Express yourself Shot on HOT S 3

THEORY OF COMPUTATION: (SHORT NOTES)

Problems → Not Computable / Not REL (Infinite Languages)
↳ Computable / REL (Infinite Languages) → TM (NTM ≡ DTM) (Infinite Memory & Process)
↳ REC (") → HTM
↳ CSL (") → LBA (bounded Memory)
↳ CFL (") → PDA ≠ 1 Stack
↳ DCFL (") → DPDA
↳ RL (") → DFA & NFA 1 Byte
↳ Finite Languages

Deterministic:
→ No ε
→ every symbol should have a transition.
→ includes dead state.

∅ → Empty Language.
ε → String of zero length.

(*) Regular Grammar generates Regular Languages, that is accepted by Finite Automata.

(*) Regular Expression helps in representation of Regular Languages.
↳ Uses (only) 4 operators: +(OR), •(concat), *(K.C), +(P.C).

Q. Understand the following Regular Expressions:

1) ε+ε
2) ε+∅
3) ∅+∅
4) ε.ε
5) ε.∅
6) ∅.ε
7) ∅*
8) ∅+
9) ε*
10) ε+
11) a+∅
12) a+ε
13) a.ε
14) a.∅
15) a.a
16) a.a*
17) a*.a
18) a+a*
19) ε+a*
20) ε+a+
21) a+a+
22) a*+a*
23) a*+a+
24) a*.a*
25) a*.a+
26) a*+(aa)*
27) (a+ε)*
28) (a+ε)+
29) (a*)*
30) (a*)+
31) (a+)*
32) (a+)+
33) a+.a*
34) a*a+
35) a+.a+
36) (a*b*)*

Answer:
1) ε 11) a 21) a+ 31) a*
2) ε 12) ε+a 22) a* 32) a+
3) ∅ 13) a 23) a* 33) a+
4) ε 14) ∅ 24) a* 34) a+
5) ∅ 15) a² 25) a+ 35) aa+/a+a≠a+
6) ∅ 16) a+ 26) a* 36) (b*a*)*
7) ε 17) a+ 27) a* =(b*a*)+
8) ∅ 18) a* 28) a* ≠(b+a+)*
9) ε 19) a* 29) a*
10) ε 20) a* 30) a*

Q. Write Regular Expressions for the following

1) L = {x | x ∈ (a,b)*}
2) L = {x | x ∈ (a,b)*, |x| ≠ 0}
3) L = { }
4) L = {ε}
5) L = {xw | x = a, w ∈ (a,b)*}.

FINITE AUTOMATA

→ without Output → NFA ≅ DFA ≅ Min DFA.

→ with Output

#States in ε-NFA = #States in NFA

$\{Q, \Sigma, \delta, q_0, F\}$.

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

NFA: $\delta : Q \times \Sigma \to 2^Q$.

Moore M/c

O/p with states.

$\lambda : Q \to \Delta$

Mealy M/c

O/p with transitions.

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

(*) For an 'n' length input, the output length of

Moore Machine : n+1

1st char of output is ignored

Mealy Machine : n

$\{Q, \Sigma, \delta, q_0, \Delta, \lambda\}$

Δ → output alphabet

λ → output function.

• No final state.

(*) If every state is non-final, both DFA & NFA accepts $\emptyset$, but when every state is final, DFA accepts Σ^* but NFA may not.

(*) Every Non Regular language has infinite Equivalence classes. But for Regular languages,

*** #Equivalence Classes = #States in min. DFA.

(*) If a given NFA has Q states, then the equivalent min. DFA has $\leq 2^Q$ states.

NOTES:

(*) If $x\underbrace{a \ldots}_{n^{th} \text{ last symbol}}$ = 2^n states. [For Q. 35, 36]

(1) Starts with contain dead state, but ends with or contains some string does not have a dead state, in the DFA.

(2) L and $\bar{L}$, for RL, has same no. of states in DFA (Exchange F&NFs)

(3) mod 'n' questions ⟶ n states in DFA.

