

Theory of Computation (ToC) Short Notes

Short Notes | Hindi + English

1

Basics & Finite Automata

Alphabet, String, Language • DFA • NFA • ϵ -NFA • NFA $\leftrightarrow$ DFA • Minimization

2

Regular Expressions & Regular Languages

RE • Kleene's Theorem • Arden's Theorem • Pumping Lemma • Closure • Moore / Mealy

3

Context Free Grammar & Languages

CFG • Derivation • Parse Tree • Ambiguity • Simplification • CNF / GNF • Pumping Lemma for CFL

4

Push Down Automata

PDA • ID • Final State / Empty Stack • DPDA • PDA $\leftrightarrow$ CFG • Two Stack PDA

5

Turing Machine & Decidability

TM • Variants • UTM • Church-Turing • Recursive / RE • Halting Problem • PCP

Theory of Computation (संगणना सिद्धांत)

Definition

Computer science की वह शाखा जो **computation के mathematical models**, कौन-सी problems computer द्वारा solve हो सकती हैं तथा उन्हें solve करने के लिए आवश्यक **resources** का अध्ययन करती है।

MAJOR AREAS

1. Automata Theory

2. Formal Languages

3. Computability Theory

4. Complexity Theory

Basic Terms

Term	Meaning (English + हिन्दी)	Example
Alphabet (Σ)	A finite, non-empty set of symbols । Symbols का finite और non-empty set; Σ (Sigma) से represent।	$\Sigma = \{0,1\}$ (binary alphabet); $\Sigma = \{a,b,c\}$
String	किसी alphabet के symbols की finite sequence । Length = symbols की संख्या।	0101 valid string, $ 0101 = 4$
Empty String	जिस string में कोई symbol नहीं; ϵ (epsilon) से represent	$ \epsilon = 0$
Language	किसी alphabet से बनी strings का कोई भी set (Language = Set of Strings)	$\Sigma = \{0,1\}$, $L = \{0, 1, 01, 10\}$
Automaton	Abstract mathematical machine जो input symbols पढ़कर defined rules के अनुसार state बदलती है	DFA, NFA, PDA, Turing Machine
Grammar	किसी language की valid strings generate करने के rules का set	$G = (V, T, P, S)$

GRAMMAR $G = (V, T, P, S)$ | **V** = Variables / Non-terminals | **T** = Terminals | **P** = Productions | **S** = Start symbol

Finite Automata / DFA

Finite Automaton

ऐसा mathematical model जिसमें **finite number of states** होते हैं।

DFA (Deterministic Finite Automaton)

Finite automaton जिसमें **हर state और हर input symbol** के लिए **exactly one transition** होता है।

Formal definition: $M = (Q, \Sigma, \delta, q_0, F)$, $\delta: Q \times \Sigma \rightarrow Q$

Q = finite set of states • **Σ** = input alphabet • **δ** = transition function • **q_0** = initial state • **F** = final/accepting states

State Transition Graph

Circle $\rightarrow$ State • Arrow $\rightarrow$ Transition • **Double circle** $\rightarrow$ **Final state** • **Incoming arrow** $\rightarrow$ **Initial state**

Transition Table

Transitions को table form में दिखाना। Example: $L =$ strings ending with 1 ($\Sigma = \{0,1\}$) - $q_0: (0 \rightarrow q_0, 1 \rightarrow q_1)$, $q_1: (0 \rightarrow q_0, 1 \rightarrow q_1)$

Language of DFA

DFA द्वारा accepted सभी strings का set: $L(M) = \{w \mid M \text{ accepts } w\}$ । Example: 1 पर end होने वाली strings $\rightarrow L = \{1, 01, 11, 001, 101, \dots\}$

NFA (Non-Deterministic Finite Automaton)

Automaton	Transitions / Meaning
DFA	Exactly one transition (हर state और input symbol पर)
NFA	State और input symbol के लिए zero, one या multiple transitions हो सकते हैं
ϵ -NFA	Automaton बिना input symbol consume किए state बदल सकता है (ϵ -transition: $q_0 \xrightarrow{\epsilon} q_1$)
Language of NFA	अगर किसी input string के लिए कम से कम एक path final state तक पहुँचे, तो string accepted

EXAM POINT ϵ -transition consumes no input symbol. | NFA accepts if at least one possible path accepts the string.

