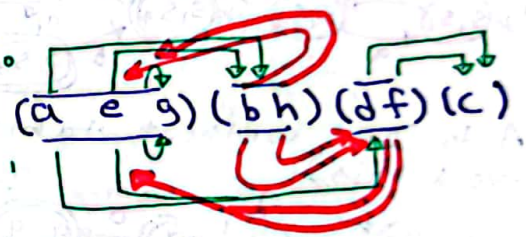
- DFA

	0	1
→a	b	f
b	g	c
*c	a	c
d	c	g
e	h	f
f	c	g
g	g	e
h	g	c

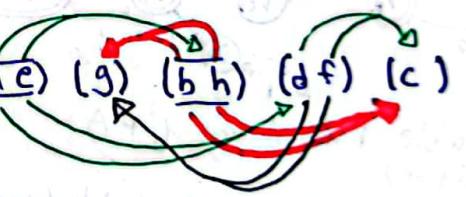
π_0 = zero equivalent [after reading ε who are equal]



π_1 =



π_2 =



Note: DFA's one many, but minimal DFA is unique.

π_3 = (ae) (g) (bh) (df) (c)

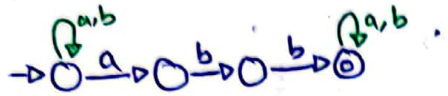
abdefgh → zero equivalent
aeg → one equivalent
ae, g, bh, df, c → forever "

*) NFA

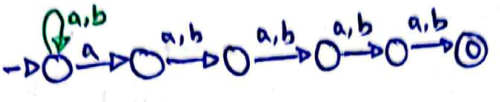
- DFA ⇒ $Q \times \Sigma \to Q$
  NFA ⇒ $Q \times \Sigma \to 2^Q$
  ε-NFA ⇒ $Q \times \Sigma \cup \epsilon \to 2^Q$
- DFA: Cover all strings (Σ^*) (either accept or reject)
  NFA: Cover only valid string (Invalid by default reject)
- NFA power = DFA power = ε-NFA Power
- If NFA 'n' states, then DFA max^m 2^n (wc)

*) NFA construction:

% a bb %

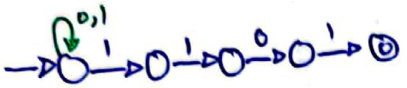


% a ---- %



Note:) N^{th} Symbol from RHS ⇒ n+1 states

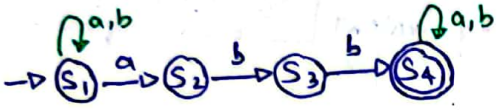
% 1101



*) NFA to DFA, then DFA to DFA complement

Ex: Not containing abb as substring

1. NFA for % abb %

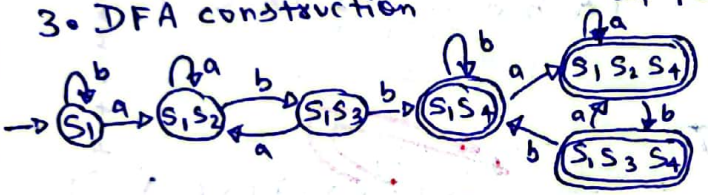


2. NFA to DFA

NFA	a	b	
→ S_1	S_1S_2	S_1	as it is copy
S_2	–	S_3	
S_3	–	S_4	
S_4*	S_4	S_4	

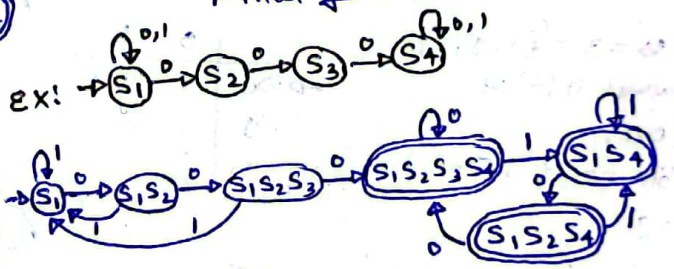
DFA	a	b
→ S_1	S_1S_2	S_1
S_1S_2	S_1S_2	S_1S_3
S_1S_3	S_1S_2	S_1S_4
* S_1S_4	$S_1S_2S_4$	S_1S_4
* $S_1S_2S_4$	$S_1S_2S_4$	$S_1S_3S_4$
* $S_1S_3S_4$	$S_1S_2S_4$	S_1S_4

3. DFA construction



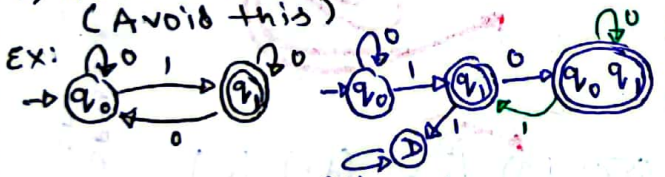
4. DFA to DFA^c

Final ⇄ Non-final



*) NFA to DFA without table: (Avoid this)

Ex:



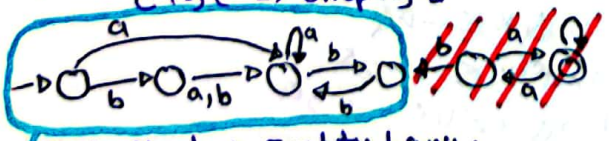
Ex:

*) Decidable Problems of FA (or Decision)

- 1. Emptyness (FA accepts empty L or not)
  2. Finiteness (——— Finite ———)
  3. Equality (Two DFAs equal or not)

- Emptyness:- FA should've no final state or unreachable final state.

Algo: 1. Delete unreachables (with edges)
(for both NFA/DFA) 2. If (Final states ≥ 1) ⇒ Non-empty L
else ⇒ empty L

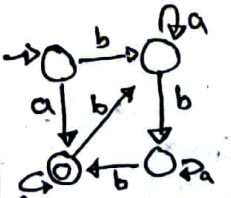


No final ⇒ Empty Lang.

- Finiteness:- To become infinite, loops should be there.

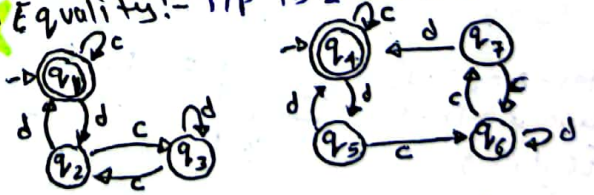
Algo: 1. Delete unreachables (with edges)
(for both NFA/DFA) 2. Delete dead states (with edges)
3. If (at least one loop && one final) ⇒ ∞ Lang.
else ⇒ finite Lang.

Note:) Empty lang. is finite.



1. ✓
2. ✓
3. Final ✓, Loop ✓
⇒ ∞ Lang.

- Equality:- i/p is 2 DFA's



	c	d
(q_1, q_4)	(q_1, q_4) Both final	(q_2, q_5) Both non-Final ⇒ No problem
(q_2, q_5)	(q_3, q_6) NF	(q_1, q_4) NF
(q_3, q_6)	(q_2, q_7) NF	(q_3, q_6) NF
(q_2, q_7)	(q_3, q_6) NF	(q_1, q_4) F

$DFA_1 = DFA_2$

i.e. Lang. acceptance same.

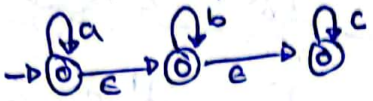
*) ϵ-NFA:

- Required when :-
  1. Needs to maintain order
  2. Needs to go to another state w/o reading any symbol.

*) ϵ-NFA construction:-

$a^m b^n$; $m,n \geq 0$

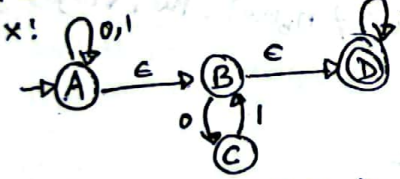
$a^m b^n c^k$; $m,n,k \geq 0$



*) ϵ-NFA to NFA :-

- Things that don't change:
  1) initial state
  2) No. of states
- Things that change :
  1) Final states
  2) Transition fx.n
- By reading ϵ in ϵ-NFA if you can go to final, then make that state final in NFA.
  (पुरानी final states तो final है ही)

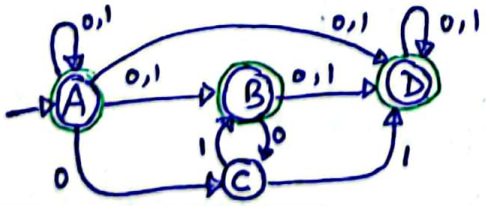
Ex:



ϵclosure: A = A, B, D
B = B, D
C = C
D = D

Transitions:

$\delta(A,0) = A,B,D,C$; $\delta(A,1) = A,B,D$
$\delta(B,0) = C,D$; $\delta(B,1) = D$
$\delta(C,0) = \phi$; $\delta(C,1) = B,D$
$\delta(D,0) = D$; $\delta(D,1) = D$



*) Regex :

- one of the mathematical way to represent RL
- FA ✓ ⇒ L is regular
  Regex ✓ ⇒ L is regular
- Operators in regex :
  1. Kleene closure (*)
  2. Concatenation (.)
  3. Union (+)
- Priority: *) .) + Associativity: L to R

*) Regex construction:

{ } {ϵ} {a,b} {ab} {ϵ,a,aa,aaa...}
ϕ ϵ (a+b) a.b a*

{ϵ, 11, 1111, 111111...} = (11)*

Start with a end with b = a (a+b)* b

Starting-ending symbol different
a(a+b)*b + b(a+b)*a

Starting ending symbol same
a(a+b)*a + b(a+b)*b + a + b +

abb as substring = (a+b)* abb (a+b)*
Length exactly 3 = (a+b)^3
Length at least 3 = (a+b)^3 (a+b)*
Length at most 3 = (a+b+ϵ)^3
Even length = [(a+b)^2]*
Odd length = (a+b)[(a+b)^2]*
Start with a & no two consecutive b's = a.(ba+a)*(b+ϵ) or (a+ab)+
No two consecutive b's = (a+ba)*(b+ϵ) or (b+ϵ)(a+ab)*
No two consecutive a's & b's = (b+ϵ)(ab)*(a+ϵ) or (a+ϵ)(ba)*(b+ϵ)
Exactly 2 a's (need not be consecutive) = b*a b*a b*
Atmost 2 a's = b*(a+ϵ)b*(a+ϵ)b*
Atleast 2 a's = (a+b)*a(a+b)*a(a+b)*

Note:) $\phi . a = \phi$; $a^* . a = a^+$
$\epsilon^* = \epsilon$
$\phi^* = \epsilon$
$\phi^+ = \phi$
$(a^*)^* = a^*$

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

Note:) Shifting rule
$a.(\gamma a)^* = (a\gamma)^* a$

Odd zero's or odd 1's $= (0+1)[(0+1)^2]^*$
or $[0\cdot(1^*01^*01^*)+1\cdot(0^*10^*10^*)]$

$\Sigma=\{0,1,2\}$; 2 immediately followed by exactly two 0's. || 1 immediately followed by 0 or 20

$0^*(200+100^*+1200)^*$

*) Properties of Regex:

1. Associativity:

Union: $(\gamma_1+\gamma_2)+\gamma_3 = \gamma_1+(\gamma_2+\gamma_3)$ ✓

Concat.: $(\gamma_1\cdot\gamma_2)\cdot\gamma_3 = \gamma_1\cdot(\gamma_2\cdot\gamma_3)$ ✓

2. Commutativity:

U: ✓ $\gamma_1+\gamma_2 = \gamma_2+\gamma_1$

C: ✗ $\gamma_1\cdot\gamma_2 \neq \gamma_2\cdot\gamma_1$

3. Annihilator:

U: ✗ γ_1+ ___ $=$ ___ (not possible)

C: ✓ $\gamma_1\cdot\underline{\phi} = \underline{\phi}$

4. Identity:

U ✓ $\gamma_1+\phi=\gamma_1$

C ✓ $\gamma_1\cdot\epsilon=\gamma_1$

5. Distributive:

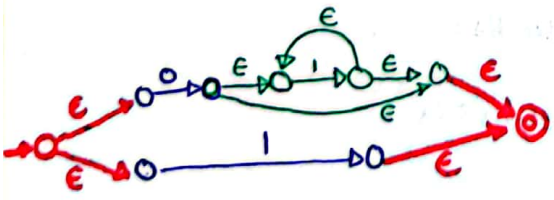
U: ✗ $\gamma_1+(\gamma_2\cdot\gamma_3) \neq (\gamma_1+\gamma_2)\cdot(\gamma_1+\gamma_3)$

C ✓ $\gamma_1\cdot(\gamma_2+\gamma_3) = (\gamma_1\cdot\gamma_2)+(\gamma_1\cdot\gamma_3)$

*) RE to FA :-

1. By Brute force (Synthesis method):

$0\cdot1^*+1$

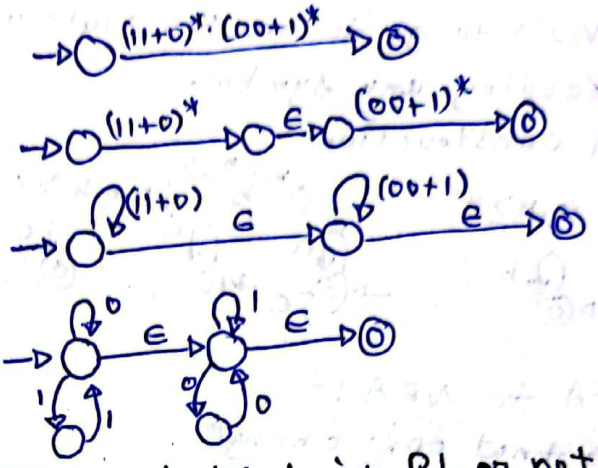


2. Direct

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

3. By Decomposition:

$(11+0)^*\ (00+1)^*$



*) To check if L is RL or not :-

$L=\{a^nb^n \mid n\leq10\}$

$\{a^nb^n \mid n<2^{\text{'c' zeroes}}\}$

$\{a^nb^nc^kd^k \mid n<100, k<1000\}$

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

$\{w \mid w\in(a+b)^*, n_a(w)<n_b(w)<10\}$

$\{a^p \mid p\in\text{Prime}, p<19\}$

$\{a^{n^n} \mid n<1000\}$

$\{w \mid w\in(a+b)^*, n_a(w)\neq n_b(w), |w|<10\}$

$\Rightarrow$ Finite $\Rightarrow$ Regular

$(aa)^* \Rightarrow$ RE $\Rightarrow$ RL

$\{a^{3n} \mid n\geq1\}=\{a^3,a^6,a^9\cdots\}=(a^3)^+ \Rightarrow$ RE $\Rightarrow$ RL

$(aa+aaa)^* \Rightarrow$ RE $\Rightarrow$ RL

$\{a^{3n+1000} \mid n\geq0\} \Rightarrow a^{1000}\cdot(a^3)^* \Rightarrow$ RE $\Rightarrow$ RL

$\{a^{mn} \mid m\geq1, n\geq1\} \Rightarrow^{n=1,\ m\in[1,\infty)} a^+ \Rightarrow$ RL

$\{(a^p)^* \mid p\in\text{Prime}, p>1\}$

$=(a^2)^*, (a^3)^*\cdots = \epsilon, a^2, a^3, a^4, a^5, a^6, a^7\cdots$

$= \epsilon+a^2\cdot a^* \Rightarrow$ RL

$\{(a^{n^2})^* \mid n\geq1\} = \epsilon, a, aa, aaa\cdots = a^* \Rightarrow$ RL

$\{(a^{n^{1000}})^* \mid n\geq1\} = \epsilon, a, aa, aaa\cdots = a^* \Rightarrow$ RL