(4) $n_a(w) = K$ or $\leq K$ ⟶ $(K+2)$ states

$n_a(w) \geq K$ ⟶ $(K+1)$ states.

(5) a^{xn+y} ⟶ $y+1$ states.

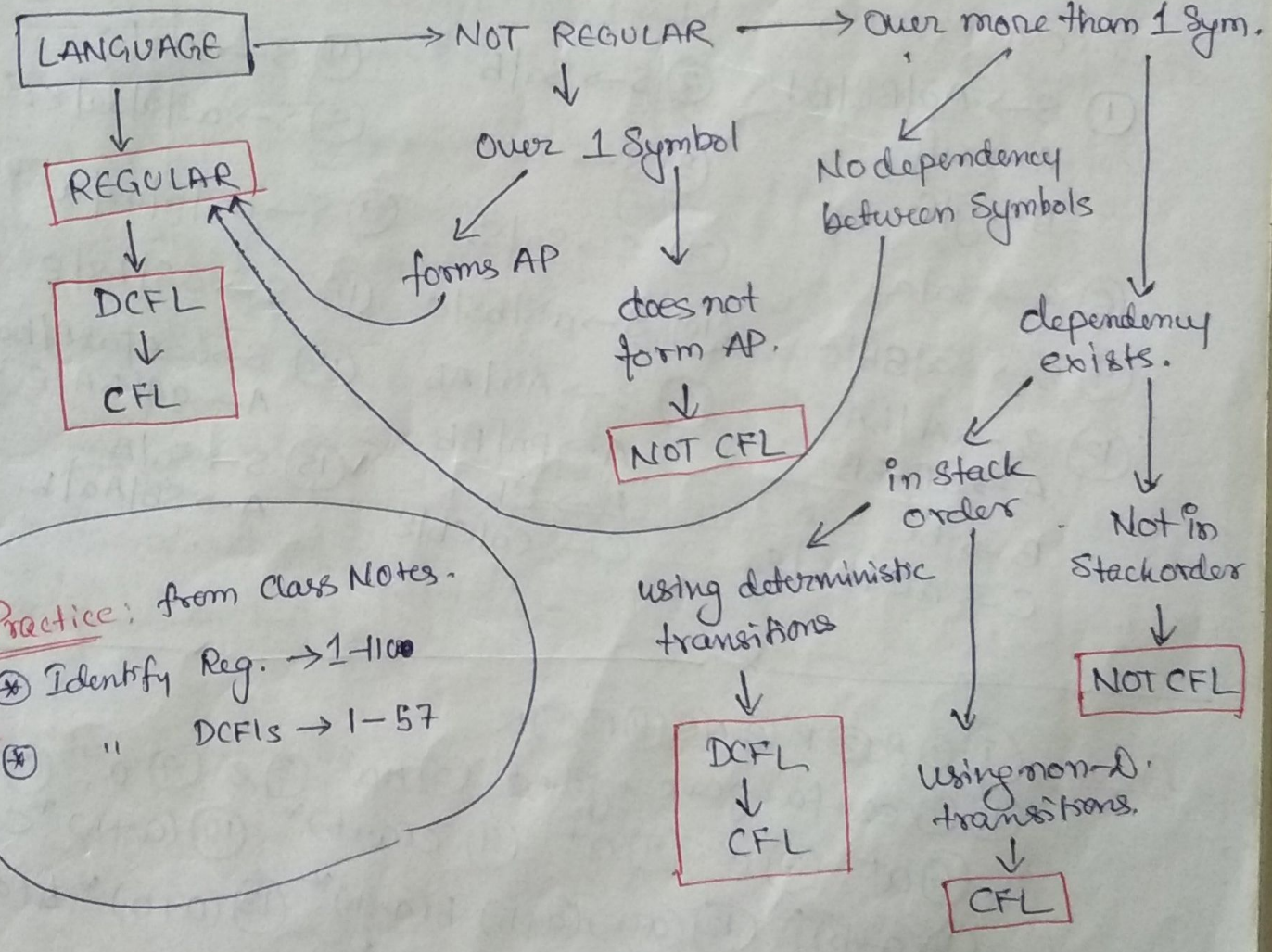
(6) $\{xa^{n}wx^{2}y^{2}z^{2} \mid x \in (a,b)^*$ and $y,z,w,x \in (a,b)\}$ ⟶ 2^5 states.

