THEORY OF COMPUTATION

* 4 Modules
  - → FA & RE
  - → CFG & PDA
  - → RL, CFL, Pumping Lemma
  - → TM & Undecidability.

about 60 Qs.

1. FINITE AUTOMATA & REGULAR EXPRESSIONS

(ii) Alphabet (Σ): finite/non empty set of symbols

(iii) Power
- → Kleen closure (Σ^*)
- → Positive closure (Σ^+)

$\Sigma^+ = \Sigma^* - \varepsilon$

(iii) String: collection of alphabets

(iv) Length $= |w|$ $\quad \varepsilon \rightarrow |=0$

(v) Prefix.

$w = abc$ $\quad n+1$

$N_S = \{\varepsilon, a, ab, abc\}$

- If $L=n$:

imp

- No. of possible prefix/suffix $= n+1$
- proper prefix/suffix $= n$
- non-empty prefix/suffix $= n$
- non-empty proper $= n-1$

(vi) $w = abc$

make it improper

$SS = \{a, b, c, ab, bc, abc, \varepsilon\}$

trivial

- Substrings $= n(n+1)/2 + 1$
- Proper substrings $= n(n+1)/2$
- Non-empty substrings $= \frac{n(n+1)}{2}$
- Proper-Non empty $= \frac{n(n+1)}{2} - 1$

(vii) Reverse (w^R)

(viii) Palindrome: ($w = w^R$)

(ix) Union ($+, |, \cup$)

→ $\{a, b\}$

(X) Concat (·) → Not a group

a·b = ab

* Language : $L \subseteq \Sigma^*$ → Set of all strings poss over Σ [Countably infinite] (imp)

Finite (finite no of strings) | Infinite

* Finite Automata : self automatic
→ finite no of states / i/p Symbols

$M = (Q, \Sigma, q_0, \delta, F)$: 5 Tuple

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

→ DFA.

* FA
- → FA without output → NFA | DFA | ε-NFA
  [Lang. Acceptor / recognizer]
  [Final State matters]
- → FA with output → Moore | Mealy
  [output generator / transducer]

* i/p buffer

| a | a | b | a | b | a | $ |

cells | read head | REM

FCS

Finite tape

FA

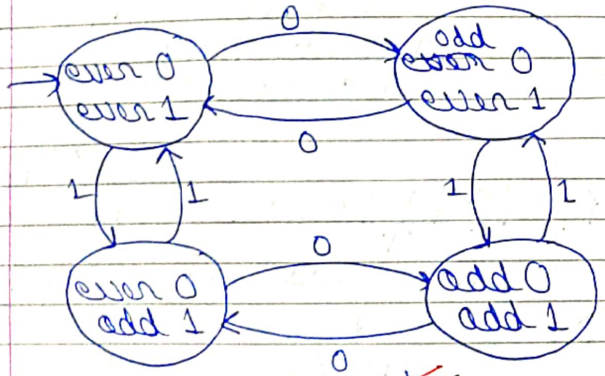
- Unidirectional
- Static memory → REM
- $: end of string

* DFA

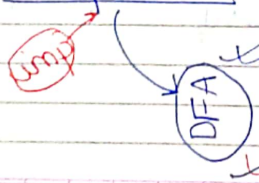
- From each state, path of i/p symbol has to be mentioned.

$$Q \times \Sigma \longrightarrow Q$$

- If only 1 string : n+2 States [VI] [dead State]



- Complement : interchange F with NF states.



- AP : FA can accept.

- $L = \{ w \mid w \bmod n = 0, w \in (a,b)^* \}$

→ n states

- divisible by 4. $\Sigma = \{0,1\}$

	0	1
→ * q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
* q_3	q_2	q_3

Trick

- Accept all binary strings that are divisible by m or divisible by n

→ m * n States

* NFA

- accept more than 1 path / no path on a symbol.

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

$\delta : Q \times \Sigma \longrightarrow 2^Q$

↳ Power Set.

eg: q_0, q_1

↓

$\{\varepsilon, \{q_0\}, \{q_1\}, \{q_0 q_1\}\} = 2^2 = 4$.

- Verify all possibilities : backtrack

↓

DFA > (efficient) NFA

- Every DFA is NFA, but every NFA is not DFA, but can be converted to its equivalent DFA. [Decidable]

[equivalent Power].

DFA $\overset{\text{power}}{\equiv}$ NFA

Vimp

* Conversion from NFA to DFA.

- decidable. (Subset Construction)

$M_{NFA} = (Q, \Sigma, q_0, \delta, F)$

↓

$M'_{DFA} = (Q', \Sigma, q_0, \delta', F')$

Q1

δ	a	b
$\rightarrow q_0$	$q_0 q_1$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

: NFA.

δ	a	b
$\rightarrow q_0$	$q_0 q_1$	q_0
$q_0 q_1$	$q_0 q_1$	$q_0 q_2$
$*q_0 q_2$	$q_0 q_1$	q_0

: DFA.