$\{a^n\cdot(a^m)^* \mid m\geq1, n\geq1\} = a\cdot a^* = a^+ \Rightarrow$ RL

Note: If infinite lang. over single alphabet produces A.P series over string length $\Rightarrow$ RL

$\{a^{n^2} \mid n \geq 1\}$

$a^1, a^4, a^9, a^{16}, a^{25}, a^{36}$

diff ⇒ 3, 5, 7, 9, 11

⇒ Not an AP over length ⇒ Not RL

$\{a^{n!} \mid n \geq 1\}$

$a^1, a^2, a^6, a^{24}, a^{120}, a^{720}$

⇒ Not AP ⇒ Not RL

$\{a^p \mid p \in \text{Prime}, p > 1\}$

$a^2, a^3, a^5, a^7, a^{11}, a^{13}$

⇒ Not AP ⇒ Not RL

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

$= a^1, a^4, a^{27}, a^{256}$

⇒ not AP ⇒ Not RL

$\{a^{2^n} \mid n \geq 1\}$

$a^2, a^4, a^8, a^{16}, a^{32}$

⇒ not AP ⇒ not RL

$\{(a^{2^n})^* \mid n \geq 1\}$

$\epsilon, a^2, a^4, a^6, a^8$

$= (a^2)^* \Rightarrow RL$

$\{a^{100^n} \mid n \geq 1\}$
$\{a^{n^{n^n}} \mid n \geq 1\}$
$\{a^{2^{n^2}} \mid n \geq 1\}$
] not AP ⇒ Not RL

$\{a^m b^n \mid m, n \geq 1\} = a^+ b^+ \Rightarrow RL$

$\{a^l b^n c^m \mid n < m < 10\}$ ⇒ Finite ⇒ RL

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

$m = n$
$m \neq n$
$m < n$
$m > n$
$m - n = 10$
] Not RL

$m + n = 10$] Finite ⇒ RL

$\{x \cdot y \mid x, y \in (a+b)^*, |x| = |y|\}$ ★

Lengths = 0, 2, 4, 6, 8, ⋯ ⇒ Even length string

$= [(a+b)^2]^* \Rightarrow RL$

$\{xy \mid |x| = |y| + 1\}$ ★

$\xrightarrow[0+\epsilon \to 0]{\epsilon+0 \to 0}$ ⇒ odd length string

$= (a+b)[(a+b)^2]^* \Rightarrow RL$

$\{a^m b^n \mid m+n = \text{even}\}$

$\xrightarrow[0+0 \to \epsilon]{\epsilon+\epsilon \to \epsilon}$ $= (a^2)^*(b^2)^* + a(a^2)^* \cdot b(b^2)^* \Rightarrow RL$

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

$w = \epsilon \Rightarrow x^* \Rightarrow (a+b)^* \Rightarrow RL$

$\{ww \mid w \in (a+b)^*\}$ ⇒ Length equal && content equal

a bb | a bb — can handle — can't handle

⇒ not RL

$\{ww^R \mid w \in (a+b)^*\}$

a bb | bb a non RL

$\{w \# w^R \mid w \in (a+b)^*\}$ abb#bba not RL

$\{a^m b^n c^{2k} \mid k, m, n \geq 1\}$

AP, non-AP [2, 4, 8, 16] ⇒ not RL

$\{a^p \cdot b^* \mid p \in \text{Prime}\}$

non-AP, AP ⇒ not RL

$\{w \# w \mid w \in (a+b)^*\}$ abb#abb not RL

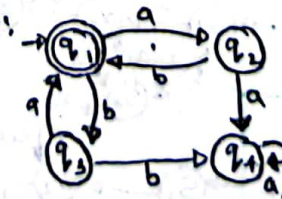
⇒ Conclusion: L

- Finite ⇒ RL
- Infinite
  - $|\Sigma| = 1$
    - AP ⇒ RL
    - non-AP ⇒ non RL
  - $|\Sigma| > 1$
    - Comparison (Logical) ⇒ not RL
    - No-comparison (Logically)
      - All in AP ⇒ RL
      - Few of them non-AP ⇒ not RL

*) FA to Regex:

1. By Arden's method:
   - Applicable only for ε-free NFA, DFA
   - $R = Q + RP$ can be written as $R = QP^*$ — Two P, Q → Two Regex
     - P doesn't contain ε
     - If P contains then ∞ sol.ⁿs

Ex:



Step-1: characteristic eqⁿs:-

$q_1 = \epsilon + q_2 b + q_3 a$ ← Final state eqⁿ

$q_2 = q_1 a$

$q_3 = q_1 b$

$q_4 = q_3 b + q_2 a + q_4 (a+b)$

Step-2: Solve final state eqⁿ in terms of a's & b's

Put q_2 & q_3 in q_1

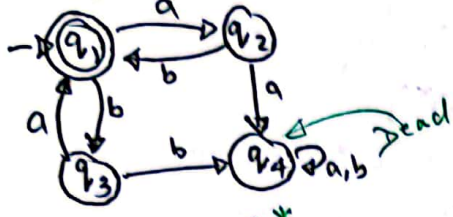
$q_1 = \epsilon + q_1 ab + q_1 ba$

$q_1 = \epsilon + q_1 (ab + ba)$

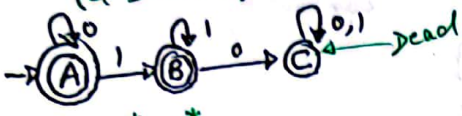
$q_1 = \epsilon \cdot (ab + ba)^*$

$= (ab + ba)^*$

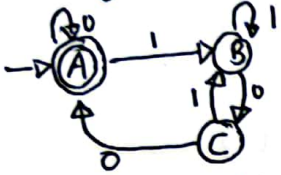
2. By intuition method:



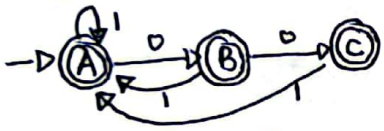
$(ab+ba)^*$



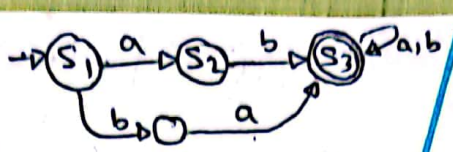
$0^* 1^*$



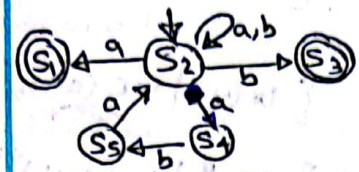
$[0^* + 1\cdot(1+01)^*00]^*$



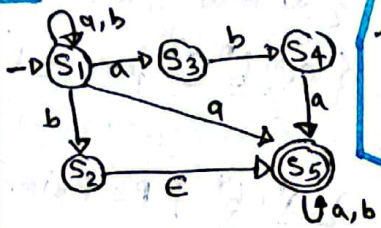
$(1+01+001)^* (\epsilon+0+00)$



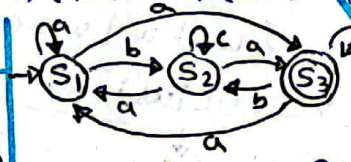
$(ab+ba)(a+b)^*$



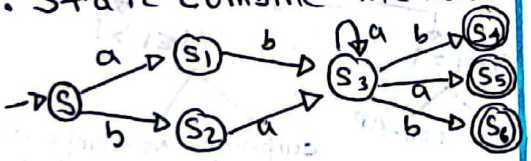
$(a+b+aba)^*(a+b)$
$= (a+b)^* (a+b)$
$= (a+b)^+$



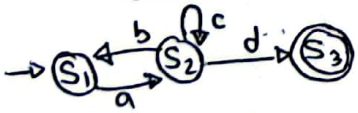
$(a+b)^*(aba+a+b)(a+b)^*$



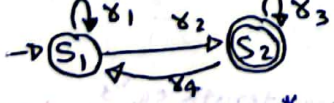
3. State combine method:



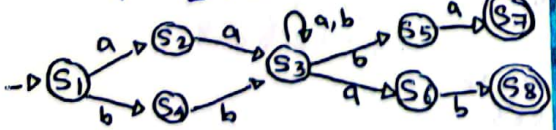
$(ab+ba)a^*(a+b)$



$a(ba+c)^* d$
or $(ac^*b)^*(ac^*d)$

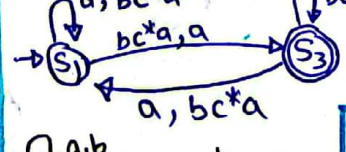


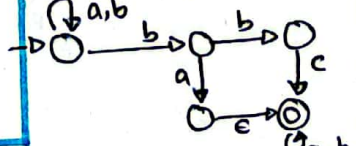
$(\gamma_1+\gamma_2\gamma_3^*\gamma_4)^*\gamma_2\gamma_3^*$
or
$\gamma_1^*\cdot\gamma_2(\gamma_3+\gamma_4\gamma_1^*\gamma_2)^*$



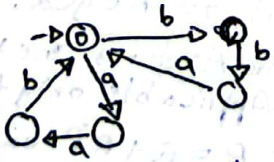
$(aa+bb)(a+b)^*(ba+ab)$

ϕ or $\{\}$

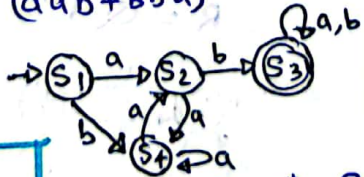


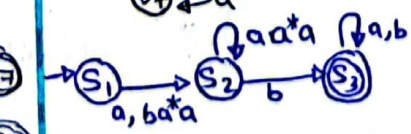


$(a+b)^* b(bc+a)(a+b)^*$

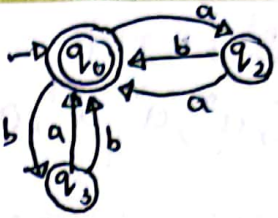


$(aab+bba)^*$

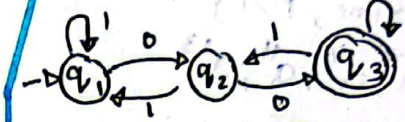




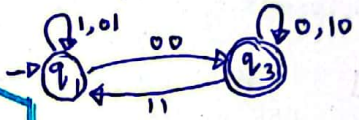
$(a+ba^*a)(aa^*a)^*b(a+b)^*$

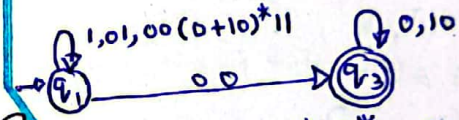


$[a(a+b)+b(a+b)]^*$
$((a+b)^2)^*$

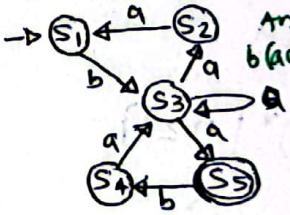


Any:
$1^*0(10)^*0(0+10)^*$

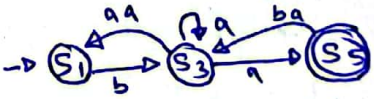


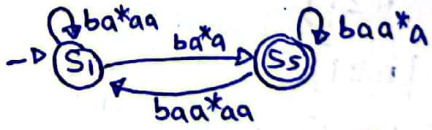


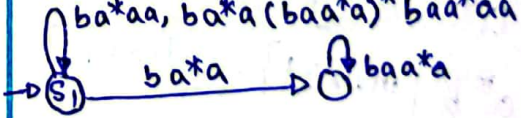
$[1+01+00(0+10)^*11]^*00(0+10)^*$



Any:
$b(aab+a)^*a(baa)^*$



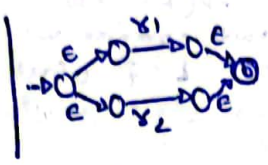




*) Closure properties of RL: Except Subset & infinite followed by any operation, everything is closed for Regular Languages.

1. Union:

$L_1 = RL$, $L_2 = RL$
$\Rightarrow L_1 \cup L_2 = RL$



Note! complement of non RL is also non RL
Ex: $a^n b^n \Rightarrow$ nRL, $(a^n b^n)^c \Rightarrow$ nRL

Questions:

$L_1 = \Sigma^*$
$L_1 \cup L_2 = \Sigma^*$
$\Rightarrow$ RL

$L_1 = \phi$
$L_1 \cup L_2 = L_2$
$\Rightarrow$ Depends on L_2

$L_1 = a^* b^*$, $L_2 = a^n b^n$
$L_1 \cup L_2 = a^* b^*$
$\Rightarrow$ RL

$L_1 = (a+b)^* b (a+b)^*$
$L_2 = (a+b)^* a (a+b)^*$
$L_1 \cup L_2 = (a+b)^+$
$\Rightarrow$ RL

$L_1 = a^* b^*$
$L_2 = (a+b)^*$
$L_1 \cup L_2 = (a+b)^*$
$\Rightarrow$ RL

$L_1 = RL$
$L_2 = $ non RL
$L_1 \cup L_2 =$ can't say
$\phi \cup a^n b^n \Rightarrow$ non RL
$(a+b)^* \cup a^n b^n \Rightarrow$ RL

$L_1 \cup L_2 = RL$
$\Rightarrow$ one RL other nRL
or Both RL
or Both nRL

$\rightarrow (a+b)^* \cup a^n b^n = (a+b)^*$
$\rightarrow ab \cup a^2 b^2 = \{ab, a^2 b^2\}$
$\rightarrow a^n b^n \cup (a^n b^n)^c = \phi$

2. Intersection:

$L_1 = RL$, $L_2 = RL$
$\Rightarrow L_1 \cap L_2 = RL$

Final state of $DFA_1 \Rightarrow F_1$
$DFA_2 \Rightarrow F_2$
$DFA_1 \times DFA_2 \Rightarrow F_1 F_2$
(See example on P.no.2 a's div. by 2 & b's div. by 3)

Questions:

$L_1 = \phi$
$L_1 \cap L_2 = \phi$
$\Rightarrow$ RL

$L_1 = (a+b)^*$
$L_1 \cap L_2 = L_2$
$\Rightarrow$ Depends on L_2

$a^n b^n \cap a^* b^*$
$a^n b^n$
$\Rightarrow$ nRL

$a^* \cap (aa)^*$
$(aa)^*$
$\Rightarrow$ RL

$a^* b \cap ab^*$
ab
$\Rightarrow$ RL

$a^* b^* \cap b^* a^*$
$a^* + b^*$
$\Rightarrow$ RL

$a^+ b^+ \cap b^+ a^+$
$= \phi \Rightarrow$ RL

$ab^+ \cap ba^+$
$= \phi \Rightarrow$ RL

$L_1 \cap L_2 = RL$
Both RL $\rightarrow a \cap b = \phi$
or Both NRL $\rightarrow a^n b^n \cap (a^n b^n)^c = \phi$
or one RL one nRL $\rightarrow ab \cap a^n b^n = ab$

$L_1 = RL$, $L_2 = $ nRL
$L_1 \cap L_2 =$ can't say
$(a+b)^* \cap a^n b^n = a^n b^n \Rightarrow$ nRL
$ab \cap a^n b^n = ab \Rightarrow$ RL

$L_1 = $ nRL, $L_2 = $ nRL
$L_1 \cap L_2 =$ can't say
$a^n b^n \cap a^n b^n = a^n b^n \Rightarrow$ nRL
$a^n b^n \cap (a^n b^n)^c = \phi \Rightarrow$ RL