(7) n_a is divisible by x, n_b is divisible by y ⟶ $x*y$ states.

↳ and, or, but not, either or but not both.

where x even, y odd.

How to Identify REGULAR Languages, DCFLs & CFLs:



Practice: from Class Notes.

* (*) Identify Reg. → 1-100

** (*) " DCFLs → 1-57

GRAMMAR: It generates a language using set of rules.

$G = \{V, T, P, S\}$

(1) REGULAR GRAMMAR

- Left Linear: $V \to VT^* | T^*$
- Right Linear: $V \to T^*V | T^*$

LMD = # RMD = # PT

(2) CONTEXT FREE GRAMMAR → $V \to (V+T)^*$

(3) " SENSITIVE " → $\alpha \to \beta$, $|\alpha| \leq |\beta|$

(4) Unrestricted Grammar → $\alpha \to \beta$

(*) AMBIGUOUS CFGs: ∃string having more than 1 Parse Tree.

(*) UNAMBIGUOUS CFGs: ∀String having exactly 1 parse tree.

(*) Checking a CFG is AMBIGUOUS — SEMIDECIDABLE.

. . . . . UNAMBIGUOUS — NOT " .

Q. which of the following is Not Ambiguous?

(1) $S \to SS|a$ (2) $S \to Sa|bS|c$ (3) $S \to SaS|b$ (d) $S \to a|b$.

NOTE: If same NT participates in left as well as Right Recursion in a production, then it is surely Ambiguous. but vice versa is not true.

e.g → $S \to AB$
$A \to aA|\varepsilon|Ab$
$B \to Bb|\varepsilon$
— Ambiguous —
$S \to AB$
$A \to aA|\varepsilon$
$B \to aB|\varepsilon$.

PUSH DOWN AUTOMATA: FA + 1 Stack = PDA

$PDA = (Q, \Sigma, \delta, q_0, F, Z_0, \tau)$

τ → top of stack symbol

Z_0 → Bottom element of stack.

$\delta(DPDA) \Rightarrow Q \times \Sigma \times \tau \longrightarrow Q \times \tau^*$

$\neq$

$\delta(PDA) \Rightarrow Q \times \Sigma \cup \{\varepsilon\} \times \tau^* \longrightarrow 2^{Q \times \tau^*}$

τ^* → operation to be performed on TOS.

(*) Any CFL can be accepted using PDA with only 1 state.

STACK OPERATIONS:

① PUSH —
$\delta(q_i, x, y) \longrightarrow (q_j, xy)$ — DPDA
$\delta(q_i, x, \varepsilon) \longrightarrow (q_j, x)$
$\delta(q_i, x, X) \longrightarrow (q_j, xX)$
$\delta(q_i, x, yy) \longrightarrow (q_j, xyy)$ — PDA

② POP —
$\delta(q_i, x, y) = (q_j, \varepsilon)$ — DPDA
$\delta(q_i, x, yX) = (q_j, X)$
$\delta(q_i, x, yy) = (q_j, y)$ — PDA

③ NO OPERATION —
$\delta(q_i, x, y) = (q_j, y)$ — DPDA
$\delta(q, x, \varepsilon) = (q, \varepsilon)$
$\delta(q, x, x) = (q, X)$ — PDA

TM REC

(*) Inherently Ambiguous : Not Even 1 unambiguous grammar exists.

(*) Not Inher. Ambiguous : ⇒ atleast 1 " "

(*) For language over 1 symbol, FA ≅ PDA

↳ RTM → PEC.