VI

- If NFA → n States

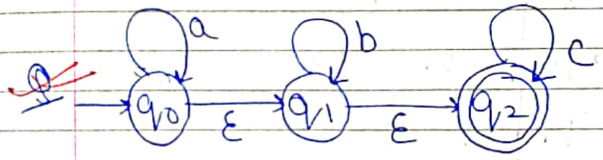
DFA : [1, 2^n] States

* ε-NFA ✓

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

$\delta : Q \times \Sigma \cup (\varepsilon) \longrightarrow 2^Q$

- Without reading i/p symbol: we can change states ✓



ε-NFA

	c	a	b	c
→	q_0	q_0	ϕ	ϕ
	q_1	ϕ	q_1	ϕ
*	q_2	ϕ	ϕ	q_2

- ε-NFA → NFA

- $\varepsilon^*(q_0) = \{q_0, q_1, q_2\}$
  $\varepsilon^*(q_1) = \{q_1, q_2\}$
  $\varepsilon^*(q_2) = \{q_2\}$

ε NFA → NFA

NFA

	a	b	c
q_0	$q_0q_1q_2$	q_1q_2	q_2
q_1	ϕ	q_1q_2	q_2
q_2	ϕ	ϕ	q_2

jump

- q_0, q_1, q_2 → all Final (reachable in ε*)

Pg-63

* $M_{\varepsilon\text{-NFA}} = (Q, \Sigma, q_0, \delta, F)$

jump

$M_{NFA} = (Q, \Sigma, q_0, \delta', F')$

VI

ε-NFA → DFA → decidable

$M = (Q, \Sigma \cup (\varepsilon), q_0, \delta, F)$
ε-NFA

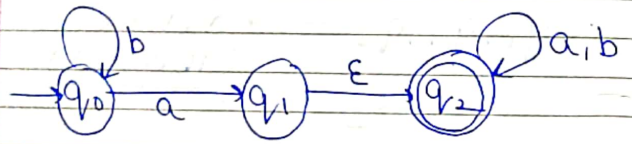
⇓

Imp

$M' = (Q', \Sigma, q_0', \delta', F')$
DFA

$q_0' = \varepsilon^*(q_0)$

Q



ε-NFA

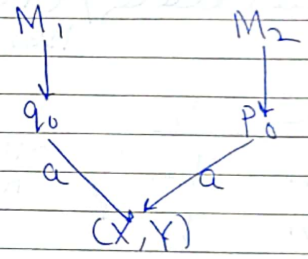
	a	b
q_0	q_1	q_0
q_1	ϕ	ϕ
q_2	q_2	q_2

$\varepsilon^*(q_0) = q_0$
$\varepsilon^*(q_1) = q_1, q_2$
$\varepsilon^*(q_2) = q_2$

δ'	a	b
→ q_0	$q_1 q_2$	q_0
* $q_1 q_2$	q_2	q_2
* q_2	q_2	q_2

* Equivalence of 2 DFAS :

→ Procedure.



X → F ; X → NF
Y → F ; Y → NF

: Continue.

Equival. of DFA : decidable
Equival. of PDA : Undecidable
Equival. of TM : Undecidable

Vimp

* Minimization of FA

- For every regular language → 1 minimal DFA
  → ∞ DFAs / NFAs.

- → π-equivalence / Partitioning
- → Table Filling / (Myhill-Nerode Algo)

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4*
* q_4	q_1	q_2

π-equivl method.

Disjoint Set

π_0 : $[q_4]$, $[q_0 q_1 q_2 q_3]$

π_1 : $[q_4]$ $[q_0 q_1 q_2]$ $[q_3]$

π_2 : $[q_4]$ $[q_0 q_2]$ $[q_1]$ $[q_3]$

↳ 4 states

till $\pi_k = \pi_{k+1}$

- Table Filling: look from Notes [Not needed]

* FA with output

6 tuples

- Final state not required

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

Δ → o/p alphabet
λ → o/p funct^n

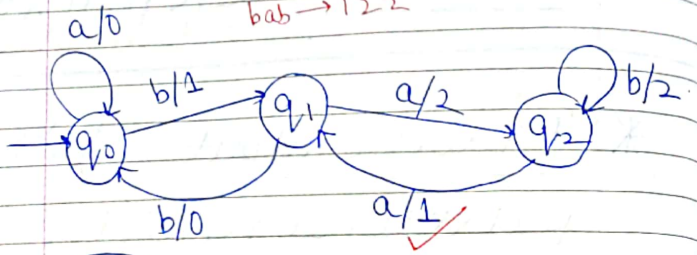
$\delta : Q \times \Sigma \to Q$ (DFA)

(1) Mealy m/c

$\delta : Q \times \Sigma \to Q$
$\lambda : Q \times \Sigma \to \Delta$

o/p depends both on current State & i/p Symbol.

- If $|w|=n$, the o/p string length = n. [Melay m/c]



Melay

PS	a	b
	NS, o/p	NS, o/p
q_0	q_0, 0	q_1, 1
q_1	q_2, 2	q_0, 0
q_2	q_1, 1	q_2, 2

imp

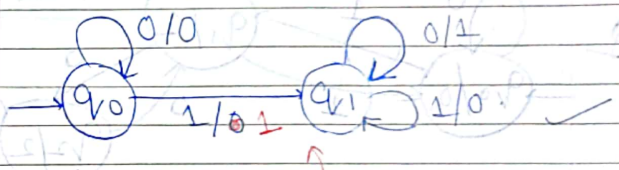
(2) Moore m/c

$$\delta : Q \times \Sigma \longrightarrow Q$$
$$\lambda : Q \longrightarrow \Delta$$

- If the i/p string length is n, the o/p string length is n+1

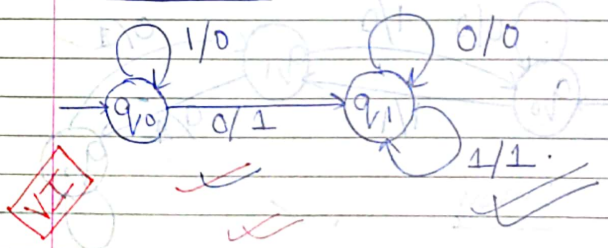
	a	b	o/p
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

- 2's complement



reads from LSB to MSB.