$L_1 = a^m b^n c^n$
$L_2 = a^m b^n c^n$
$L_1 \cap L_2 = a^n b^n c^n$
$\Rightarrow$ nRL

Infinite union of RL = ? can't say

$ab \cup a^2 b^2 \cup a^3 b^3 \ldots = a^n b^n \Rightarrow$ nRL
$a \cup a^2 \cup a^3 \ldots = a^* \Rightarrow$ RL

Infinite intersection of RL = ? can't say

$\overline{ab} \cap \overline{a^2 b^2} \cap \overline{a^3 b^3} \ldots = (\Sigma^* - ab) \cap (\Sigma^* - a^2 b^2) \cap (\Sigma^* - a^3 b^3) \ldots = \overline{a^n b^n} \Rightarrow$ nRL
$ab \cap a^2 b^2 \cap a^3 b^3 \ldots = \phi \Rightarrow$ RL

$L_1 = \Sigma^*$, $L_1 \cap L_2 = RL$
$\Sigma^* \cap L_2 = RL \Rightarrow L_2 = RL$

$L_1 = \Sigma^*$, $L_1 \cap L_2 =$ NRL
$\Sigma^* \cap L_2 =$ NRL $\Rightarrow L_2 =$ nRL

$\Sigma^* \cap L_2 = (ab)^*$
$\Rightarrow L_2 = (ab)^*$

3. Complementation:

$L = RL$
$\Rightarrow L^c = RL$

In DFA: final $\leftrightarrow$ non-final

Questions:

$L = \phi$
$L^c = \Sigma^*$
$\Rightarrow$ RL

$L = a(a+b)^*$
$L^c = \Sigma^* - L$
$= b(a+b)^* + \epsilon$
$\Rightarrow$ RL

$L = (a+b)^* b (a+b)^*$
= containing b
$L^c = \Sigma^* - L$
$= a^*$
$\Rightarrow$ RL

4. Difference:

$L = RL$, $L_2 = RL$
$\Rightarrow L_1 - L_2$ or $L_2 - L_1 = RL$

$L_1 - L_2 = L_1 \cap L_2^c$

Questions:

$\phi - L_1$
$= \phi$
$\Rightarrow$ RL

$\Sigma^* - L_1$
$= L_1^c$
$\Rightarrow$ RL (if L_1 is RL)

$L_1 - \Sigma^*$
$= \phi$
$\Rightarrow$ RL

$a^* - b^*$
$= a^+$
$\Rightarrow$ RL

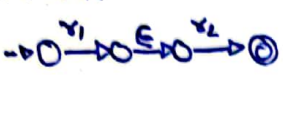
$b^* - a^*$
$= b^+$
$\Rightarrow$ RL

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

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

5. Concatenation:

$L_1 = RL, L_2 = RL$
$\Rightarrow L_1 \cdot L_2 = RL$



Questions:

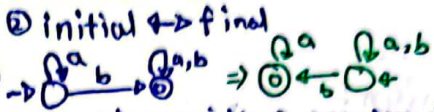
a^*b^* $\quad \phi \cdot L_1$ $\quad \epsilon \cdot L_1$ $\quad \epsilon^* a$ $\quad a \cdot \epsilon^*$
$= a^* b^*$ $\quad = \phi$ $\quad = L_1$ $\quad (a+b)^* \cdot a$ $\quad a \cdot (a+b)^*$
$\Rightarrow RL$ $\quad \Rightarrow RL$ $\quad \Rightarrow RL$ (if L_1 is RL) $\quad \Rightarrow RL$ $\quad \Rightarrow RL$

$(a+b)^* \cdot a^*$ $\quad a^* \cdot (a+b)^*$ $\quad (a+b)^* \cdot a^+$ $\quad a^+ \cdot (a+b)^*$
$= (a+b)^*$ $\quad = (a+b)^*$ $\quad = (a+b)^* \cdot a$ $\quad = a(a+b)^*$
$\Rightarrow RL$ $\quad \Rightarrow RL$ $\quad \Rightarrow RL$ $\quad \Rightarrow RL$

6. Reversal: of DFA

$L = RL$
$\Rightarrow L^R = RL$

① Reverse edges
② Initial $\leftrightarrow$ final



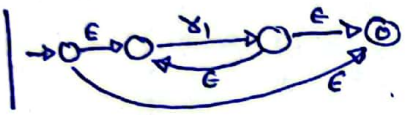
If many finals, combine with ϵ production

$L = \epsilon^*$ $\quad L = a^*$ $\quad L = \phi$ $\quad a(a+b)^*$ $\quad a^*b^*$
$L^R = \epsilon^*$ $\quad L^R = a^*$ $\quad L^R = \phi$ $\quad (a+b)^* a$ $\quad b^* a^*$
$\Rightarrow RL$ $\quad \Rightarrow RL$

$(a+b)^* ab$ $\quad (a+b)^* ab (a+b)^*$
$ba(a+b)^*$ $\quad (a+b)^* ba (a+b)^*$

7. Kleene closure:

$L = RL$
$\Rightarrow L^* = RL$



Questions:

$L = \phi$ $\quad L = a^*b^*$ $\quad (a^*b^*)^+$ $\quad L = \epsilon + a^+$ $\quad L = a \cdot a^+$
$\phi^* = \epsilon$ $\quad (a^*b^*)^*$ $\quad ((a^*b^*)^+)^*$ $\quad (\epsilon + a^+)^*$ $\quad (a \cdot a^+)^*$
$\Rightarrow RL$ $\quad = (a+b)^*$ $\quad = (a+b)^*$ $\quad = a^*$ $\quad = aaa^+$
$= a^+a^+$

$L = \epsilon^*$ $\quad (a+b)^*$ $\quad (aa^+)^*$
$(\epsilon^*)^* = \epsilon$ $\quad (a+b)^*)^* = (a+b)^*$ $\quad ((aa^+)^*)^* = (aa^+)^*$

Note: $(a^+)^+ = a^+$
$(a^*)^+ = a^*$

8. Subset: <Not closed under ∞ subset> <Closed under finite subset>

$L = (a+b)^* = RL$ $\quad a^n b^n = nRL$ $\quad$ **Note:** Subset of nRL may be RL
$L_1 = a^n b^n = nRL$ $\quad L_1 = ab = RL$
$L_2 = a^n b^n = nRL$

$L = (a+b)^* = RL$ $\quad$ finite subset $= ab$ $\quad L = a^n b^n = nRL$ $= ab$

Note: Finite subset of RL or nRL is always regular

9. Quotient: $L_1/L_2 = \{u \mid \exists uv \in L_1 \text{ and } v \in L_2\}$

$\frac{abc}{c} = ab$ $\quad \frac{abc}{b} = \phi$ $\quad \frac{abc}{bc} = a$ $\quad \frac{abc \cdot \epsilon}{\epsilon} = abc$ $\quad \frac{\epsilon \cdot \phi}{\phi} = \epsilon$ $\quad \frac{abc}{a\epsilon} = \phi$ $\quad \frac{\{a, ab\}}{\{\epsilon, b, aa\}}$ $\quad \frac{\{aaaaa\}}{\{a, aa, aaa\}}$

$\frac{a^*}{a} = \frac{a}{a}, \frac{a^2}{a}, \frac{a^3}{a} \cdots$
$= \epsilon, a, a^2 \cdots$
$= a^*$

$\frac{a}{a^*} = \frac{a}{\epsilon}, \frac{a}{a}, \frac{a}{a^2} \cdots$
$= a, \epsilon, \phi \cdots$
$= \{\epsilon, a\}$

$\frac{ba^*}{b} = \frac{b}{b}, \frac{ba}{b}, \frac{ba^2}{b} \cdots$
$= \epsilon, \phi, \phi$
$= \{\epsilon\}$

$= \{a, ab\}$ $\quad = \{aaaa, aaa, aa\}$

$\frac{L}{\phi} = \phi$ $\quad \frac{\phi}{L} = \phi$ $\quad \frac{\phi}{\phi} = \phi$

$\frac{a^*b}{ab} = \frac{b}{ab}, \frac{ab}{ab}, \frac{a^2b}{ab} \cdots$
$= \phi, \epsilon, a, aa \cdots$
$= a^*$

$\frac{0^*}{0^*} = \frac{0^*}{\epsilon}, \frac{0^*}{0}, \frac{0^*}{0^2} \cdots$
$= 0^*$

Note-1 If L_2 includes ϵ then $L_1/L_2 = $ at least L_1
$\because \frac{L_1}{\epsilon}, \frac{L_1}{\cdots} \cdots = \{L_1, \cdots\}$

Note-2 If L is non-empty then $\Sigma^*/L = \Sigma^*$ and $L/\Sigma^* =$ All prefixes of L

$\because$ Let $L = a$
$\frac{\Sigma^*}{L} = \frac{(a+b)^*}{a} = \frac{\epsilon}{a}, \frac{a}{a}, \frac{b}{a}, \frac{ab}{a} \cdots$
$= \epsilon, b, a, ab, ba \cdots = \Sigma^*$

$\because$ Let $L = abcd$
$\frac{L}{\Sigma^*} = \frac{L}{\epsilon}, \frac{L}{d}, \frac{L}{cd}, \frac{L}{bcd}, \frac{L}{abcd}$
$= abcd, abc, ab, a, \epsilon =$ All prefix of L

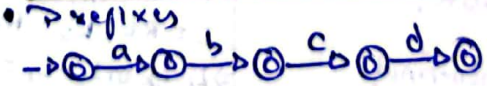
10. Prefix, Suffix, Substring:

All prefixes, suffixes are substrings

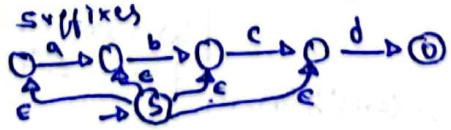
If $L = RL$
$\Rightarrow$ Prefix, Suffix, Substring of $L = RL$

Let, $L = \{abcd\}$
Prefixes $= \{\epsilon, a, ab, abc, abcd\}$
Suffixes $= \{\epsilon, d, cd, bcd, abcd\}$
Substring $= \{\epsilon, a, b, c, d, ab, bc, cd, abc, bcd, abcd\}$

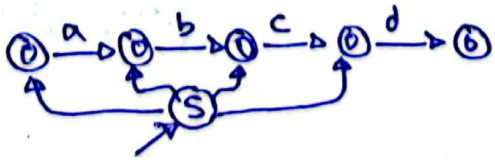
- Prefixes



- Suffixes



- Substring



Q: If for a string w Prefix, Suffix, Substring all are same, $w = ?$
w = any string of any length where $|\Sigma| = 1$
ex: a^*, bbb

11. Homomorphism: Substitution where symbol replaced by string

Here, Δ = Alphabet, Δ^* = String, $P(\Delta^*)$ = Language

we map every symbol from Σ to Δ^* ($\Sigma \to \Delta^*$)

note:
- $h(01) = h(0) \cdot h(1)$
- $h(1^*) = [h(1)]^*$
- $h(\epsilon) = \epsilon$
- $h(\phi) = \phi$
- $h(0+1) = h(0) + h(1)$

Q: $\Delta = \{a,b,c\}$, $\Sigma = \{0,1\}$
$h(0) = ab$, $h(1) = bbc$
$h(010) = ?$
$= h(0) \cdot h(1) \cdot h(0)$
$= ab \cdot bbc \cdot ab$

Q: $\Delta = \{a,b\}$, $\Sigma = \{0,1\}$
$L = (01)^* 11$
$h(0) = abb$, $h(1) = \epsilon$
$h(L) = ?$
$= h[(01)^* 11]$
$= [h(0) \cdot h(1)]^* \cdot h(1) \cdot h(1) = (abb)^*$

12. Substitution: Just like homomorphism

Symbol replaced by lang. $s: \Sigma \to P(\Delta^*)$

- $s(01) = s(0) \cdot s(1)$
- $s(0+1) = s(0) + s(1)$
- $s(0^*) = [s(0)]^*$
- $s(\epsilon) = \epsilon$
- $s(\phi) = \phi$

Q: $\Delta = \{0,1\}$, $\Sigma = \{a,b\}$
$s(a) = 0^*$
$s(b) = 0^*1$
$s(ab) = ?$
$= s(a) \cdot s(b)$
$= 0^* \cdot 0^*1$
$= 0^*1$

Q: $\Delta = \{a,b\}$, $\Sigma = \{0,1\}$
$f(0) = a$
$f(1) = b^*$
i) $f(010) = ?$
ii) $f(0^*(0+1)1^*) = ?$
i) $= f(0) \cdot f(1) \cdot f(0)$
$= ab^*a$
ii) $= [f(0)]^* [f(0)+f(1)][f(1)]^*$
$= a^*(a+b^*)b^*$
$= a^*b^*$

Q: $\Delta = \{0,1\}$, $\Sigma = \{a,b\}$, $L = (ab)^*$
$f(a) = 0^*$
$f(b) = \phi$
$f(L) = ?$
$= [f(ab)]^*$
$= [0^* \cdot \phi]^*$
$= \phi^*$
$= \epsilon$

13. Inverse homomorphism:-

String is replaced by symbol

$h^{-1}(\omega) = \{x \mid h(x) = \omega\}$

Q: $\Sigma = \{0,1\}$ $\Delta = \{a,b\}$
$L = (ab+ba)^*a$
$h(0) = aa$ $h(1) = aba$
$h^{-1}(L) = ?$
$h^{-1}(aa) = 0$, $h^{-1}(aba) = 1$
$h^{-1}(L) = [h^{-1}(ab) + h^{-1}(ba)]^* \cdot h^{-1}(a)$
Directly not possible
expand $L = (ab+ba)^*a$
$= a$ (not poss), aba (1), baa (not poss.), ...
$\Rightarrow h^{-1}(L) = 1$

Q: $\Sigma = \{0,1,2\}$ $\Delta = \{a,b\}$
$h(0) = a$, $h(1) = ab$, $h(2) = ba$
i) $h^{-1}(ababa) = ?$ ii) $h^{-1}(a(ba)^*) = ?$
i) $= h^{-1}(ab) \cdot h^{-1}(ab) \cdot h^{-1}(a) = 110$
or $= h^{-1}(a) \cdot h^{-1}(ba) \cdot h^{-1}(ba) = 022$
or $= h^{-1}(ab) \cdot h^{-1}(a) \cdot h^{-1}(ba) = 102$
ii) $= h^{-1}(a)[h^{-1}(ba)]^* = 02^*$
or $= h^{-1}((ab)^* \cdot a) = 1^*0$

Q: $L = \{abb, aab\}$; $h(0) = a$
$h(1) = ab$, $h(2) = b$
i) $h^{-1}(aab) = ?$ ii) $h^{-1}(abb) = ?$
i) $= h^{-1}(a) \cdot h^{-1}(ab) = 01$
or $= h^{-1}(a) \cdot h^{-1}(a) \cdot h^{-1}(b) = 002$
$\Rightarrow \{01, 002\}$
ii) $= h^{-1}(ab) \cdot h^{-1}(b) = 12$
or $= h^{-1}(a) \cdot h^{-1}(b) \cdot h^{-1}(b) = 022$
$\Rightarrow \{12, 022\}$
iii) $\Rightarrow h^{-1}(L) = \{01, 002, 022, 12\}$