NOTE:
① $\overline{\{ww^R \mid w \in (a,b)^*\}}$ = CFL

② $\overline{\{ww \mid w \in (a,b)^*\}}$ = CFL

③ $\overline{\{a^n b^n c^n \mid n \geq 0\}}$ = CFL

PREFIX PROPERTY: L satisfies Prefix Property, iff every string is not a prefix to every other string.

Q. Check whether the following satisfies Prefix property.

① $L=a^*$ ② $L=a^+b$ ③ $L=ab^*$ ④ $L=(a+b)^*$

⑤ $L=\{a^nb^n \mid n\geq 1\}$ ⑥ $L=\{a, ab, \dots\}$ ⑦ $L=\{ab, abb\}$

Ans: ②, ⑤.

(*) PDA by empty stack never accepts NULL.

NOTE: DPDA by Empty Stack | Accepts | set of Languages that satisfies Prefix property.

$\emptyset \cap S = \emptyset$

$\Sigma^* \cup S = \Sigma^*$

$\Sigma^* \cap S = S$.

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

$L_1 - L_2 = L_1 \cap \bar{L_2}$

CLOSURE PROPERTIES:

✓ ∀ Closed : h^{-1} and $\subseteq$ finite.

× ∀ Not Closed : $\subseteq$ and Infinite <Any>.

✓ RL ⟹ All Closed

✓ DCFL ⟹ $\bar{L}$ and prefix(L) ⟹ Closed.

× CFL ⟹ $\bar{L}, \cap, /, -, \cap$ finite ; − finite ⟹ Not Closed.

× REC/CSL ⟹ f, h, f^{-1} finite ⟹ Not Closed.

× RE ⟹ $\bar{L}$, −, − finite ⟹ Not Closed.

Q. Write F for Finite and R for Regular:

① $F_1 \cup F_2$ ② $F_1 \cap F_2$ ③ $F_1 - F_2$ ④ $\bar{F}$ ⑤ $R_1 \cup R_2$ ⑥ $R_1 \cap R_2$

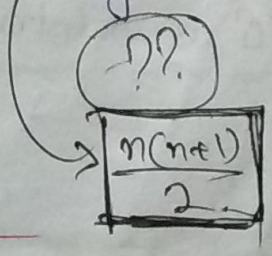
⑦ $R_1 - R_2$ ⑧ $\bar{R}$ ⑨ $F \cup R$ ⑩ $F \cap R$ ⑪ $F - R$ ⑫ $R - F$

Answer: ① F ② F ③ F ④ Not Finite ⑤ R ⑥ R ⑦ R ⑧ R ⑨ R ⑩ F = ⑪ • ⑫ = R

$n+1 \leq$ No. of substrings $\leq \frac{\Sigma n + 1}{\text{over } >1 \text{ symbol.}}$

↓ if all characters are same in string. (over 1 symbol)

Subsequence: $2^n - 1$



Q. $L_1 = a^*b$, $L_2 = ab^*$

Find L_1/L_2 and L_2/L_1 ?

Ans: a^* ε+ab^*

(*) $\overline{\text{NOT CFL}}$ = CFL (or may be not) (*) $\overline{\text{CFL}}$ = may or may not be CFL

(*) CFL ∪ DCFL = CFL (*) REG ∩ CFL = CFL

(*) CFL ∩ DCFL = Need not be CFL (CSL)

(*) CFL ∩ CFL = "

(*) DCFL ∩ DCFL = "

$\{a^n b^n c^*\} \cap \{a^* b^n c^n\}$ = not CFL

DCFL (CFL) ∩ CFL = CFL

TURING MACHINE: → It accepts/enumerates/produces REL.

(Unlimited resources and memory) TM ≅ COMPUTER (Limited Resources & Memory)

DTM ≅ NTM

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

$\delta(NTM): Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$.

(*) FA and PDA halts only after reading the entire input, but TM can halt anytime.

Enumerates Effectively { LBA → TM with bounded tape — Always halts.
HTM — Unbounded tape — Always halts.

Enumerates Not Effectively { TM — " " — halts for logic only.