- increment.



VI

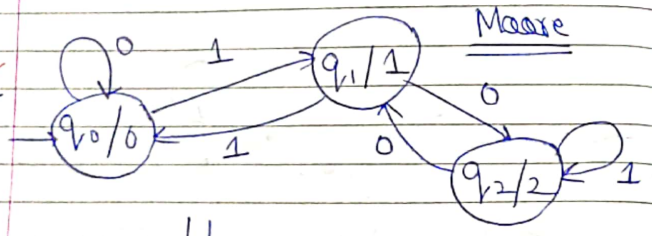
* Conversion from Moore to Melay m/c

$M = (Q, \Sigma, q_0, \delta, \Delta, \lambda)$ [Moore]

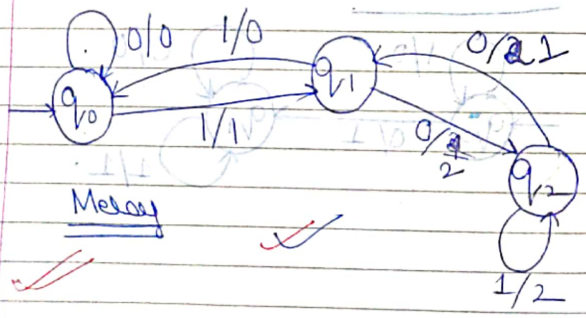
$\Downarrow$

$M' = (Q, \Sigma, q_0, \delta, \Delta, \lambda')$ [Melay]

Moore



$\Downarrow$



Melay

* Conversion from Melay to Moore m/c

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

$\Downarrow$ jump

$M' = (Q', \Sigma, [q_0 b], \delta', \Delta, \lambda')$

	a	b
q_0	q_0, a	q_1, b
q_1	q_1, b	q_2, a
q_2	q_1, a	q_0, a

Melay

$\Downarrow$

	a	b	o/p
$\rightarrow q_0$	q_0	$q_1 b$	a
$q_1 a$	$q_1 b$	q_2	a
$q_1 b$	$q_1 b$	q_2	b
q_2	$q_1 a$	q_0	a

Moore

* REGULAR EXPRESSIONS

- Operators + i/p alphabets
  - → *
  - → .
  - → +
- accepted by FA
- FS representation → lexical Analyser.
- describe language / compiler rules.

* IDENTITY RULES

ϕ:

$\phi^* = \varepsilon$

$\phi + r = r$

$\phi \cdot r = \phi$

ε:

$\varepsilon^* = \varepsilon$

$\varepsilon \cdot r = r \cdot \varepsilon = r$

$\varepsilon + r \cdot r^* = r^*$

$r \cdot r^* = r+$ → since

$\varepsilon + r^* = r^*$

$(\varepsilon + r)^* = (\varepsilon^* \cdot r^*)^* = r^*$.

r:

$r + r = r$

$(r^* \cdot r^*) = (r + r)^* = r^*$.

$r^* \cdot r^* = r^*$

$(r^*)^* = r^*$

p, q: (Yaing)

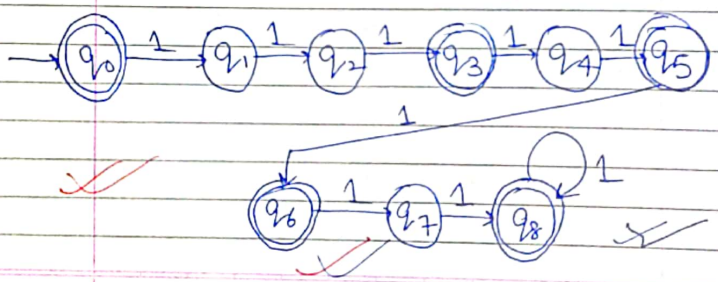
$$(p+q)^* = (p^* \cdot q^*)^* = (p^* + q^*)^*$$

- $\{\varepsilon, 1^3, 1^5, 1^6, 1^8, 1^9, 1^{10} \ldots 1^n\}$

min = 8 states.

$\therefore 8 + 1 \ (n+1) = 9$.

$\Sigma = \{1\}$



* Arden's Theorem

↳ FA ⟶ RE

→ $R = Q + RP$.

$R = QP^*$

→ $R = Q + PR + Q$. VI

$R = P^*Q$

2. CONTEXT FREE GRAMMAR & PUSH DOWN AUTOMATA

* CFG (Type-2)

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

$\alpha \longrightarrow \beta$

$\alpha = 1NT \longrightarrow V$.

$\beta = (V \cup T)^*$

$\alpha \longrightarrow \beta$

↓ INT ↓ $(VUT)^*$

* RG (Type-3)

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

$\alpha \longrightarrow \beta$.

imp $\alpha = 1NT \longrightarrow V$.

β = only 1 NT (either left/right) + T*

$\alpha \longrightarrow \beta$

↓ INT ↓ -1NT(L/R) -T*

eg: $S \longrightarrow aA \mid a$.