Q: $\Sigma = \{a,b\}$ $\Delta = \{0,1\}$
$h(a) = 01$ $h(b) = 0$
i) $L_1 = (10+1)^*$
ii) $L_2 = (a+b)^*$
iii) $L_3 =$ equal no. of 0's & 1's
i) $= [h^{-1}(10) + h^{-1}(1)]^*$
Directly not possible
$L_1 = \epsilon, 1, 10, 110, \ldots$
$h^{-1}(L_1) = \epsilon, \times, \times, \times$
$\Rightarrow h^{-1}(L_1) = \epsilon$
ii) $= [h(a) + h(b)]^* = (01+0)^*$
iii) $= h^{-1}((01)^*) = a^*$

14. L/2 , L/3 :

$L = \{\epsilon, a, ab, aab, bcc\}$ $L = \{\epsilon, a, ab, a, aa, bb, cc\}$

Half(L) = L/2 ⇒ First(L/2) = {ϵ, a, aab} L/3 ⇒ {ϵ, a, aa}

Second Half(L) ⇒ {ϵ, b, bcc} middle(L/3) = {ϵ, b, bb}

last(L/3) = {ϵ, c, cc}

15. Symmetric Difference:

$A \Delta B \Rightarrow (A \cup B) - (A \cap B)$ or $(A-B) \cup (B-A)$

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

$L_1 \Delta L_2 = a^* \Delta a^* = \phi$

Questions:

$L_1 = a^*$, $L_2 = b^*$, $L_1 \Delta L_2$ ⇒ $a^+ + b^+$

$L_1 = a^*$, $L_2 = a^* b$, $L_1 \Delta L_2$ ⇒ $a^* + a^* b = a^*(\epsilon + b)$

*) Moore & Melay:

→ Characteristics of moore, melay :-

- They're deterministic, so covers all strings present in Σ^*
- Have zero final states, as they're language acceptors not language recognizers.
- If n-length i/p, moore → n+1 length o/p, melay → n length o/p

→ Specifications of moore, melay :-

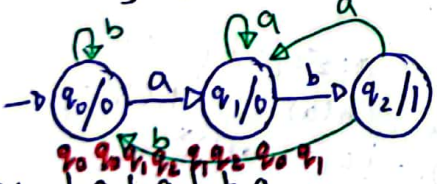
$m = (Q, \Sigma, \delta, S, \Delta, \lambda)$; Δ = o/p alphabet, λ = o/p fx^n (6 tuples)

Moore m/c :) $\lambda : Q \to \Delta$ (वर्तमान state से ही o/p decide होता)

Melay m/c :) $\lambda : Q \times \Sigma \to \Delta$ (state + transition से o/p decide हुआ)

→ Moore m/c construction :-

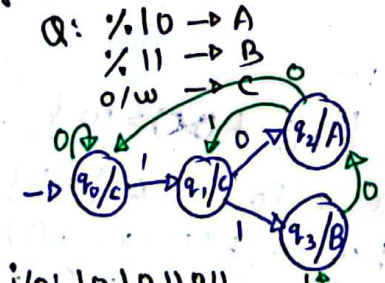
Q: Count no. of occurrences of substring ab



i/p: b a b a b b a

o/p: 0 0 0 1 0 1 0 0

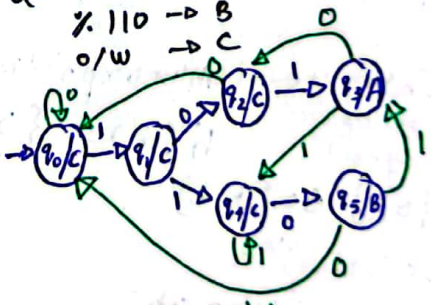
Q: %10 → A, %11 → B, o/w → C



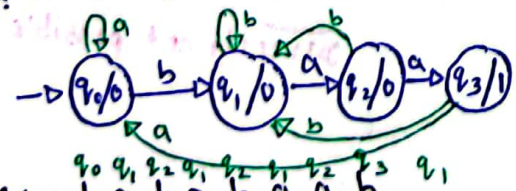
i/p: 1 0 1 0 1 1 0 1 1

o/p: C C A C A C B A C B

Q: %101 → A, %110 → B, o/w → C



Q: ... same ... baa



i/p: b a b a b a a b

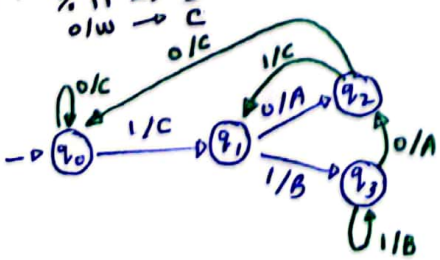
o/p: 0 0 0 0 0 0 1 0

Note: for moore, if ϵ is i/p, the o/p would be o/p of first state

States	Inputs 0	1	outputs
→ q_0	q_0	q_1	C
q_1	q_2	q_3	C
q_2	q_0	q_1	A
q_3	q_2	q_3	B

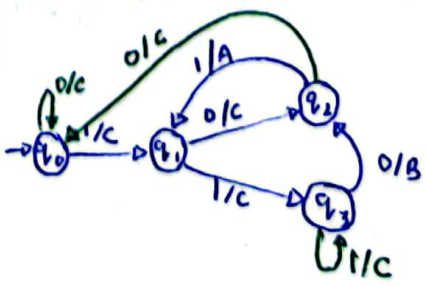
→ Melay m/c construction :-

Q-1) %10 → A, %11 → B, o/w → C



States	i/p=0	o/p	i/p=1	o/p
→ q_0	q_0	C	q_1	C
q_1	q_2	A	q_3	B
q_2	q_0	C	q_1	C
q_3	q_2	A	q_3	B

Q:2) %101 → A, %110 → B, o/w → C

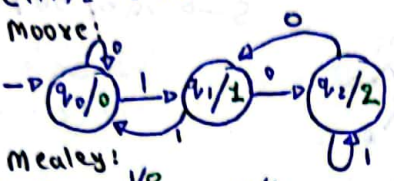


Q-3) Construct moore & mealey that produces residue mod 3 as o/p, where i/p is binary no.
i.e. $M \% 3 \Rightarrow 0, 1, 2$?

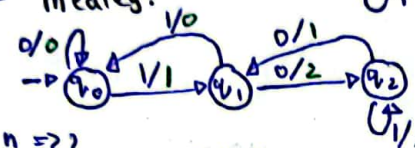
Samples:) i/p — o/p
$(1010)_2 = (10)_{10} \Rightarrow 10 \% 3 = 1$
$(111)_2 = (7)_{10} \Rightarrow 7 \% 3 = 1$

remainder	0	1
→ q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Moore:



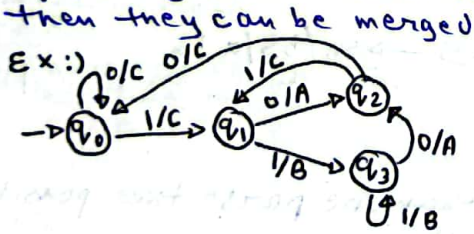
Mealey:



Note:) $m \% n \Rightarrow ?$
m ealey & moore → n states

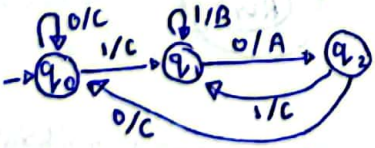
→ Minimization of mealey:

- Take two strings such that their ending is same
  string-1 → x z
  string-2 → y z
- Parse string 1 on one state & string 2 to other state.
- If both generate same o/p, then they can be merged.
- Ex:)



consider q_2 & q_3
string-1 → 101 → q_2 → C
string-2 → 001 → q_3 → C
⇒ merge q_2 & q_3

⇒ minimized mealey:-



- Minimized mealey is unique.

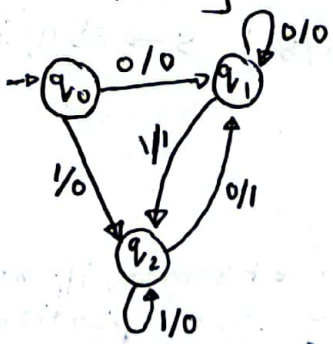
→ Moore to Mealey conversion:

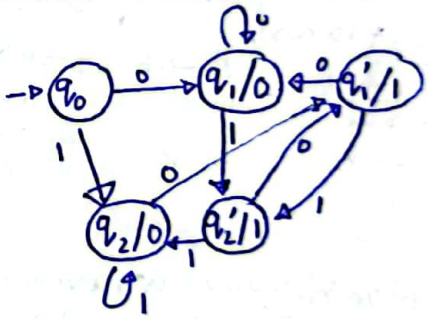
Present State	i/p = 0	i/p = 1	o/p
moore → q_0	q_1	q_0	A
q_1	q_2	q_0	B
q_2	q_0	q_1	C

q_0 पे 0 पर q_1 पे गया & q_1 का o/p है B
q_0 पे 1 पर q_0 पे गया & q_0 का o/p है A

Present State	i/p = 0	o/p	i/p = 1	o/p
→ q_0	q_1	B	q_0	A
q_1	q_2	C	q_0	A
q_2	q_0	A	q_1	B

→ Mealey to moore conversion:





m state n o/p mealy → (m x n) max. m state moore

note:)
T-oe ⇒ moore to mealey Table is required
D-eo ⇒ mealey to moore Diag. is required

#> Grammar :-

*> Basics:

G(V, T, P, S)

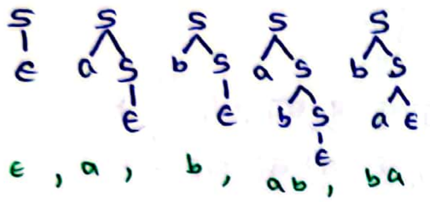
- V = variables
- T = Terminals
- P = Productions
- S = start symbol

Ex: V = {S, A, B}, T = {a, b}

P = { S → AB, A → a, B → b }, S = {S}

*> Grammar → Language:

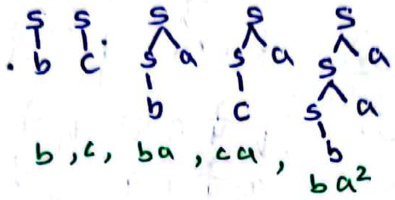
S → aS / bS / ε



ε , a , b , ab , ba

= (a+b)*

S → Sa / b / c



b , c , ba , ca , ba²

= (b+c) a*

S → Sab / Sd / ef

= ef (ab+d)*

*> Language → Grammar:

L = (baa+abb)*

S → baaS / abbS / ε

*> RG (LLG, RLG): If G is LLG or RLG, G is RG

LLG: V → VT* / T*

RLG: V → T*V / T*

Note:
- For every LLG, equivalent RLG possible.
- RL ✓ ⇒ DFA ✓, NFA ✓, ε-NFA ✓; Regexp ✓, LLG ✓, RLG ✓, RG ✓

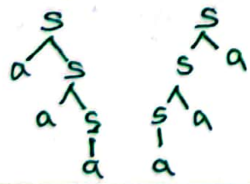
Questions:

- S → AB, A → a, B → b: LLG ✗, RLG ✗] RG ✗
- S → ε: LLG ✓, RLG ✓, RG ✓
- S → Sa / bcd: LLG ✓, RLG ✗, RG ✓
- S → eSa / bcd: LLG ✗, RLG ✗, RG ✗
- S → aS / bcdS / ε / abcde (RLG, RLG, RLG/LLG, RLG/LLG): RLG ✓, LLG ✗, RG ✓
- S → Sa / bS / a / b (LLG, RLG, RLG/LLG, RLG/LLG): RLG ✗, LLG ✗] RG ✗

*> Ambiguous grammar: • From G, if more than one parse tree possible for a particular string w.

- No Algo. for checking Ambiguity (undecidable problem)
- Ex: S → aS / Sa / a

w = aaa



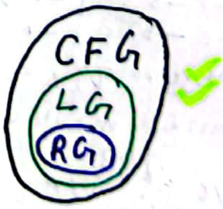
*> Linear Grammar (LG)

V → T*VT* / T*

*> CFG:

V → (V+T)*

i.e. LHS single variable, RHS anything



Questions:

- S → abScd / ab: RG ✗, LG ✓ ⇒ CFG ✓
- S → aS / bS / d: RLG ✓ ⇒ RG ✓ ⇒ LG ✓ ⇒ CFG ✓
- S → Sa / aS / ε: RG ✗, LG ✓ ⇒ CFG ✓
- A → Aa, B → bB: RG ✗, LG ✓ ⇒ CFG ✓
- S → aSbS / bSaS / ε: RG ✗, LG ✗, CFG ✓

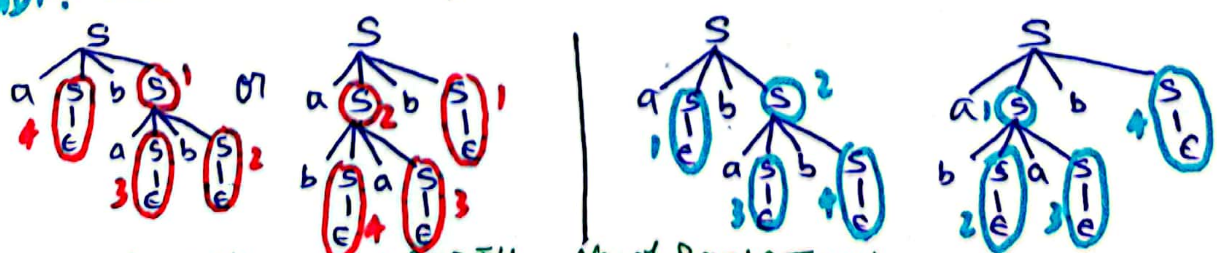
*> LMDT, RMDT:

RMDT: when we're substituting rightmost variable first

LMDT: ———— leftmost ————

Ex: S → aSbS / bSaS / ε

i/p: abab



No. of LMDT's = No. of RMDT's = No. of Parse Trees

Note:) when only 1 variable at every level, the tree itself is RMD & itself is LMD
Ex: $S \to aS/Sa/a$; $w = aaa$

*) Language → CFG construction:

$L = \{a^m b^n \mid m,n \geq 0\}$
$= \{\epsilon, a, b, ab, abb, aab, \dots\}$
$= a^* b^*$

$S \to AB$
$A \to aA/\epsilon$
$B \to bB/\epsilon$

$L = \{a^m b^n c^n \mid m,n \geq 0\}$

$S \to A \cdot B$
$A \to aA/\epsilon$
$B \to bBc/\epsilon$

For L; $S \to aSb/\epsilon$
For L*; ? $S \to aSb/\epsilon/SS$

$L = \{a^i b^j \mid i \neq j ; i,j \geq 1\}$
$i > j$ or $j > i$

$S \to AB/CA$
$a^i b^i b^+ \quad a^+ a^i b^i$
$A \to aAb/ab$
$B \to b/bB$
$C \to a/aC$

$L = \{a^i b^j c^k \mid j = i+k ; i,j,k \geq 1\}$

$S \to AB$
$a^i b^j b^k c^k$
$A \to aAb/ab$
$B \to bBc/bc$

$L = \{ww \mid w \in (a+b)^*\}$
= {All odd lengths, even length strings of type: $w_1 \neq w_2$}
= {a, b, aab, aba, baa ..., ab, ba, aabb, bbaa ...}

$S_1 \to a/b/aaS_1/bbS_1/abS_1/baS_1$
$S \to AB/Ba/S_1$
$A \to aAa/aAb/bAa/bAb/a$
$B \to aBa/aBb/bBa/bBb/b$