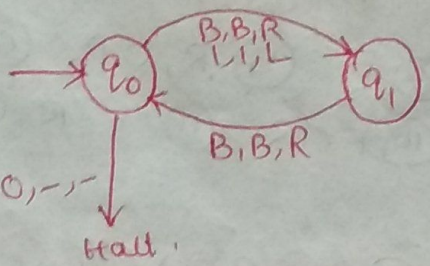
$P(FA) \leq P(DPDA) \leq P(PDA) \leq \boxed{P(LBA) \leq P(HTM) \leq P(TM)}$

(*) No. of moves in DTM ≅ Time Complexity.

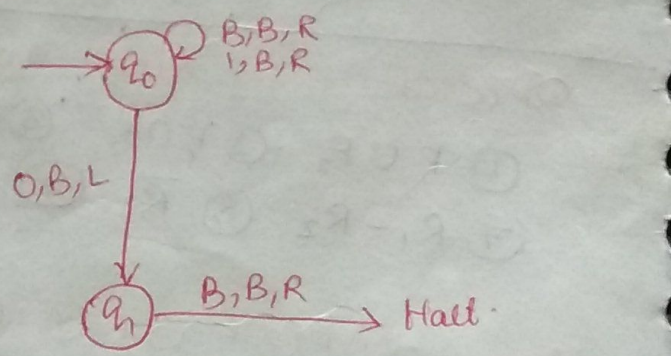
(*) No of tape cells in DTM ≅ Space Complexity.

Q) UNDERSTAND THE FOLLOWING TMS over $\Sigma = \{0, 1\}$.

(i)



(ii)



(Answer)

(i) String starting with '0' always halts.

(ii) String containing '0' as substring always halts.

(*) $\overline{F}$ = Regular but not Finite (*) $\overline{R} = R$

(*) $\overline{DCFL} = DCFL$ (*) $\overline{CFL}$ = may or may not be CFL (CSL).

(*) $\overline{CSL} = CSL$ (*) $\overline{REC} = REC$ (*) $\overline{REL}$ = need not be REL.

(*) $\overline{Decidable}$ = Decidable (*) $\overline{SD}$ = Not REL

(*) $\overline{Reg.\ But\ not\ Finite}$ = Regular (need not be Finite)

(*) $\overline{DCFL\ but\ not\ Regular}$ = DCFL but not Regular.

(*) $\overline{CFL\ but\ not\ DCFL}$ = CSL (need not be CFL) (not also DCFL)

(*) $\overline{NOT\ REL}$ = RE but ~~not~~ REC. may be Not Rel or Rel but not REC.

① Finite Language over $\Sigma \rightarrow D$
② Regular " " $\rightarrow D$
③ DCFL " " $\rightarrow D$
④ CFL " " $\rightarrow D$
⑤ CSL " " $\rightarrow D$
⑥ REC " " $\rightarrow D$

⑦ REL but not REC $\rightarrow$ SD.
⑧ REL $\rightarrow$ SD
⑨ Set of ① to ⑧ $\rightarrow$ SD.
⑩ {CFG | CFG is Ambiguous} $\rightarrow$ SD.

Decision Properties	FA (Regs)	DPDA	PDA	LBA/HTM	TM
① Halting Problem | NH	D	D	D	D	SD | NR
② Membership " | NM	D	D	D	D	SD | NR
③ Emptiness " | NEm	D	D	D	NR | SD	NR | SD
④ Finiteness " | NF	D	D	D	NR | NR	NR | NR
⑤ Totality " | NT	D	D	NR | SD	NR | SD	NR | NR
⑥ Equivalence " | NEq	D	D	NR | SD	NR | SD	NR | NR
⑦ Disjoint " | ND	D	UD	UD	UD	UD
⑧ Set Containment " | NSc	D	UD	UD	UD	UD

(*) HALTING PROBLEM : A given machine halts or Not.

(*) MEMBERSHIP " : Is string 'w' accepted by a Machine 'M'?