↳ Right Linear Grammar.

$S \rightarrow Aab$

↳ Left linear Grammar.

- GFG ⟶ CFL ⟶ PDA.

(T)

- Collection of terminal variables = Strings.

$L = \{a^n b^n \mid n > 0\}$

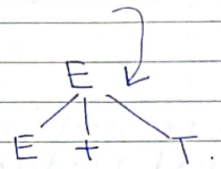
↳ Lang. Definition

T terminals : leaf nodes in Parse
NT Non-terminals : branch nodes

- Derivation : process to get String from S. (Production used)
  - → Left Most Derivation
  - → Right Most Derivation.

1) LMD: Select the left most NT first.

- Parse Tree :



bbaaaabb

- $S \rightarrow aSa \mid bSb \mid \epsilon$ (Stop Condn)

↳ even length palindrom

- Generate the language first from the Grammar : See the pattern.

Tip:

* CFG (Undecidable Approach)
  - → Ambiguous : More than 1 PT
  - → UnAmbiguous : only 1 PT

* Simplification of CFG :

- reduce Grammar Size.
- reduce no. of steps.

1. Eliminate ε productions
2. Eliminate unit productions
3. Eliminate useless productions
   - non-generating
   - non-reach.

1. $S \to ABC$
   $A \to aA \mid \epsilon$
   $B \to bB \mid \epsilon$
   $C \to c$

(imp)

$A \to \epsilon$
$B \to \epsilon$
ε-prodn

$\Rightarrow S \to ABC \mid BC \mid AC \mid C$
$A \to aA \mid a$
$B \to bB \mid b$
$C \to c$.

2. $S \to ABC$
   $A \to aB \mid B$
   $B \to dA \mid C$
   $C \to b$

Unit Productns.

$\Rightarrow S \to ABC$
$A \to aB \mid aA \mid b$
$B \to aA \mid b$
$C \to b$.

3. $S \to AB\bar{C}$
   $A \to a\bar{A} \mid \bar{a}$
   $B \to bB \mid b\bar{D} \mid \bar{b}$
   $C \to c\bar{D} \mid \bar{c}$
   $D \to dD$
   $E \to e$

→ non-generating

→ non-reachable.

Soln:

eliminate useless Symbols.

$S \to ABC$
$A \to aA \mid a$
$B \to bB \mid b$
$C \to c$
$D \to dD$.

* Normal Forms

- If L = n, how many steps required to derive the String?

CNF GNF.

1) CHOMSKY Normal Form (binary tree)

imp → 1NT ⟶ NT · NT · | T .

- L = n
  (2n ± 1) steps to derive String.

2) CGF ⟶ CNF (Simplified Grammar)

S ⟶ ABC	S ⟶ ABC
A ⟶ aA | B	A ⟶ aA | bB | b
B ⟶ bB | b	B ⟶ bB | b
C ⟶ c.	C ⟶ c

Soln. S ⟶ XC $X_a \to a$; $X_b \to b$.
X ⟶ AB
A ⟶ X_A A | X_b B | b
B ⟶ X_b B | b
C ⟶ c
$X_c \to c$

2) Greibach Normal Form (GNF)

imp → 1NT ⟶ T(NT)*

- L = n
  n steps required to derive String

- $A_i \longrightarrow A_j \cdot \alpha$.

ij: i < j : No change
i > j : Substitution
i = j : LR method

A ⟶ Aa = LR

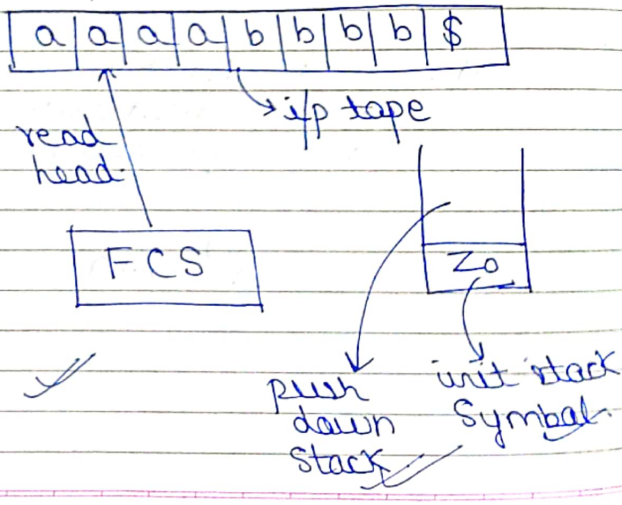
$A \rightarrow A\alpha_1 | A\alpha_2 | \cdots | \beta_1 | \beta_2 \cdots$

$A \rightarrow \beta_1 A' | \beta_2 A' | \cdots | \beta_1 | \beta_2 |$

$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \cdots | \alpha_1 | \alpha_2 |$

* PDA

- FA + Stack.
- Limitations of FA covered.



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

Z_0 : init stack Symbal; Γ : Stack Alphabet

$Z_0 \in \Gamma$ — VI

$\delta : Q \times \Sigma \times \Gamma \rightarrow Q \cdot \Gamma^*$ (jump) → DPDA

$\delta : Q \times \Sigma \times \Gamma \rightarrow 2^{Q \cdot \Gamma^*}$ → NPDA.