S = start pattern समझ में आ

$L = \{b^* a b^* a b^*\}$

$S \to AaAaA$
$A \to bA/\epsilon$

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

$S \to AbbbA$
$A \to aA/bA/\epsilon$

$L = \{a^n b^n \cup b^n a^n \mid n \geq 1\}$

$S \to A/B$
$A \to aAb/ab$
$B \to bBa/ba$

$L = \{a^i b^j c^k \mid i \neq j \text{ or } j \neq k ; i,j,k \geq 1\}$

$i < j \Rightarrow a^n b^n b^+$
$i > j \Rightarrow a^+ a^n b^n$
$j < k \Rightarrow b^n c^n c^+$
$j > k \Rightarrow b^+ b^n c^n$

$A \to aA/a \Rightarrow a^+$
$B \to bB/b \Rightarrow b^+$
$C \to cC/c \Rightarrow c^+$
$D \to aDb/ab \Rightarrow a^n b^n$
$E \to bEc/bc \Rightarrow b^n c^n$

$S \to DBC/ADC/AEC/ABE$

L = {All even length pallindromes}
$S \to aSa/bSb/\epsilon$

L = {All odd length pallindromes}
$S \to aSa/bSb/a/b$

L = {All pallindromes}
$S \to aSa/bSb/a/b/\epsilon$

L = {equal no. of a's & b's}
$S \to aSbS/bSaS/\epsilon$
or $S \to aSb/bSa/SS/\epsilon$

L = {a's are twice of b's}
$S \to aSaSbS/bSaSa/aSbSa$

L = {set of all regex over $\epsilon = \{a,b\}$}
$R \to \epsilon/a/b/R+R/R \cdot R/R^*/R^+/(R)$

L = {All arithmetics over a, b}
$A \to A+A/A-A/A/A/A*A/(A)/a/b$

L = {all Gmail ids
$G \to L(L+D+S)^*$@gmail.com
$L \to A \dots Z/a \dots z$
$D \to 0 \dots 9$
$S \to \$/\#/\_/\dots$

L = {$a^n b^n$ | $n \geq 1$} Relation there, so handle with single variable
$S \to aSb/ab$

L = {$a^n b^n$ | $n \geq 0$}
$S \to aSb/\epsilon$

L = {set of all valid balanced parenthesis}
Ex: ((())), ()()(), (()())
$S \to (S)/SS/\epsilon$

L = {$(baa+abb)^*$}
$S \to baaS/abbS/\epsilon$

*) CFG simplification :-
1. Null production elimination
2. Unit " "
3. Useless symbol " "
Do simplification in this order

3.) Useless symbol elimination (Reduced grammar)
Step-1: Eliminate unreachable variable
Step-2: Eliminate productions which can't derive any string

Q: ✓S → AB
✓A → a
✓B → b/C ✗
E → b/ϵ ✗
✗ C → cC/BC

Result:) S → AB
A → a
B → b

Q: S → AB/CA ✗
B → BC/AB ✗
A → a
C → aB/b ✗

S → CA
A → a
C → b

Q: S → aAa ✓
A → Sb/bCC/DaA ✗
C → D/abb/DD ✗
E → aC ✗
D → aDA ✗

S → aAa
A → Sb/bCC
C → abb

2.) Unit production elimination: (single var → single var is unit production)

Q: S → Aa/B
B → A/bb
A → a/bc/B

S → Aa/a/bc/bb
B → bb/a/bc
A → a/bc/bb

Q: S → AB
A → a
B → C/b
C → D
D → E
E → a

S → AB
A → a
B → a/b
C → a
D → a
E → a

1.) Null production elimination: (A → ϵ or A ⇒ ϵ)

Note: After null production elimination we get equivalent grammar if grammar not giving epsilon, but we don't get equivalent grammar if G contain null production

Q: S → aS/AB
A → ϵ ✗
B → ϵ ✗ direct null
D → b

S → aS/AB/A/B
D → b

Ex: S → a/Xb/aYa
X → Y/ϵ ✗
Y → b/X

This grammar not generating ϵ, so, we'll get equivalent grammar.
S → a/Xb/aYa/b/aa
X → Y
Y → b/X

Q: S → aX/bX
X → a/b/ϵ ✗
S → aX/bX/a/b
X → a/b

Q: S → aX/bS/a/b
X → aX/a/ϵ ✗
S → aX/bS/a/b
X → aX/a

Q: S → ABaC
A → BC
B → ϵ ✗
C → ϵ ✗
D → d

S → ABaC/AaC/ABa/Aa/BaC/aC/a/Ba
A → BC/B/C
D → d

*) Elimination of left recursion:

Recursion

aSb (Sa) aS

Rec. LR RR

Ex: S → Sa / b L = {b, ba, baa, ...} = ba*

a is recursive, replace with S'

replace S with non-recursive terms

S → bS'

S' → aS' / ∈

Ex: S → Sa / b / c / d

S → bS' / cS' / dS'

S' → aS' / ∈

*) Normal forms :-

1) CNF

- cond.n-1: CFG ✓ (V → (V+T)*)
- cond.n-2: V → V1V2 / T ; i.e. 2 variable or single terminal
- To generate 'n' length string from given CNF ⇒ 2n-1 produc.ns

2) GNF

- cond.n-1: CFG ✓
- cond.n-2: V → TV*
- To generate 'n' length string from given GNF ⇒ n productions

*) CFG → CNF

Ex: S → bA / aB

A → bAA / aS / a

B → aBB / bS / b

step-1: CFG simplification

i) ✓ ii) ✓ iii) ✓

step-2: conversion to CNF

S → CbA / CaB

Cb → b

Ca → a

A → CbD / CaS / a

D → AA

B → CaE / CbS / b

E → BB

Note:) For every ∈-free CFG, an equivalent CNF is possible.

Ex: S → aAD

A → aB / bBB

B → b

D → d

step-1 ✓

step-2

S → EF

E → a

F → AD

A → EB / BG

G → BB

B → b

D → d

*) CFG → GNF

Ex: S → aSb

S → ab

step-1: CFG simplification

step-2: conversion to GNF

S → aSB

B → b

S → aB

Ex: S → ∈ ✗

step-1: ✓ ⇒ ϕ

step-2: ϕ < CNF ✓ / GNF ✓

Ex: S → AA / a

A → SS / b

Left recursion there so solve differently

① Rename variables

S = A1

A = A2

A1 → A2A2 / a

A2 → A1A1 / b

② If Am → An ; if m > n then substitute

A1 → A2A2 / a A2'

A2 → A2A2A1 / a A1 / b

eliminate left recursion

A1 → A2A2 / a

A2 → aA1A2' / bA2' / aA1 / b

A2' → A2A1 / A2A1A2'

not in GNF so, substitute again

A1 → aA1A2'A2 / bA2'A2 / aA1A2 / bA2 / a

A2 → aA1A2' / bA2' / aA1 / b

A2' → aA1A2'A1 / bA2'A1 / aA1A1 / bA1 / aA1A2'A1A2' / bA2'A1A2' / aA1A1A2' / bA1A2'

now it is in GNF

Ex: $S \to AB$, $A \to BS/b$, $B \to SA/a$ — left rec. there so solve differently

① variable rename

$S = A_1$, $A = A_2$, $B = A_3$

$A_1 \to A_2A_3$

$A_2 \to A_3A_1/b$

$A_3 \to A_1A_2/a$

② $\forall n\ A_m \to A_n$, if $m>n$ then substitute

$A_1 \to A_2A_3$

$A_2 \to A_3A_1/b$

$A_3 \to A_2A_3A_2/a$,

↳ $A_3 \to A_3A_1A_3A_2/bA_3A_2/a$ ($A_3A_1A_3A_2$ is left rec.)

↳ Eliminate left rec.

↳ $A_3 \to bA_3A_2A_3'/aA_3'/bA_3A_2/a$

$A_3' \to A_1A_3A_2/A_1A_3A_2A_3'$

finally,

$A_1 \to A_2A_3$

$A_2 \to A_3A_1/b$

$A_3 \to bA_3A_2A_3'/aA_3'/bA_3A_2/a$

$A_3' \to A_1A_3A_2/A_1A_3A_2A_3'$

not in GNF so substitute again

$A_2 \to bA_3A_2A_3'A_1/aA_3'A_1/bA_3A_2A_1/aA_1/b$

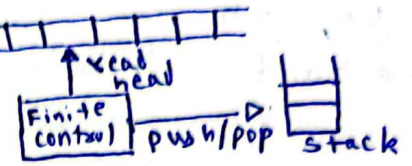
$A_1 \to bA_3A_2A_3'A_1A_3/aA_3'A_1A_3/bA_3A_2A_1A_3/aA_1A_3/bA_3$

$A_3 \to bA_3A_2A_3'/aA_3'/bA_3A_2/a$

$A_3' \to bA_3A_2A_3'A_1A_3A_3A_2/aA_3'A_1A_3A_3A_2/bA_3A_2A_1A_3A_3A_2/aA_1A_3A_3A_2/bA_3A_3A_2/bA_3A_2A_3'A_1A_3A_3A_2A_3'/aA_3'A_1A_3A_3A_2A_3'/bA_3A_2A_1A_3A_3A_2A_3'/aA_1A_3A_3A_2A_3'/aA_1A_3A_3A_2A_3'/bA_3A_3A_2A_3'$ now it is in GNF

PDA

* PDA machine :-
  - PDA = FA + 1 stack
  - i/p tape



- $M = (Q, \Sigma, \delta, q_0, F, \Gamma, Z_0)$ = 7 tuples

  Γ = stack alphabet $Z_0 \in \Gamma$

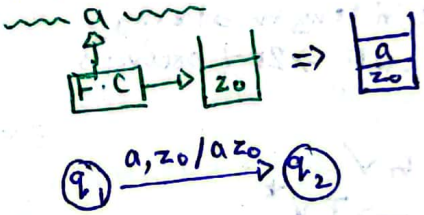
  Z_0 = initial TOS

- DPDA: $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$

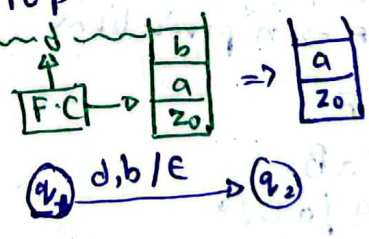
  NPDA: $\delta : Q \times \Sigma \times \Gamma \to P(Q \times \Gamma^*)$

- NPDA power > DPDA power
- Stack operations :-

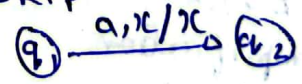
1) push



2) Pop



3) skip



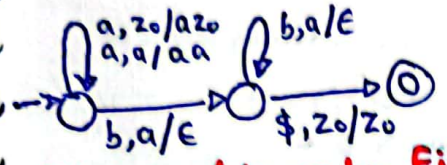
- PDA acceptance :-
  1) By final state
  2) By empty stack
  3) By both
- PDA constr.n :
  1) No. of states matter? No
  2) Every valid string logic? yes
  3) — invalid — ? No
- complementation not possible in PDA
- DPDA ✓ ⇒ DCFL ✓ ⇒ CFL

*) PDA construction:-

1) $L = \{a^n b^n \mid n \geq 1\}$

a a a a b b b b $

Logic: push for a's
pop for b's

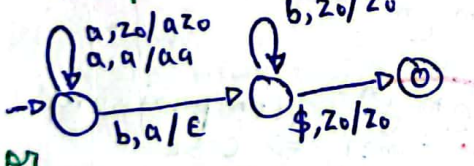


PDA acceptance by Final state

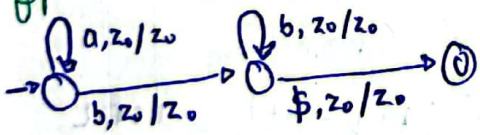
PDA acceptance by empty Stack

2) $a^m b^n \mid m,n \geq 1$

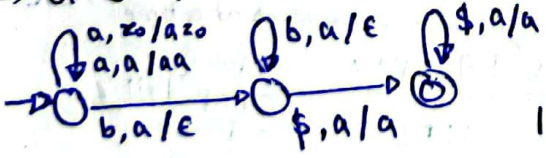
Logic: push for 1st a
skip for rest of a's
pop for 1st b
skip for rest of b's



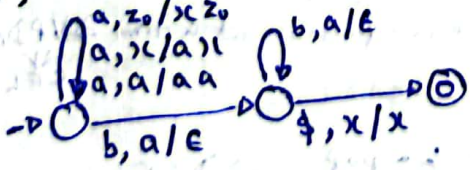
OR



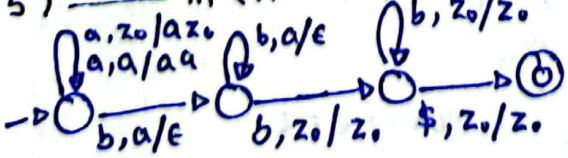
3) $a^m b^n \mid m > n;\ m,n \geq 1$



4) —— $m = n+1$ ——



5) —— $m < n$ ——

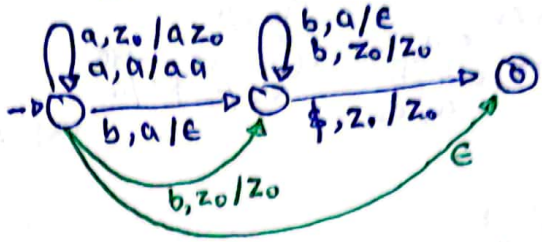


11) $wcw^R \mid w \in \Sigma^+$

Same like 10 just this won't be there

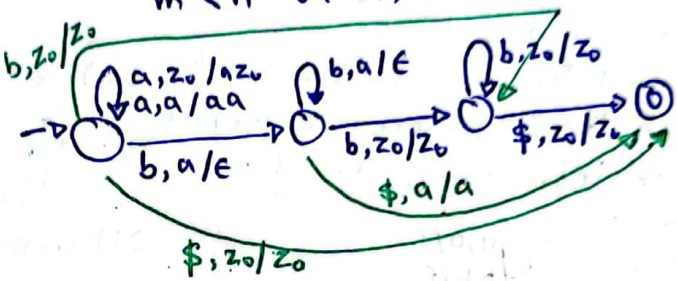
6) —— $m \leq n$; $m,n \geq 0$

$m == n$ or $m < n$



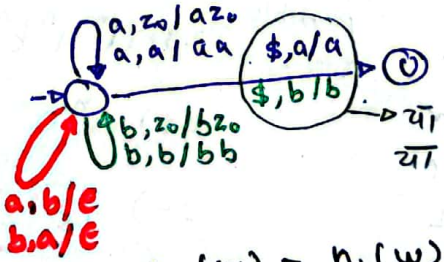
7) —— $m \neq n$; $m,n \geq 0$

$m < n$ or $m > n$



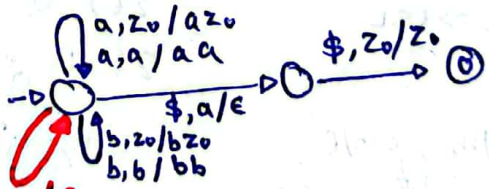
8) $w \mid w \in \Sigma^*;\ n_a(w) \neq n_b(w)$