(*) Emptiness " : Does a machine accepts empty language ($\emptyset$).

(*) Finiteness " : Does a machine has only finite no. of strings that halts at final?

(*) Totality " : Every string of Σ^* should halt at final.
($\bar{L}$ = Some " " " " non-final).

(*) Equivalence " : For every string, both machines must halt at either final or non-final, but not both.
$\bar{L}$ = atleast 1 string, such that M_1 halts at final and M_2 halts at non-final.

(*) DISJOINT " : For every string, One machine should accept but other should not accept.
$\bar{L}$ = atleast 1 string is accepted/rejected by both.

(8) SET CONTAINMENT : Is $L_1 \subseteq L_2$? or $L_1 \cap \overline{L_2} = \emptyset$?.

Some Examples:

1. Does TM accepts given 'w'? → Membership → SD
2. Does TM accept some string? → non-emptiness → SD
3. Does TM accept only 'w'? → Finiteness → NR
4. Can u design a TM? → Decidable.
5. Can u verify a TM? → Not REL

Some Trivial Questions:

1. {DFA | DFA accepts RL}
2. {DPDA | DPDA accepts DCFL}
3. {PDA | PDA accepts CFL}
4. {TM | TM accepts REL}
5. {HTM | HTM accepts REC}

} Decidable.

Q. Identify D, SD and NR.

① Does given TM accepts Finite Language?
② Is " " " Regular " ?
③ " " " " only string w ?
④ {TM | TM reaches a state 'q' with n moves} ?
⑤ {TM | TM reaches state 'q'} ?
⑥ {TM | " " " " in more than n moves} ?
⑦ {Does a TM has N states ? or Is |TM| = 10 ?
⑧ Is |L(TM)| = 10 ?
⑨ {CFG | CFG is unambiguous}

Answers:

① Finiteness — NR ② Finiteness — NR ③ Finiteness — NR

④ L = Go 2n moves and check if TM reaches state q — logic > D.
$\bar{L}$ " " " " " " " " does not " " " — logic

⑤ L = keep checking for state q (if it has to occur, it will occur — finite moves) — logic → SD
$\bar{L}$ = As the string is not present, m/c never halts — No logic ↑

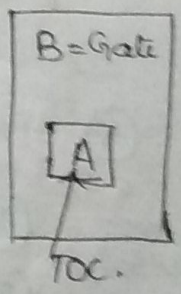
⑥ Same as 5 → SD ⑦ Membership — SD ⑧ Finiteness — NR

⑨ L = keep checking infinitely for Ambiguity — No logic > NR.
$\bar{L}$ = " " finitely " " — logic

REDUCIBILITY: ✱ $A \leq B$ means A is reducible to B.

SWAP ≤ SORT ✱ $A \propto B$ or $A \leq_m B$; A is many to one reducible to B.

✱ $A \propto_p B$ means A is polynomially reducible to B.



✱ If $A \leq B$ and $B \leq A$, then A == B.
✱ If $A \leq B$ and $B \leq C$, and if C is REL and B is REC, then A is REC.
✱ If $A \leq B$ and $B \leq C$, C is REC and B is REL (REC), then A is REC.

↓

Tricky Question.

Assume $A \leq B$, then Answer the following.

1. If B is decidable, then A is (D/UD)?
2. If A " " , then B is _____
3. If A is undecidable, then B is _____
4. If B is finite, then A is _____
5. If A is infinite, then B is _____
6. If B takes 100 days, then A takes _____
7. If A " " " , " B " _____
8. If $B = \theta(n^3)$, then A = _____
9. If $A = \theta(n^3)$, then B = _____
10. If A is NR, then B is _____
11. If A is REL, then B is _____
12. If B is REL, then A is _____

STUDY Theory of Complexity from Class Notes

Answers: 1. D 2. may or may not be D 3. UD 4. finite 5. Infinite 6. $\leq$ 100 days 7. $\geq$ 100 days 8. $O(n^3)$ 9. $\Omega(n^3)$ 10. NR 11. may or may not be REL 12. REL