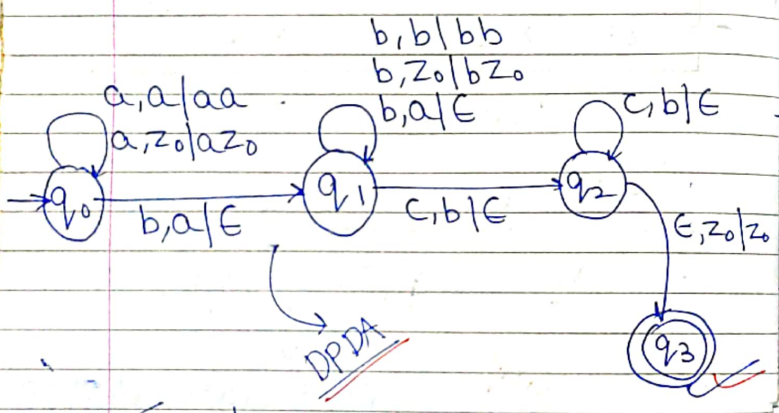
q^* : Stack Operation

- Push (q, State, i/p, TOS) ⇓ (next State, i/p · TOS) (jump)
- Pop (State, i/p, TOS) ⇓ (next State, ϵ)

- No change : (State, i/p, TOS)
  ⇓
  (next State, TOS)

- PDA
  - final state acceptance
  - empty stack acceptance

eg. $L=\{a^m b^{m+n} c^n \mid m,n>0\}$ → min string: abbc



- a, z0 | az0 : agar a dikhra a, z0 ke upar, toh push kar
- c, b | ϵ : agar c dikhra a b ke upar, toh nikal b ka.

- ww^Cw^R : DPDA.
- ww^R : NPDA.

NPDA > (power) DPDA

- DPDA & NPDA are not interconvertible.
  - accepts more no. of languages

- PDA used for constructing syntax analyser in compiler.

- Every finite lang. is regular.
- $L=\{a^n \mid n \text{ is prime}\}$ → [no AP observed]
  - not regular (CSL)
  - $\forall \in (a,b)^*$
- ww^R : regular. on $\Sigma=\{a,b\}$
  - $L = a(a+b)^*a + b(a+b)^*b$

imp

- All RLs are CFLs.

* Pumping Lemma

$L = \{a^n b^n \mid n > 0\}$

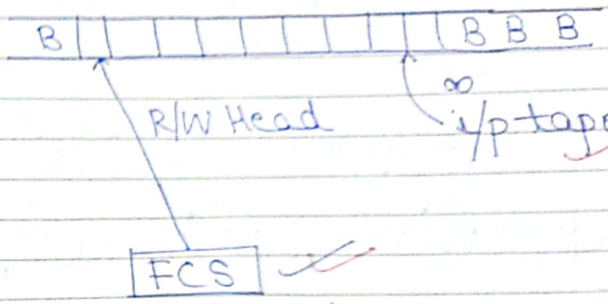
1. $|z| \geq n$
2. $|uv| \leq n$
3. $|v| \geq 1$

$z = uv^i w$ $i \in 0$ to k

* Drawbacks of PDA:

- unidirectional
- only read operation is allowed.

4) TURING MACHINE



- bidirectional head movement.

→ $M = (Q, \Sigma, q_0, \delta, F, \Delta, T)$ 7 tuples

→ $\delta: Q \times T \longrightarrow Q \times T \times \{L, R\}$

TM. (jump)

T → Tape alphabet (Σ + o/p + Δ).

Δ → Blank Symbols.

F → Hault State.

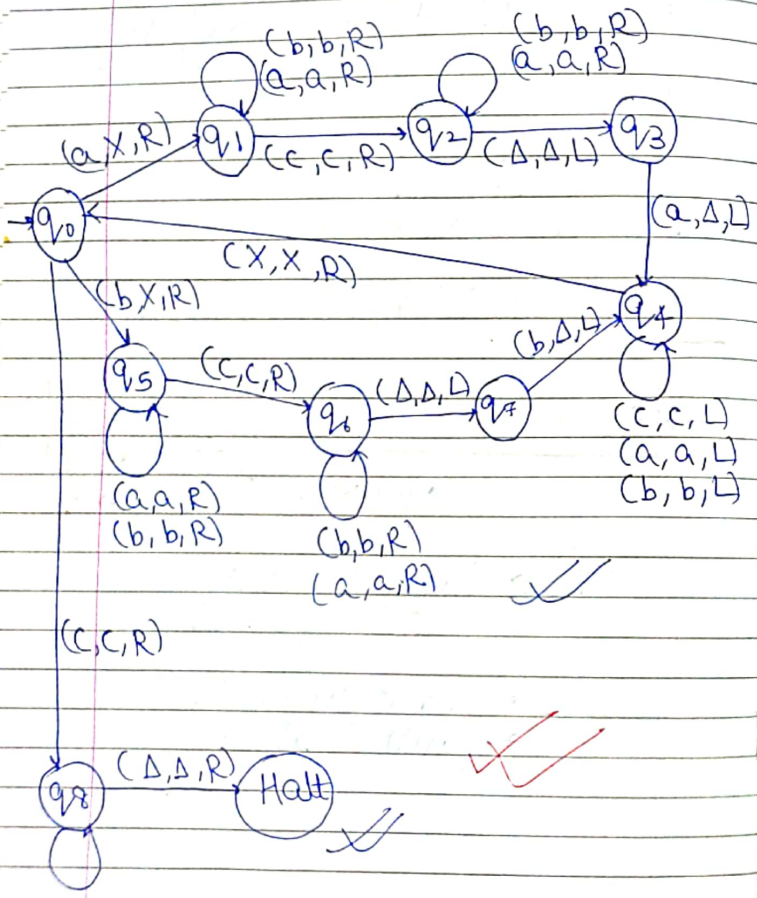
CFL

Q. TM for $wcw^R \mid w \in (a,b)^*$.

abbacabba

XXXX ΔΔΔΔ

Vimp



TM recognizable & decidable.