$n_a(w) < n_b(w)$ or $n_a(w) > n_b(w)$



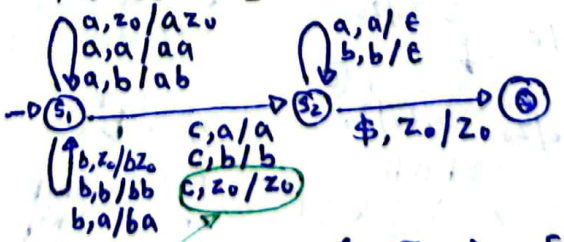
या तो a's बचे हों
या b's बचे हों

9) $n_a(w) = n_b(w) + 1$



10) $wcw^R \mid w \in \Sigma^*$

i.e. odd length pallindrome

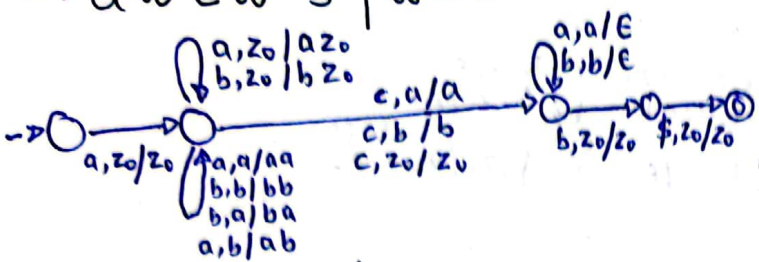


Transition fx^n:-

$S = \{S_1\}$
$Q = \{S_1, S_2, S_3\}$
$F = \{S_3\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b\}$

$\delta \Rightarrow \delta(S_1, a, Z_0) = (S_1, aZ_0)$
$\delta(S_1, a, a) = (S_1, aa)$
$\delta(S_1, a, b) = (S_1, ab)$
$\delta(S_1, b, Z_0) = (S_1, bZ_0)$
$\delta(S_1, b, b) = (S_1, bb)$
$\delta(S_1, b, a) = (S_1, ba)$
$\delta(S_1, c, a) = (S_2, a)$
$\delta(S_1, c, b) = (S_2, b)$
$\delta(S_1, c, Z_0) = (S_2, Z_0)$
$\delta(S_2, a, a) = (S_2, \epsilon)$
$\delta(S_2, b, b) = (S_2, \epsilon)$
$\delta(S_2, \$, Z_0) = (S_3, Z_0)$

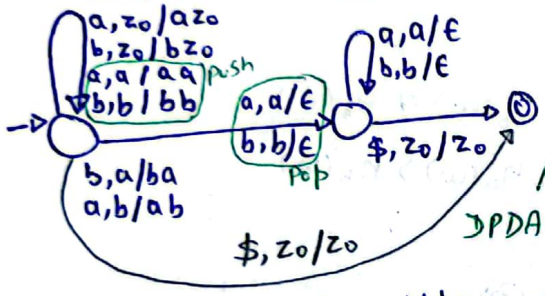
12) $a w c w^R b \mid w \in \Sigma^*$



13) $w w^R \mid w \in \Sigma^*$

i.e. even length palindromes

Logic: If two consecutive symbols same, draw a line b/w them
before that line is w, afterwards w^R
[Although the 2 consecutive symbols may both belong to w itself]



NPDA

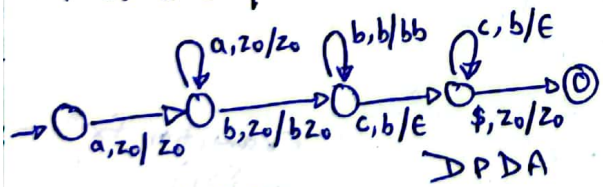
DPDA not possible

Note:) If $L \Rightarrow$ NPDA possible
$\Rightarrow$ CFL ✓ $\not\Rightarrow$ DCFL ✓

If $L \Rightarrow$ DPDA possible
$\Rightarrow$ DCFL $\Rightarrow$ CFL $\Rightarrow$ CSL

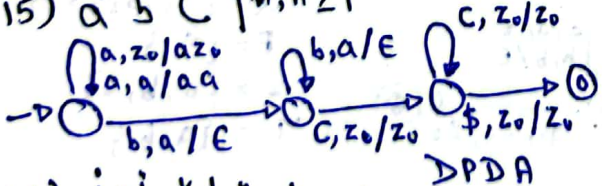
If $L \Rightarrow$ Regular
$\Rightarrow$ DCFL $\Rightarrow$ CFL $\Rightarrow$ CSL

14) $a^m b^n c^n \mid m,n \geq 1$



DPDA

15) $a^m b^m c^n \mid m,n \geq 1$

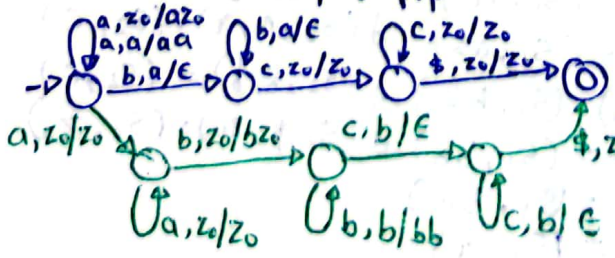


DPDA

16) $a^i b^j c^k \mid i=j$ or $j=k$; $i,j,k \geq 1$

a-push
b-pop
c-skip

a-skip
b-push
c-pop



17) $a^m b^n c^n d^m$ a-push, b-push, c-pop (for every b), d-pop (for every a)

DPDA ✓

18) $w w^R x \mid w, x \in \Sigma^*$

put $w = \epsilon$
so, x cover everything
$\Rightarrow$ RL $\Rightarrow$ DCFL

19) $x c w w^R \mid w, x \in \Sigma^*$

same.

20) $x c w w^R \mid x \in (a+b)^*, w \in (c+d)^*$

skip NPDA

NPDA $\Rightarrow$ CFL

21) $ww \mid w \in \Sigma^*$ abb|abb

DPDA ✗ NPDA ✗ CSL ✓

1 stack insufficient

22) $a^{m+n} b^m c^n$ a-push, b-pop, c-pop $\Rightarrow$ DPDA

23) $a^m a^n b^m c^n$ same.

24) $a^n a^m b^m c^n$ same

Note: checking formal & actual parameters

25) $a^m b^n c^m d^n$ $n,m \geq 1$ 2 comparisons $\Rightarrow$ CSL

26) $a^m b^m c^n d^n$ DPDA

27) i) $a^p b^q c^r d^t \mid p+q = r+t$ DCFL

ii) $a^p b^q c^r \mid p < q < r$ 2 comparisons at a time $\Rightarrow$ CSL

iii) $a^p b^q c^r \mid q = p+r$ a-push, b-pop till z_0, b-push, c-pop

DCFL $\Leftarrow$

iv) $a^p b^q c^r \mid p < q$ or $q < r$

one comparison at a time $\Rightarrow$ NPDA $\Rightarrow$ CFL

v) $a^p b^q c^r \mid p = 3(q+r)$

1st a - skip
2nd a - skip
3rd a - push
$\Rightarrow$ push for every 3rd a

b - pop
c - pop
$\Rightarrow$ DCFL

vi) $w \mid w^R$ push | pop NPDA $\Rightarrow$ CFL

vii) $w c w^R$ push skip pop $\Rightarrow$ DCFL

viii) $a^n b^{2n} \cup a^n b^{3n} \cup a^n b^{4n}$
push pop for every 2nd b; push pop for every 3rd b; push pop for every 4th b
NPDA ⇒ CFL

28) $w\,x\,w^R \mid w \in (a+b)^+,\ x \notin \{a,b\}$
push skip pop; x is marker
⇒ DCFL

29) $w\,x\,w^R \mid w, x \in (a+b)^+$
Put w = a or b
$a(a+b)^+a + b(a+b)^+b$
⇒ RL ⇒ DCFL

30) i) $w\,x\,w \mid w \in (a+b)^*,\ x \notin \{a,b\}$
CSL

ii) $w\,x\,w^R \mid w \in (a+b)^*,\ x \notin \{a,b\}$
DCFL; x = marker

iii) $w^R x w$ ~~Same~~
DCFL

iv) $w\,x\,w \mid w, x \in (a+b)^*$
$a(a+b)^*a + b(a+b)^*b$
⇒ RL ⇒ DCFL

Note:)
a,b/ε, a,b/b
NPDA

a,ε/b
NPDA

• $a^n b^n \cup a^n b^{2n} \mid n \geq 1$
e,ε/ε; c,ε/ε; $a^n b^n$; $a^n b^{2n}$
NPDA

• $c a^n b^n \cup a^n b^{2n} \mid n \geq 1$
c; a; $a^n b^n$; $a^n b^{2n}$
DPDA

• $c a^n b^n \cup d a^n b^{2n} \cup a^n b^{3n} \mid n \geq 1$
⇒ DCFL

K) Closure properties of DCFL & CFL
• If L_1 = CFL; L_2 = RL then $L_1 \cap L_2 = L_1 - L_2$ ⇒ CFL
• If L = CFL (not DCFL)
add ss for L^+
add ss/ε for L^*
S → aSb/ab — $a^n b^n$
S → aSb/ab/SS — $(a^n b^n)^+$
S → aSb/ab/SS/ε — $(a^n b^n)^*$
• For Subfix & Substring
DCFL ⇒ not closed
CFL ⇒ closed

Note:)
a,ε/a means, on i/p symbol 'a' TOS could be anything I'm pushing 'a' before & after same means SKIP
NPDA
a,z0/az0, a,a/aa
DPDA

a,ε/ε on i/p 'a' TOS anything I'm doing skip operation

a,any/ε on i/p 'a' TOS anything I'm doing pop operation

ε,ε/ε without reading any i/p TOS could be anything do skip operation

DPDA	operation	NPDA
a,any/a any	push	a,ε/a
a,any/any	skip	a,ε/ε
a,any/ε	pop	a,any/ε

31) $a^m b^n c^k d^l \mid \text{if } (m==n) \text{ then } (k==l)$
DCFL

32) $a^m b^n c^k d^l \mid \text{if } (n==k) \text{ then } (m==l)$
Push Push Pop Pop
DCFL

L = CFL? ∀ DCFL?

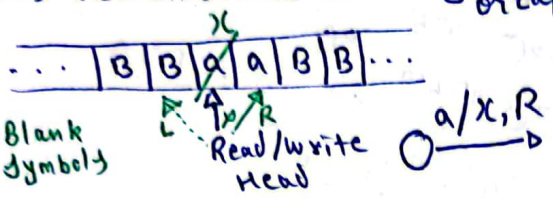
Infinite
- non-RL
  - non-CFL: ww, a^{n^2}, $a^{n!}$, $a^p \mid p = \text{prime}$, $a^n b^n c^n$
  - CFL (not DCFL): ww^R, $a^n b a^n b$, $a^n b a^{2n}$
- RL → ACFL → CFL: $a^n b^n \mid n \geq 1$, $wcw^R \mid w \in \Sigma^*$, $a^n b^{2n} \mid n \geq 1$, $(a+b)^* \mid a^n b^n \mid n \geq 1$

Finite
- RL → ACFL → CFL: $a^n b^n \mid n \geq 1000$, $a^p \mid p = \text{prime} \mid p \leq 1000$

#) TM :

*) Turing m/c :

- TM = FA + 2 Stack or PDA + 1 Stack | Tape = 2 Stack
- TM's remembrance is through state or tape.

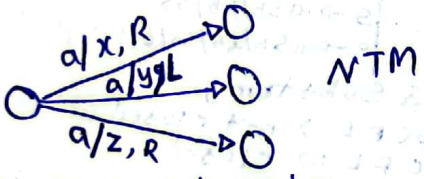


- Turn around capability: TM can move in both dir^n's & read/write also possible.
- $M = (Q, \Sigma, \delta, q_0, F, B, \Gamma)$ B = Blank symbol, Γ = Tape alphabet

7 tuple

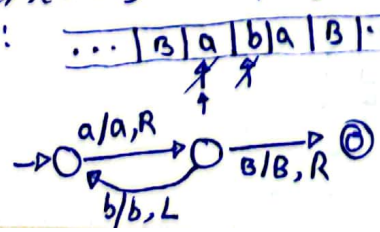
$B \in \Gamma$, $\Sigma \subseteq \Gamma$

- DTM :) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
  NTM :) $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$



- TM can work as :-
  1) Language recognizer
  i/p String → TM → yes / No
  2) Enumerator (Series generator)
  i/p n → TM → Produce Primeno's before n
  3) Transducer (o/p generator)
  i/p 5, 7 → TM → o/p 35
- DTM power = NTM power
  Interconversion also possible
- For i/p String 'x' TM have 3 possibilities :-
  1) TM go to Halt Final
     ⇒ x = valid
  2) TM go to Halt Non-Final
     ⇒ x = invalid
  3) TM go to ∞ loop
     ⇒ x = maybe valid/may be invalid

Ex :



- TM Languages

Recursive L

∀ valid string
TM - Halt final

∀ invalid string
TM - Halt non final

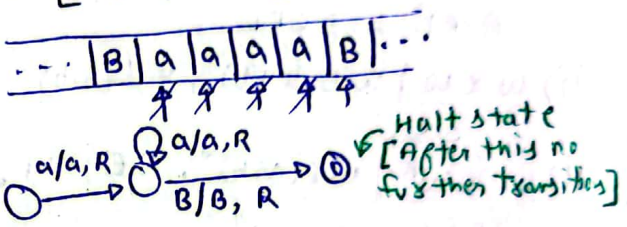
REL

x - valid = Halt Final
x - invalid = Halt non final
x - valid/invalid = ∞ loop

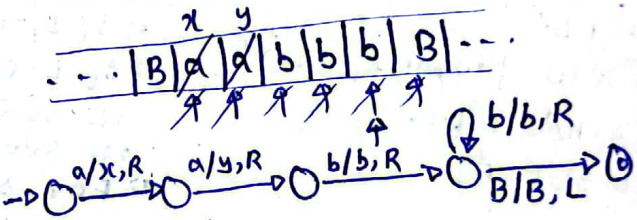
Rec L ⊆ REL

*) Language recognizer :