SHORTCUT FOR REGULAR CLOSURE:

$X \mid \cap \cup - / \mid R.L = X$

NOTE: REG − CFL = REG ∩ $\overline{CFL}$ = REG ∩ CSL = CSL

(*) SPECIAL PROBLEMS ON REGULARS and NON-REGULARS:

Identify Regulars:

1. $\{w \mid w \in (0,1)^*, n_{01}(w) = n_{10}(w)\}$
2. $\{ \ldots, n_{00}(w) = n_{11}(w) \}$ — Not Reg (Binary exp. possible)
3. $\{ \ldots,$ Decimal eq. (w) is div. by 1024$\}$
4. $\{ \ldots, \ldots 27\}$
5. $\{w \mid w \in (a,b)^*$, every prefix (s) of w satisfies $|n_a(s) - n_b(s)| \leq K\}$

Answers: 1. Regular 2. Non-Regular 3. Regular — 11 states 4. Regular — 4 states 5. Regular

PUMPING LEMMA: (For RLs)

→ It is a proof for RLs (or It satisfies Regular Languages).
→ It is used to prove Non-regular Languages by Contradiction.
→ It can only be applied to Infinite Languages, as finite languages are already Regular.
→ It cannot be used to check given language is regular or not.
→ It can only be used to check why a language is non-regular.

Algorithm: ① Choose pumping Constant (min. no. of states in DFA)
② Choose a string w, $|w| \geq$ Pumping Constant.
③ Divide w into 3 parts, $w = xyz$, such that, $y \neq \varepsilon$, and $|xy| \leq$ Pumping Constant.
④ $\forall i \geq 0$, $xy^iz \in L$ iff L is Regular.

E.g: $L = (a+b)^*aaa$

STEP 1: Pumping Constant ≥ 4
STEP 2: Consider $w = baaa$, such that $|w| \geq 4$.
STEP 3: Divide w into 3 parts, such that $x = \varepsilon, y = b, z = aaa$.
STEP 4: $\forall i \geq 0$, $\varepsilon b^i aaa \in L$

∴ L is Regular.

E.g: $L = a^n b^n$

STEP 1: Pumping Constant ≥ 2
STEP 2: Consider $w = a^n b^n$ such that $|w| \geq 2$
STEP 3: Divide w into 3 parts, $x = a^{n-1}$, $y = ab$, $z = b^{n-1}$
STEP 4: $\forall i \geq 0$ $a^{n-1}(ab)^i b^{n-1} \in L$, when $i=0$, $a^{n-1}b^{n-1} \in L$
$i=1$, $a^{n-1}abb^{n-1} \in L$
$i=2$, $a^{n-1}ababb^{n-1} \notin L$

∴ L is not RL by Contradiction.

ARDEN'S THEOREM: If $R = RP + Q$, then $R = QP^*$ where P doesnot Contain ε.

NOTE: If P contains ε, then R has infinite no. of solutions.

NOTE: FA Cannot do - Multiplication of 2 arbitrary Numbers.
→ FA can do — Multiplication of 1 " Number with a constant (by fixing the no. of multiplications).

SIMPLIFICATION OF CFG: (For Parsers to clearly understand CFG)

1. DELETE NULL PRODUCTIONS
2. " UNIT "
3. " USELESS "

Given CFG
$S \to Aa|\varepsilon$
$A \to aB|AB|\varepsilon$
$B \to Sa|Ab|a$

✳ Elimination of NULL Productions:

(a) Delete $S \to \varepsilon$

$S \to Aa$
$A \to aB|AB|\varepsilon$
$B \to Sa|Ab|a|a$

(b) Delete $A \to \varepsilon$.

$S \to Aa|a$
$A \to aB|AB|B$
$B \to Sa|Ab|a|b.$

✳ Elimination of UNIT PRODUCTION ($A \to B$)

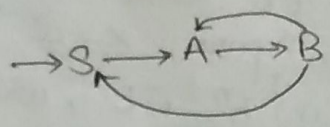
(a) $S \to Aa|a$
$A \to aB|AB|Sa|Ab|a|b$
$B \to Sa|Ab|a|b.$

✳ Elimination of USELESS PRODUCTION

STEP 1: Find NT that do not derive any string and delete them.