Jump

* Recursive & Recursively Enumerable Language:

Recursive: accepted/rejected by TM.

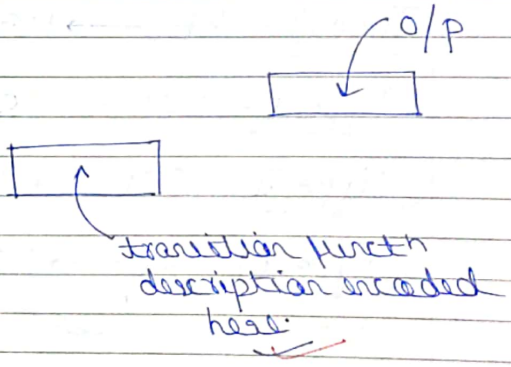
- Halting problem of TM.
  - undecidable.

[goes into ∞ loop]; waste lot of CPU cycles

- Universal TM:

FCS

M#w



$Q \times \Sigma \times T \longrightarrow 2^{Q \times T \times \{L,R\}}$

NTM.

Chomsky Hierarchy

Type 0 : REL ⟶ UG ⟶ TM [Not recursive]

Type 1 : CSL ⟶ CSG ⟶ LBA

Type 2 : CFL ⟶ CFG ⟶ PDA

Type 3 : RL ⟶ RG ⟶ FA.

Context Sensitive Grammar

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

$\alpha \longrightarrow \beta$

$(VUT)^* \quad (VUT)^*$

$|\beta| \geq |\alpha|$

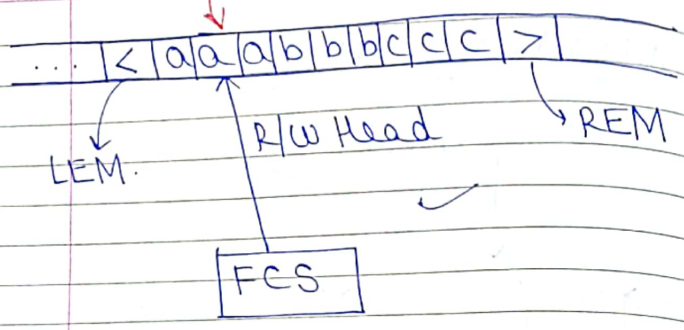
→ ε-productions not all (VI)

→ All CFL / RL are CSLs.

[Except ε productions, all productions in CFG are CSG].

LBA

- Tape length is finite.
- Static memory (need to specify memory before). (imp)



* $M = (Q, \Sigma, q_0, \delta, F, <, >, T)$

8 tuple

$\delta : Q \times T \longrightarrow Q \times T \times \{L R\}$

But, $> : R X$

$< : L X$

* UNDECIDABILITY

(Context free Grammar)

- Grammar is ambig./not : UD.
- eq. of 2 FAs/ RLs : Decidable
- equiv. of 2 CFGs/ PDA : UD

- Halting Problem (TM) : UD.
- Post correspondence Problem : UD.

* CLOSURE PROPERTIES

1) RL

closed under :

(i) Union
(ii) Concatenation
(iii) Intersection
(iv) Complement
(v) Kleen closure
(vi) Reverse
(vii) ϵ

not closed under :

(i) Subset operation.
(ii) Union all.

- Finiteness / Emptiness / Membership problem of RL:
  - Decidable ✓

- Finiteness / emptiness / Membership problem of CFL:
  - Decidable
  - (UNDECIDABLE)
  - (FEEM)

* CFL are not closed under:
  - intersection
  - complement ✓

—x—x—x—x—

(VANI NOTES).

PREVIOUS YEAR GATE (LEARN)

- RL are not closed under infinite union.

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

→ Σ^* = {Set of all strings}
  - {ε, a, b, aa, ab, ba, bb ...

∴ there is a method of enumeration.

[Σ^* is countably infinite]

→ 2^{Σ^*} : Set of all languages
  - [uncountable] by diagonalisation

→ Set of all TMs (language accepted by TM) is countable (RELs)

∴ Set of all RLs / CFLs / CSLs = countable

- C / Java → Context Sensitive.

Gate [imp]

- effectively enumerated in lexicographic order.
  - recursive but not context free.

- A TM prints a specific letter: [state entry problem]
  - undecidable.
  - whether it goes to the halt state for which has this letter as the output.

- A FSM can add 2 integers of any arbitrary length.
  - memoryless

Given $\Sigma = (a, b)$

- Set of all strings over Σ: countably infinite (Σ^*)
- Set of all languages: 2^{Σ^*}: uncountable
- Set of all regular lang: countably infinite.