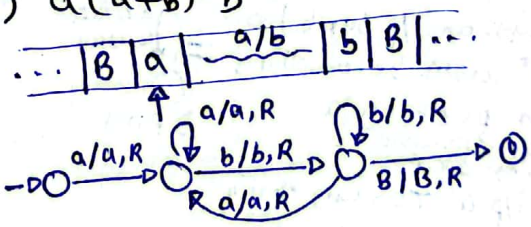
1) $L = \{ a^n \mid n \geq 1 \}$



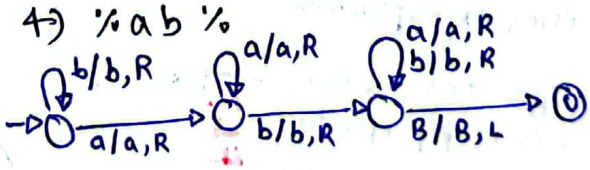
2) $aab^n \mid n \geq 1$



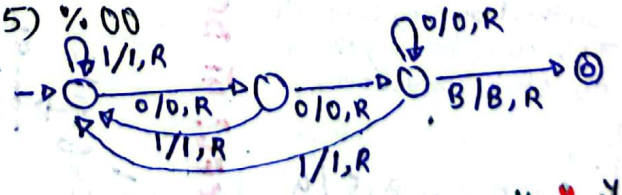
3) $a(a+b)^*b$



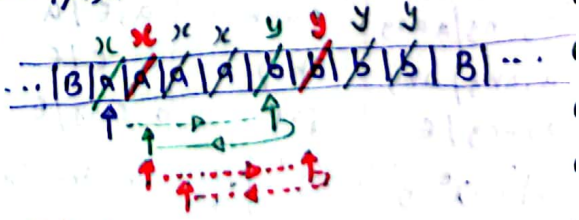
4) % a b %

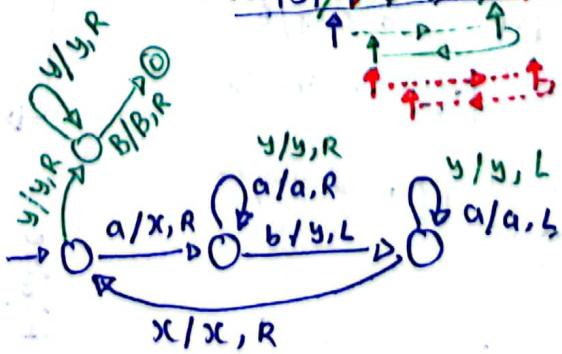


5) % 00

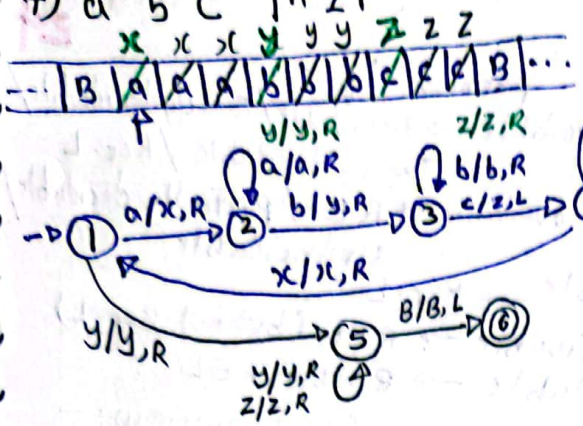


6) $a^n b^n \mid n \geq 1$



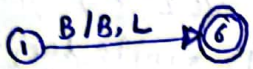


7) $a^n b^n c^n \mid n \geq 1$

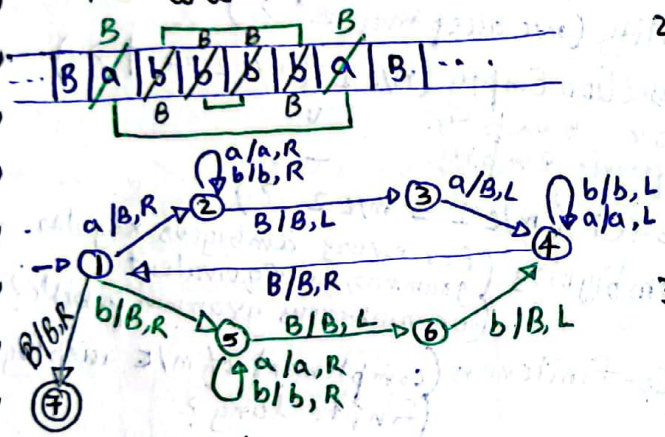


8) _____ $n \geq 0$

Same just add

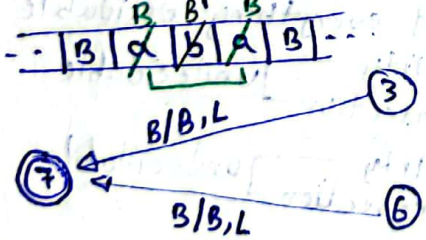


9) All even palindromes of a,b i.e. ww^R

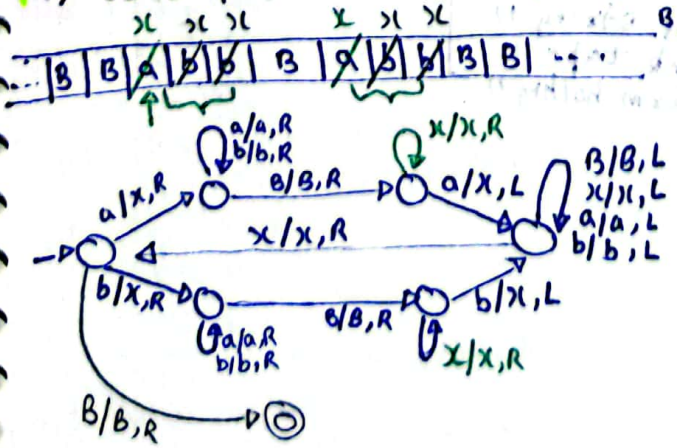


10) All palindromes

Same like 9 just add odd length palindromes transitions



11) $ww \mid w \in \Sigma^*$.



*) Some undecidable problems of TM:-

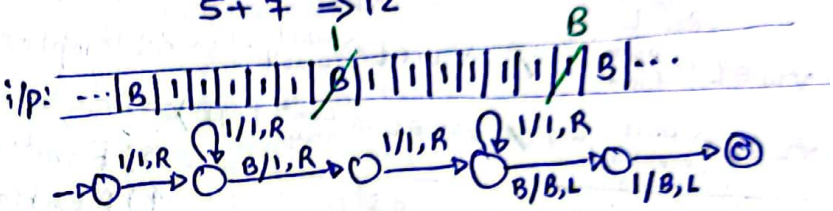
1) Halting problem: On given turing m/c TM & given string w, TM will accept 'w' or not is an undecidable problem, cuz. TM may go to infinite loop.

2) State entry problem: After reading 'w', TM will be at state no. x (say) or not is undecidable.

*) Transducer:-

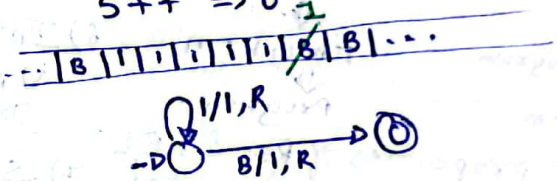
1) Add two +ve non-zero integers

5 + 7 ⇒ 12

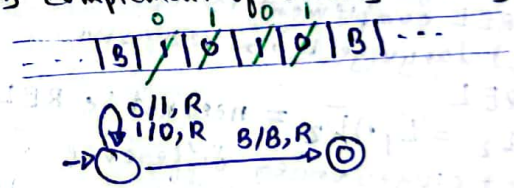


2) Increment fx^n for I+

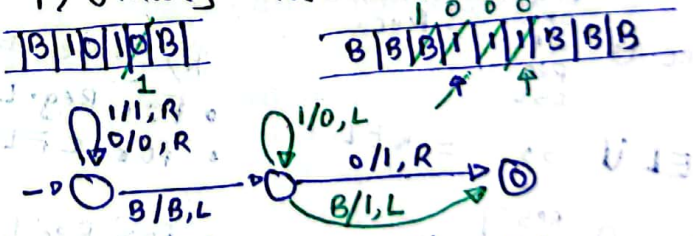
5++ ⇒ 6



3) 1's complement of binary string

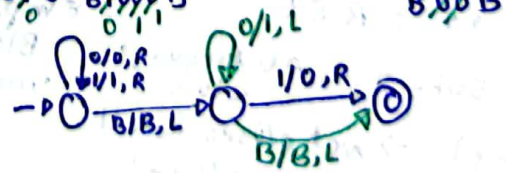


4) Binary incrementation

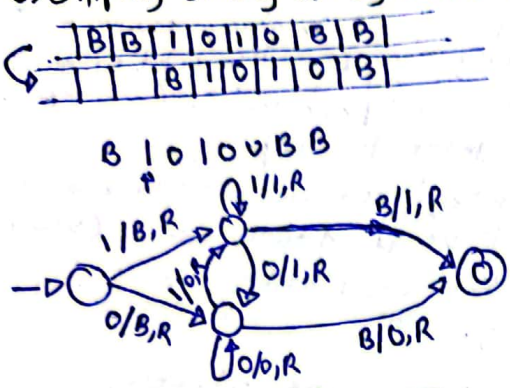


5) Binary decrementation:

Just [illegible] bit combination



6) Shifting binary no. by 1 cell right



Note:) TM → Rec L, REL

$\forall w \in L \Rightarrow$ Halt final Logic ✓ $\forall w \in L \rightarrow$ Halt final Logic ✓

$\forall w \notin L \Rightarrow$ Halt non-final Logic ✓ $\forall w \notin L \Rightarrow$ can't say Logic ✗

Recursive	REL
⇓	⇓
Halting TM	TM
⇓	⇓
Halting Program	TM or HTM
⇓	⇓
Algorithm	Program

*) Closure properties of Rec & REL:

- complement of REL need not be REL, if at all REL complement is recursive, then that language L is recursive.
- If L = REL
  $L_1 - L_2 = L_1 \cap \bar{L}_2$ = need not be REL
  i.e. not closed under difference
- Rec ∪ {Reg, CFL, DCFL, Rec, CSL} = Rec.
- REL ∪ " = REL
- Rec. ∩ {Reg, CFL, DCFL, CSL, Rec, REL} = Rec. / REL
- REL ∩ " = REL

NOTE:) • Finite Subset ⇒ All languages closed
• subset ⇒ not closed for all
• Inverse Homo ⇒ All closed

#) Undecidability & Decidability :-

*) Definition:

- To solve problem if :-
  1) Algo available → Decidable / Totally decidable / Turing decidable / Rec. L
  2) No Algo available → REL / partially decidable / Undecidable
- Decidable → Rec. L
  Semi decidable → REL (but not Rec. L)
  Undecidable → Non-REL

*) Decidability properties overview :-

1) Emptiness (Accepts emty lang. or not)
2) Equality (m/c 1 = = m/c 2 ?)
3) Finiteness (m/c accepts finite lang?)
4) Membership (w ∈ L or not) (Valid → Halt final, Invalid → Halt non-final)
5) Totality (m/c accepting Σ^* ?)
6) Intersection Empty (m/c 1 ∩ m/c 2 == ϕ ?)
   Union empty ∪
   Difference empty −
7) Subset (m/c 1 ⊆ m/c 2 ?)
8) Ambiguity (for every ambigous regular grammar, an equivalent un-ambiguous grammar possible?)
9) Co-finiteness (complement of m/c accepting finite lang?)
10) Uniform Halting ($\forall x \in \Sigma^*$, TM halt or not halt)

*) Some Dc/Un - decidable properties:

- For Reg. L ⇒ Almost everything decidable
- For CFL ⇒ Totality, Intersection] undecidable
- For Rec. L ⇒ Totality, Intersection] undecidable

Halting problem, State entry, Empty String, Blank tape, uniform halting] Decidable

*) When to say Decidable, Semidecidable, undecidable :-

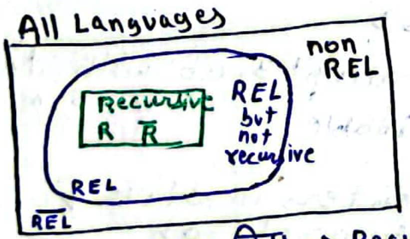
• 1) L is decidable if :-
- i) L has TM ($\forall w \in L \Rightarrow$ Halt Final)
- i.e. $\overline{L}$ has TM ($\forall w \notin L \Rightarrow$ Halt non final)
- ii) L has Halting TM (HTM)
- i.e. iii) L is REL && $\overline{L}$ also REL
- i.e. iv) L is Recursive

• 2) L is semidecidable if :-
- i) L has TM but $\overline{L}$ no TM
- i.e. ii) L is REL but $\overline{L}$ not REL
- i.e. iii) $\forall w \in L$ has logic [valid things logic possible]
- $\forall w \notin L$ has no logic [not for invalid]

• 3) L is undecidable if :-
- i) L has no TM
- i.e. ii) $\forall w \in L$ no logic

• Note :- i) Decidable → Recursive
- semi- " → REL (but not recursive)
- un- " → non-REL

•



• REL → REL मिला ⇒ Recursive है
- REL का complement किया → non REL मिला ⇒ $\overline{REL}$ है

• Mostly semidecidable is undecidable

• Examples (Decidable / Recursive) :

1) $a^n b^n \mid n < 10$
- Finite → Reg.
- L → TM ✓ , $\overline{L}$ → TM ✓
- ⇒ L → HTM ✓ ⇒ Decidable

2) $a^* b^*$
- Regular
- L → TM ✓ , $\overline{L}$ → TM ✓
- ⇒ L → HTM ✓ ⇒ Decidable

3) $a^n b^n \mid n \geq 1$
- DCFL ⇒ HTM ⇒ Decidable

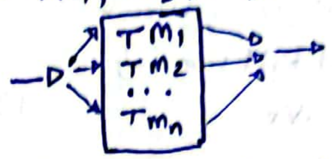
4) $w w^R \mid w \in \Sigma^*$
- CFL ⇒ HTM ⇒ Decidable

5) $w w \mid w \in \Sigma^*$
- CSL ⇒ HTM ⇒ Decidable

6) Check two regex equal or not
- CSL ✗ but HTM ✓ ⇒ Decidable

• Examples (Semidecidable / REL but not recursive) :

1) L = { All regular languages of $\Sigma = \{a,b\}$ }
- = { a^*, b^*, a^*b^*, b^*a^*, ... }
- = $TM_1, TM_2, TM_3, TM_4, \ldots$



⇒ L → TM ✓

- $\overline{L}$ = { All non-Regular languages }
- = maybe non-REL also
- ⇒ $\overline{L}$ → TM ✗
- Therefore, semidecidable

2) L = { All DCFL's }
- L → TM ✓
- $\overline{L}$ = { All non-DCFL's } = maybe non-REL also
- ⇒ $\overline{L}$ → TM ✗
- Therefore, semidecidable

3) i) All CFL's
- ii) All CSL's
- iii) All Rec. L
- iv) All REL

} Semidecidable

• Examples (Undecidable) :

1) L = { All non-regular languages }
- = maybe non-REL also ⇒ L → TM ✗
- Therefore, undecidable

2) All non-DCFL's
- non-CFL's
- non-CSL's
- non-rec.
- non-REL

} undecidable

• Mixed Examples :—

1) L = { All DFA's that will halt on w }

Valid (Yes)	Invalid (No)
⇓	⇓
w	Other than w
⇓	⇓
HF / HNF	HF / HNF
⇓	⇓
Logic ✓	Logic ✓

⇒ Decidable

2) DFA not halts on w
- DPDA halts on w
- DPDA not halts on w
- PDA "
- LBA "
- HTM

} Decidable

3) $L = \{TM \mid TM \text{ halts on } w\}$

Yes → Valid string → Halt on final → Logic ✓

No → Invalid string → HNF or ∞ loop → Logic ✗

⇒ Semidecidable ⇒ Undecidable

4) $L = \{TM \mid TM \text{ not halts on } w\}$

Yes → Invalid string → HNF or ∞ loop → Logic ✗

No

⇒ Undecidable

5) $L = \{DFA \mid DFA \text{ accepts } \epsilon\}$

Yes → ϵ → HF → Logic ✓

No → ϵ → HNF → Logic ✓

⇒ Decidable

6) DPDA accepts w
PDA " "
HTM " "
] Decidable [use cyk algo to verify]

7) HTM not accepts w

Yes → Invalid → HNF → Logic ✓

No → valid → HF → Logic ✓

⇒ Decidable

8) $L(DFA_1) = L(DFA_2)$
$L(DPDA_1) = L(DPDA_2)$
] Decidable [use equivalence algo. to verify]