$NT = \{S, A, B\}$ ∴ No deletion.
⇓ ⇓ ⇓
a a,b a,b

STEP 2: Draw an edge (directed) from A to B if $A \to \ldots B \ldots$



∴ $S \to Aa|a$
$A \to aB|AB|Sa|Ab|a|b$
$B \to Sa|Ab|a|b.$
→ Simplified CFG.

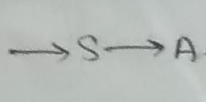
→ Consider: $S \to Aa|Bb$, $A \to e|Db$, $D \to a|Eb$, $E \to Ea$, $C \to g$, $F \to h|e$

STEP 1: $NT = \{S, A, D, E, C, F, B\}$
⇓ ⇓ ⇓ X ⇓ ⇓ X
ea e a g h.

Delete all productions containing E & B

$S \to Aa$
$A \to e|Db$
$D \to a$
$C \to g$
$F \to h|e$

STEP 2:



∴ Simplified CFG =

$S \to Aa$
$A \to e|Db$
$D \to a$

Points to Be NOTED:

(*) If a DFA has 'x' states and 'y' i/p symbols (alphabets), with designated Initial State, # DFAs = $x^{(x \times y)}$.

(*) No. of Functions for $F: A \to B = B^A$.

(*) In CNF, For generating $|w| = n$, # Steps $= (2n-1)$.

$S \to \varepsilon$ only S. not A, B, etc...

(*) In CNF, $A \to VV$ or $A \to T$ ($V \in NT$, $T \in$ Terminals).

(*) In GNF, $A \to TVV\ldots$ or $A \to T$ (—— " ——).

(*) If a TM has read only input, but has ≥ 2 Counters, then it is equivalent to basic model of TM.

(*) TM that accepts only CSLs, cannot leave their input.

(*) The language of extended Regular Expressions whose complement is empty is REC but not CSL.

(*) Universal TM $\cong$ TM.

(*) Set of all 'N' that are powers of 47 in Unary Representation is CSL but not CFL.

(*) Time Complexity of CYK Algorithm (BUP with dynamic programming) is $O(n^3)$.

(*) Graph Coloring Problem is always decidable (Intractable — NP Complete).

(*) Register Optimization can be solved using Graph Coloring.

(*) $(A \cap B \cap C)^c = A^c \cup B^c \cup C^c$ [where A, B, C are conditions] (and / or)

(*) $(RL)^*$ is not always infinite, e.g., $\emptyset^* = \varepsilon$.

(*) $(RL) \cdot (Non\text{-}RL)$ may be RL or not, e.g. $\emptyset \cdot a^n b^n = \emptyset$.

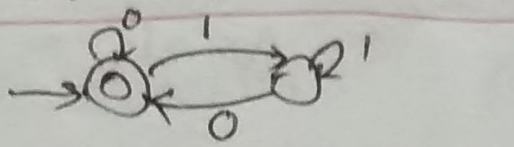
(*) Checking if a grammar is RG (/CFG/CSG/REC/RE) or not is always decidable.

(*) TM accepts REC → not decidable. [Not REC]

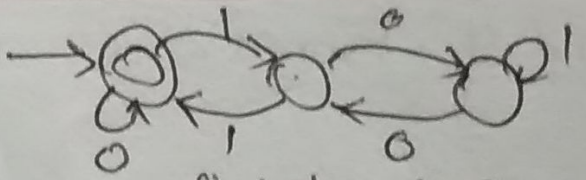
Only TM accepts RE → decidable. [REC]

(*) Binary numbers that are divisible/not by 'n' has n states.

(No dead State)



divisible by 2



divisible by 3.

(*) Every TM can be assigned to a unique Binary Number.