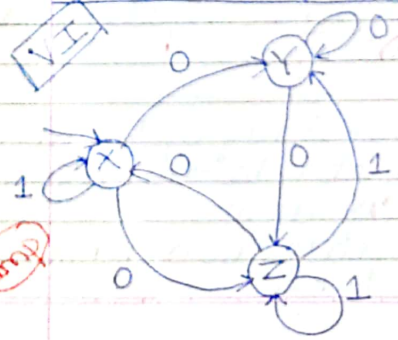
- Set of all lang accepted by TM = Set of all RE lang = countably infinite

$\Rightarrow 01^*0$ (Regular?)
- Infinite Set of finite string
- String: finite length (def)

Pumping lemma

If a language L is regular, it must satisfy the pumping lemma for regular lang.
- Converse not True

∴ Both Regular & Non-regular lang can satisfy pumping lemma for regular language



LLG

$Y = X0 + Z1 + Y0$

$Z = X0 + Z1 + Y0$

$X = \in + Z0 + X1$

$\therefore Y = Z$

→ $(a+b)^*$

- Σ^* is the regular superset of every language.
- ϕ is a regular subset of every lang.

- $a^n b^n$ → subset of every RL.
  - not regular. (CFL)
- Every subset of a finite lang is regular. (∵ finite) VI

- Regular languages not closed under infinite union.

- $a^n b^n \mid n>0$ = CFL (Not regular)
- $a^n b^m \mid m \neq n$ = CFL (Not regular)
- Union = Regular. $(a+b)^*$
- The union of 2 non-regular sets can be regular.

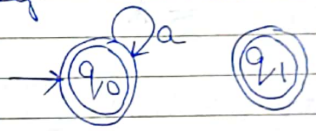
- Let w be any string of length n in $(0,1)^*$. Let L be the set of all substrings of w.
  Min. no of states in a NFA = $n+1$.

(imp)

VI

- Look at the FA, derive the lang. & then check the options.

- If all states of an NFA are accepting states then the lang accepted by NFA is not Σ^*



(no transition).

- In case of DFA:

  $L(\overline{M}) = \overline{L(M)}$ VI

  L(M) (exchanging final & Non-finals)

- But in NFA:
  - no connection.

11

- P → RL (a^*b^*)
  Q — CF ($a^nb^n | n>0$)

$P \cap \bar{Q} = P \cap Q$

- $P \cap Q = a^nb^n$ ∴ CF — VI
- $P - Q$ = CF
- $\Sigma^* - P$ = R (closed under comp.)

* RE = $\Sigma^* 0011 \Sigma^*$
  - Substring = 0011
  - Now construct the DFA.

- Pascal, Fortran → Context Sensitive languages
  - Variable declared before use (imp)
  - Matching formal & actual parameters of functions.

(Syntax analyser)
- CFG
  - Syntax of recursive procedures
  - Variable names of arbitrary length (Lexical analyser)
  - Syntax of if-else statements

→ CFL is closed under: L_1=CF, L_2=R.
- regular difference ($L_1 - L_2$) — VI
- regular intersection

- $a^lb^mc^n | l \neq m, m \neq n$ (imp)
  - VI
  - accepted by NPDA not DPDA.

* { ww^R → NDCFL
  $w\#w^R$ → DCFL
  ww → CSL } — VI

- $(CFL)^c = (CSL)^c = CSL$
- $(RL)^c = RL$

→ A CSL may/may not be regular.

→ Every NFA —convert→ equiv. DFA

→ Every NDTM ——→ equiv. DTM

imp

VI

* Grammar may change but language remain the same in the gramm. & its disambiguated version.

RE ——→ Lexical Analysis
PDA ——→ Syntax Analysis
DataFlow Analysis ——→ Code Optimi.
Register Allocat^n ——→ Code gener.

Video

imp

→ Compiler design

4. $L_1 = CFL$
- $L_1 - RL = L_1 \cap \overline{RL} = L_1 \cap R = CFL$

∵ closed under regular intersection.

- $a^m b^n c^p d^q \mid m+p = n+q$, where $m, n, p, q \geq 0$

imp

$\Rightarrow m - n + p - q = 0$

∴ CFL (take an eg (Stack))

[3 comparisons]

- $a^m b^n c^p d^q \mid m = n = p, p \neq q$

↳ CSL

imp

* The derivation trees of strings generated by a CFG in CNF are always binary trees.

[Strict]

eg: $S \to AB/a$ (CNF)

→ CSL

- Checking that ident. are declared b4 use → $WcW \mid w \in (a,b)^*$

  ↳ TM

- Membership problem in CFL is decidable.

- Regularity of CFL is undecidable (imp) (VI)

- Membership problem for RE is undecidable.

  ↳ Halting Problem.

State entry Problem in TM is undecidable

- Recursive Lang. are proper superset of CFL.

  ∵ All CFL → RL

  but, some RL not CFL.

  ↳ recognizable & decidable by TM.

- Recursively Enumerable Lang. & CFL are not closed under complementation (VI)

(VI)

NFA ⟷ DFA (recognition power)

NTM ⟷ DTM

DPDA ⟷ NPDA

Equivalence of 2 PDA: undecidable
Equivalence of 2 TMs: undecidable
Equivalence of 2 FAs: decidable.

→ Emptiness, Membership problem, finiteness of CFG is decidable.

* $a^p \mid p$ is a prime

  ↳ CSL; accepted by LBA/TM.