9) TM accepts ϵ

Yes → valid → HF → Logic ✓

No → Invalid → HNF / ∞ loop → Logic ✗

⇒ Semidecidable

10) TM accepts something, i.e. $L(TM) \neq \phi$

Yes → Atleast 1 valid → HF → Logic ✓

No → All Invalid → HNF / ∞ loop → Logic ✗

⇒ Semidecidable
⇒ Undecidable

11) TM accepts only ab

Yes → ab accept && other strings reject → Logic ✓ && Logic ✗

No

⇒ Undecidable

12) TM accepts ab

Yes → HF

No → HNF / ∞ Loop

⇒ SD ⇒ UD

13) TM accepts only 2 strings
i.e. $|L(TM)| = 2$

⇒ UD

14) TM | no. of states in TM = 2
⇒ Decidable

15) TM | Tm accepts ab with 5 moves
⇒ Decidable

16) Tm | Tm accepts ab with atleast 5 moves

Yes → ab accepted after ≥ 5 moves → Logic ✓

No → No Logic ['.' infinite]

⇒ SD

17) TM | Tm accepts ab with atmost 5 moves
⇒ Decidable

18) TM | TM enters in state q to accept ab

Yes → valid → Logic ✓

No → Invalid → maybe ∞ loop → Logic ✗

⇒ SD

19) TM | TM enters into state 'q' within 5 moves to accept 'ab'
⇒ Decidable

Decidable / Undecidable chart

Problems	Reg FA	DCFL DPDA	CFL PDA	Rec. HTM	REL TM	CSL LBA
Halting Problem	⟷	Decidable	→	SD UD	D	
membership problem	⟷		→	SD UD	D	
Emptiness	⟷		→	NR UD	NR UD	✗
Finiteness	⟷		→	NR UD	NR UD	✗
Totality	⟷	→	NR UD	NR UD	NR UD	
Equality	⟷	→	NR UD	NR UD	NR UD	✗
Disjoint set (intersection)	⟷	NR UD	NR UD	NR UD	NR UD	✗
Subset	⟷	NR UD	NR UD	NR UD	NR UD	✗

		FA	DPDA	PDA	HTM	TM
	non-Halt i.e. $\overline{H}$	⟷	Decidable	→		NR UD
	$\overline{M}$	⟷		→		NR UD
	$\overline{E_m}$	⟷		→	SD UD	SD UD
First	$\overline{F}$	⟷		→	NR UD	NR UD
Tha	$\overline{T}$	⟷	→	SD UD	SD UD	NR UD
	$\overline{E_q}$	⟷	→	SD UD	SD UD	NR UD
	$\overline{D}$	⟷	SD UD	SD UD	SD UD	SD UD
	$\overline{S}$	⟷	SD UD	SD UD	SD UD	NR UD

$\cap$ is ϕ or not NR → non REL

Regularity	⟷	✗	✗	✗	✗
Ambiguity	⟷	✗	✗	✗	✗
Co-finiteness	⟷	✗	✗	✗	✗
completeness	⟷	✗	✗	✗	✗
$\overline{L}$ complement	⟷	✗	✗	✗	D

		REL
R	BTHP	✗
	SEP	✗
	PCP	✗
A	MCP	✗

Blank tape halt problem
State entry problem
Post correspondence problem
Modified PCP

coffee

Cum write definition of problems from p.no. 657

Regularity: whether given L is regular?
$\overline{L}$: If L is CFL (say) then $\overline{L}$ also CFL?

	Reg.	DCFL	CFL	CSL	Rec.	REL
1. Union	✓	✗	✓	✓	✓	✓
2. Concatenation	✓	✗	✓	✓	✓	✓
3. Kleene closure	✓	✗	✓	✓	✓	✓
4. Reversal	✓	✗	✓	✓	✓	✓
5. Intersection	✓	✗	✗	✓	✓	✓
6. Complement	✓	✓	✗	✓	✓	✗
7. Inverse-H	✓	✓	✓	✓	✓	✓
8. Homomorphism	✓	✗	✓	✓	✓	✓
9. Substitution	✓	✗	✓	✓	✓	✓
10. Subset	✗	✗	✗	✗	✗	✗
11. Infinite-U	✗	✗	✗	✗	✗	✗
12. Infinite-I	✗	✗	✗	✗	✗	✗
13. Infinite-D	✗	✗	✗	✗	✗	✗
14. Prefix	✓	✓	✓	✓	✓	✓
15. Quotient	✓	✗	✓	✓	✓	✓
16. Cycle	✓	✗	✓	✓	✓	✓
17. Min	✓	✗	✗			
18. Max	✓	✗	✗			
19. Half	✓	✗	✗			
20. L U Reg	✓	✓	✓	✓	✓	✓
21. L ∩ Reg	✓	✓	✓	✓	✓	✓
22. L - Reg	✓	✓	✓	✓	✓	✓
23. Reg - L	✓	✓	✗	✓	✓	✗
24. L / Reg.	✓	✓	✓	✓	✓	✓

(10-13) ✗ for all

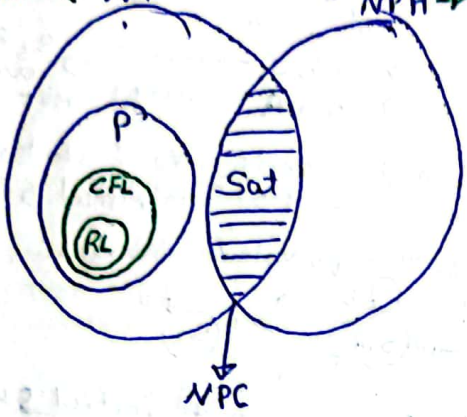
DCFL ✓	CFL ✗	REL ✗
	5	
6	6	6
7	17	
14	18	
(20-24)	19	
	23	23

For Suffix & Substring
DCFL → Not closed
CFL → closed

#) NP Complete & NP Hard:

- P → Deterministic algo. possible, taking polynomial time (like $n^{\log n}$)
- NP → Non-Deterministic algo. possible, taking non-polynomial time (like 2^n)
- If ($SAT \leq_P L$ AND writing non-Deterministic algo. for L is possible) then: L is NPC
  - i.e. L is NPH

- Decidable ⊂ NP
- NPH → Undecidable



- If L is in NP then L is decidable (RecL) & $\bar{L}$ also decidable (RecL)
  But $\bar{L}$ may/may not be in NP
- $P_1 \leq_P P_3$ i.e. problem P_1 is reducible to problem P_3 in polynomial time
  ⇒ P_3 is at least as hard as P_1, but P_3 can be even more hard also.

#) Pumping lemma for regular languages:

*) Point to remember:- ① It is necessary but not sufficient for L to be RL

① It is used to prove certain languages are non-regular by systematic procedure.
② It uses proof by contradiction.
③ It uses pigeonhole principle (loop)
④ non-RL → [PL] → non-RL
⑤ Pumping lemma used for infinite language.
⑥ RL → [PL] → may get contradiction / may not get contradiction

*) Procedure: Given L (an infinite lang.)
1) Assume L is RL
2) That means ∃ FA with (m) states (pumping constant)
3) Select one string $w \in L$ such that $|w| \geq m$ (i.e. loop is confirmed)
4)

$|uv| \leq m$; $|v| \geq 1$

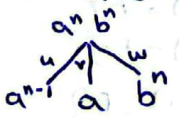
5) $uv^i w \in L$; $\forall i \geq 0$ then L is regular, o/w our assumption wrong, so L is non-RL

---

minimal string ⇒ 2, 10
confirmation of Loop ⇒ 5 onwards which is greater than 10 & belongs to L
⇒ 11 ∵ $a^2(a^3)^3$
So, PC = 11

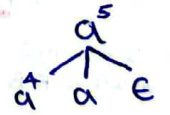
*) Questions -

1) Prove $L = \{a^n b^n \mid n \geq 1\}$; is non-RL
   i) Assume L is RL
   ii) Then, ∃ FA with m-states
   iii) Select a string $w \in L$; $|w| \geq m$
   $w = a^n b^n$ $|w| = m$ i.e. $2n = m$
   iv)



   $|u \cdot v| \leq m$
   $|v| \geq 1$
   v) $a^{n-i} a^i b^n \in L$; $\forall i \geq 0$
   at $i = 0$; $a^{n-1} b^n \notin L$ ⇒ L is non-RL

2) $L = \{a^p \mid p \text{ is Prime}\}$



$a^t a^i \in L$ $\forall i \geq 0$
at $i = 2$
$a^6 \notin L$ ⇒ L is nRL

3) $L = \{a^{2+3K} ; K \geq 0\}$ which of the following can be pumping length (the constant guaranteed by pumping lemma) for L?
A) 3 B) 5 C) 9 D) 24
$w = a^2 \cdot (a^3)^*$; यहाँ वाला Loop लेना सही $|v| \geq 1$ (See point ④)
⇒ Pumping constant = 5

4) $L = \{b^{10+12K} ; K \geq 0\}$ PC = ?
$w = b^{10} \cdot (b^{12})^*$ ⇒ PC = 22
Note:) 22 या 22 से बड़ा कुछ भी होगा

5) $L = \{a^{2+3K} \cup b^{10+12K} \mid K \geq 0\}$
$w_1 = a^2(a^3)^*$ ⇒ $PC_1 = 5$
$w_2 = b^{10}(b^{12})^*$ $PC_2 = 22$
now, see minimal strings
$s_1 = a^2$
$s_2 = b^{10}$

#) Myhill-Nerode theorem:

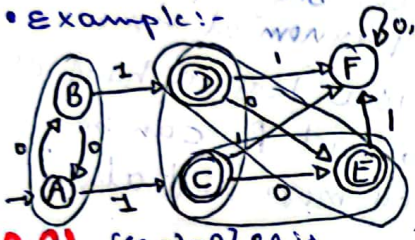
- It is used for minimization of DFA
- Steps:-

1) Draw a table for all pairs of states (P,Q)
2) Mark all pairs where P ∈ Final & Q ∉ Final
3) If there are any unmarked pairs (P,Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked, then mark [P,Q]; x = i/p symbol

Repeat this until no more markings can be made.

4) Combine all the unmarked pairs & make them a single state in minimized DFA.

- Example:-



	A	B	C	D	E	F
A	✗	✗	✗	✗	✗	✗
B		✗	✗	✗	✗	✗
C	✓	✓	✗	✗	✗	✗
D	✓	✓		✗	✗	✗
E	✓	✓			✗	✗
F	✓	✓	✓	✓	✓	✗

(B,A) - $\delta(B,0) = A$, $\delta(A,0) = B$ } BA is unmarked so no need to do anything
$\delta(B,1) = D$, $\delta(A,1) = C$ } unmarked so, leave

(D,C) - $\delta(D,0) = E$, $\delta(C,0) = E$; $\delta(D,1) = F$, $\delta(C,1) = F$

(E,C) - $\delta(E,0) = E$, $\delta(C,0) = E$; $\delta(E,1) = F$, $\delta(C,1) = F$

(E,D) - $\delta(E,0) = E$, $\delta(D,0) = E$; $\delta(E,1) = F$, $\delta(D,1) = F$

(F,A) - $\delta(F,0) = F$, $\delta(A,0) = B$; $\delta(F,1) = F$, $\delta(A,1) = C$ } FC is marked so mark FA also

(F,B) - $\delta(F,b) = F$, $\delta(B,b) = A$ } FA is marked so mark FB also

Repeat (u will get same)

Combined states: (AB), (DC), (E,C), (ED)

So, minimized DFA

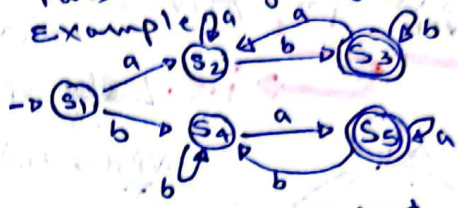
Note:) Pumping Constant = The minimal string which belongs to L from where loop will exist.

- Other way of saying Myhill-nerode theorem:-

To minimize dfa (ie. 2 finals equal >, or 2 non-finals equal >)

Parse strings xz & yz; ∀z ∈ Σ

Example:



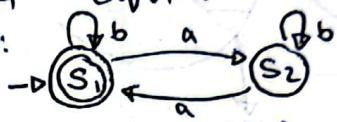
Starting ending symbol different

$S_3 \Rightarrow aba$ → reject
$S_5 \Rightarrow baa$ → accept
$\Rightarrow S_3$ & S_5 are not same

- No. of myhill-nerode Equivalence classes = No. of states in minimal DFA.

- DFA → Equivalence classes:

Given:

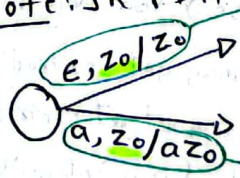


Equivalence class represented by S_1 ⇒ S_1 को initial State मान
⇒ Even no. of a's

Equivalence class represented by S_2 ⇒ S_2 को initial State मान
⇒ odd no. of a's

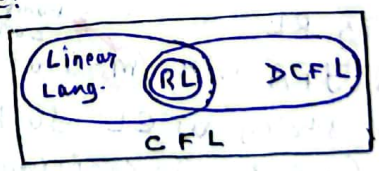
Note: Dead configuration: a situation where a state do not have transition for an i/p symbol.

Note: In PDA

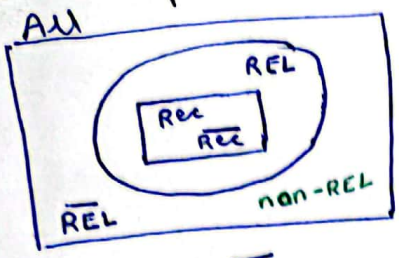


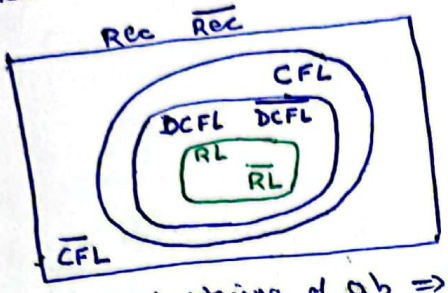
⇒ NPDA

Note:

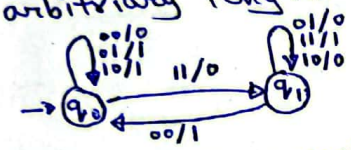


#) Some important notes:-





- Proper substring of $ab \Rightarrow \{\epsilon, a, b\}$
- FA to add two integers of any arbitrary length.



- For every NTM, an equivalent DTM is possible.
- Turing recognizable $\Rightarrow$ Rec L, REL
  Turing decidable $\Rightarrow$ Rec L
- Set of all REL (i.e. set of all TM's) is infinite but countable
- For any type of language L: $L \cup \overline{L} \Rightarrow \Sigma^*$
- $A \leq_p B$ means A is reducible to B
  i.e. A can't harder than B
- A is recursive if A & $\overline{A}$ both accepted by TM.
- A TM accepts a string 'w' within K steps is decidable
- Countable: A set is countable, means there exists an enumeration procedure to generate each of it's elements.
  And for a given element (inside this set) it takes finite steps to generate it using enumeration procedure.
  Ex: If $\Sigma = \{a, b\}$, then $\Sigma^* = \{\epsilon, a, b, aa, \cdots\}$
  thus Σ^* is countable infinite
  Ex: 2^{Σ^*} i.e. $P(\Sigma^*)$ is uncountable
  bec. power set of countably infinite set is uncountable.
- If L is an infinite language then subsets of L are uncountable.
- Every language is countable
- Σ^* is countable
- A subset of countable set is countable.

- Every ∞ lang. L has a subset S which is undecidable.
- Identifiers declared before use:-
  $\{wcw \mid w \in \Sigma^*\}$
- Balanced parenthesis:-
  $X \rightarrow XbX / XcX / dXf / g$