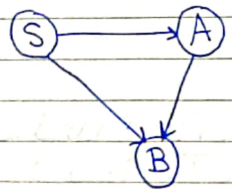
VIMP

imp

* The Set of all recursively enumerable languages is:
  (i) closed under intersection.
  (ii) countable Set
  (iii) not closed under complement

(LLG)

* S → Ab | Bb | c [Spotting Left recursion]
  A → Bd | ε
  B → e

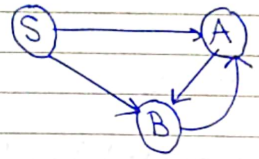


(No cycle)
[No Left Recursion]

VI

S → Aa | Bb | c
A → Bd | ε
B → Ac | ε

can be figured without diag



(Left Recursion)

- No. of trivial substring for any string is 2.

- Minimal DFA equivalent to a NDFA has sometimes more states

R.(Φ, 2^n)

Language is regular if: VI
→ LG & RLG exists
→ NFA with single final state exists
→ NFA without ε-move exists.

VI:
M = (Q, Σ, δ, S, F)
M' = (Q, Σ, δ, S, Q−F)

i) M → NFA
(No relation b/w M and M')

CSLs → decidable. [Complement, Intersect, Membership].

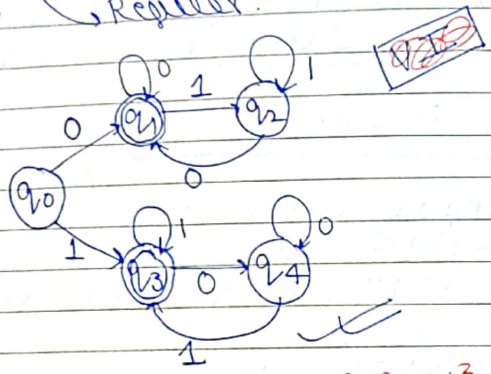
imp

but, eg 00, 11 → CF

[difference = IC (finite)]

* L_2 = set of all strings having eq. no. of 01 and 10

→ Regular.



{ab, a²b², a³b³, ...}

$L_1 = \{a^m b^n, m+n = \text{even}\}$ (jump)

→ $\{(aa)^*(bb)^* + a(aa)^* b(bb)^*\}$

∴ RL

$L(M)$ = Lang. accepted by M

$L(\overline{M})$ = Lang. accepted by comp of M

$\overline{L(M)}$ = Comm of Lang accepted by M.

DFA: $L(\overline{M}) = \overline{L(M)}$

NFA: $L(\overline{M}) \neq \overline{L(M)}$

$L \subset \Sigma^*$ → cofinite

⊙ → if their compliment is finite.

L_1 → regular (jump) $L_1 = (a+b)^*$, $L_2 = a^n b^n$

$L_1 \cup L_2$ → regular

L_2 → may not be regular.

[closed under reverse]

If L is regular;
LR is regular.

* Infinite intersection → Not regular.

$wx \mid |w| = |x|$ (VI) (Half)

$L = \{0110, 01, 1010, \ldots\}$

(jump) → $(01+10+11+00)^*$

∴ Regular.

Recursive Language.

↳ Turing Decidable

T of a Standard TM. is superset of Σ

$$T = \Sigma + B + Q + P$$

$L = L_1 \cup L_2$ — VI

L_1 regular

If L_1: regular & finite.

L_2: regular.

Σ = m symbols

n States — VI

Total types = 2^n.

⇒ No. of DFA / type = n^{m*n}.

$$= \text{Total DFAs} = n^{m*n} \times 2^n$$

Q: How many 2 State DFAs can be constructed over $\Sigma = \{a, b\}$ with designated init state. $m=2$, $n=2$

Types — 2^4

1. → (X) (Y)
2. → (X) Y
3. → X (Y)
4. → (X) (Y)

4 Types.

$n^{m \times n}$

$2^{4} \times 2^{2}$

16×4

δ	a	b
x	2	2 → 2 pairs.
y	2	2

x(b) → x, y } 2.

16 DFAs/type

$n^{m \times n} \times 2^n$

$2^4 \times 4$

$= 16 \times 4 = 64$

∴ $16 \times 4 = 64$

m = No. of Symbols in Σ
n = No. of States
Total no. of Types = 2^n
No. of DFA's in each Type = n^{m*n}

VI

Total DFAs = $2^n * n^{mn}$

· 1 eq → Solution (PS-5)

Q/4 , Q/5

* If L: finite language VI

max. length of string = n
min. length of string = m

imp

Min no of States in DFA = n+2.

· Every RL can be accepted by NFA with only 1 final State.

→ ε moves.

imp

→ Whether a given CFL is regular; is decidable. [Undecidable]

RE language

VI

→ closed : union, intersection

→ not closed : complementation.

If L_1 = CFL

$\overline{L_1} = \overline{CFL} = \overline{CSL} = CSL$.

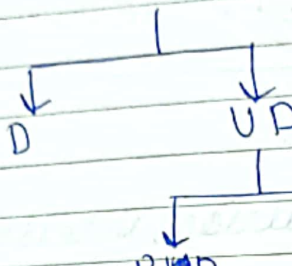
∵ CSL closed under complementation.

→ Deciding if a given string is generated by a given context free Grammer.

↳ Decidable.

→ Membership Problem,
Regularity problem,
equivalence problem of REL is
Undecidable

REL → Partially ~~undecidable~~ decidable.



* If Strings of language L can be effectively enumerated in lexicographic order

→ Recursive lang. defⁿ.

REL & CFL are not closed under complementation.

If A → UD
& reduced to B
AB = UD

If B → D
& A reduced to B
A = D