Equivalence of NFA and DFA

NFA $\equiv$ DFA

NFA और DFA computational power में **equivalent** हैं; हर NFA के लिए equivalent DFA बनाया जा सकता है। **Technique:** **Subset Construction / Powerset Construction** (NFA $\rightarrow$ DFA conversion)। **Exam Point:** DFA and NFA recognize exactly the **Regular Languages**।

Minimization of Finite Automata

DFA Minimization

Equivalent states को **merge** करके **minimum number of states** वाला equivalent DFA बनाना। **Methods:** Partition method, Table-filling method, Myhill-Nerode approach। यदि दो states सभी possible future inputs पर same behavior लेती हैं, तो वे equivalent हो सकती हैं।

Distinguishing One String from Another

दो strings को distinguish करने के लिए ऐसा **suffix** खोजा जाता है जिससे एक string accepted हो और दूसरी rejected। Example: $x = 0, y = 1$, suffix 1 लगाने पर $01 \rightarrow$ accepted, $11 \rightarrow$ rejected, इसलिए suffix 1 दोनों को distinguish करता है। यह concept **DFA minimization** और **Myhill-Nerode theorem** में महत्वपूर्ण है।

2 Regular Expressions & Regular Languages

Regular Expression (नियमित अभिव्यक्ति)

Definition

Formal notation जिसका उपयोग Regular Language को describe करने के लिए किया जाता है। Example: $(0+1)^*$ $\rightarrow$ सभी binary strings।

Operator	Symbol	Example	Meaning
Union	+ or	a+b	a OR b
Concatenation	(juxtaposition)	ab	पहले a, फिर b
Kleene Star	*	a*	ϵ , a, aa, aaa, ...

PRECEDENCE (Exam Trick) Kleene Star (*) > Concatenation > Union (+) → Star → Concatenation → Union

Algebraic Laws	Rules
Identity Laws	$R + \emptyset = R$ $R\epsilon = R$
Zero Law	$R\emptyset = \emptyset$
Idempotent Law	$R + R = R$
Star Laws	$R^* = \epsilon + RR^*$ $R^* = \epsilon + R^*R$

Kleene's Theorem & Conversions

Kleene's Theorem

Regular Expressions and Finite Automata are equivalent in expressive power (दोनों Regular Languages represent करते हैं)। इसलिए **RE → FA** और **FA → RE** दोनों संभव हैं।

Conversion	Approach
RE → FA	Common approach: RE → ϵ-NFA → DFA . Example: $RE = a^* \rightarrow FA$ zero या अधिक a accept करेगा
DFA → RE	State elimination method या Arden's theorem

Arden's Theorem

Regular expressions और finite automata के **equations solve** करने में। यदि **$X = AX + B$** और ϵ, A की language में नहीं है, तो **$X = A^*B$** ।

Non-Regular Languages & Pumping Lemma

Non-Regular Language

जिन्हें कोई finite automaton recognize नहीं कर सकता। Example: **$L = \{0^n1^n \mid n \geq 0\}$** (0 और 1 की संख्या equal)। **Why?** Finite automaton के पास unlimited counting के लिए पर्याप्त memory नहीं होती।

Pumping Lemma for Regular Languages

यह prove करने के लिए कि कोई language **NOT regular** है। यदि L regular है, तो pumping length **p** exist करती है ताकि sufficiently long string **$w = xyz$** हो, जहाँ: 1) $|xy| \leq p$ 2) $|y| > 0$ 3) $xy^iz \in L$ for all $i \geq 0$ ।

APPLICATION: $L = \{0^n1^n\}$ is not regular मान लो L regular है, pumping length = p। चुनो **$w = 0^p1^p$** । Pumping lemma के अनुसार y केवल initial 0s में होगा। y को pump/remove करने पर (xy^0z) 0 और 1 की संख्या unequal हो जाएगी, इसलिए string L में नहीं → **L is not regular.**

Closure & Decision Properties of Regular Languages

Property	Details
Closure (closed under)	Union, Intersection, Complement, Difference, Concatenation, Kleene Star, Reversal (L_1, L_2 regular $\rightarrow L_1 \cup L_2, L_1 \cap L_2, L_1^*$ भी regular)
Decidable problems	Membership (string language में है?), Emptiness (language empty है?), Finiteness , Equivalence (दो automata same language?), Universality (सभी strings accept?)

Finite Automata with Output

Feature	Moore Machine	Mealy Machine
Output depends on	केवल current State : Output = $f(\text{State})$	State + Input दोनों: Output = $f(\text{State}, \text{Input})$
Output shown on	State	Transition
Response	Generally slower	Generally faster
States	May require more	May require fewer
Example	$q_0 \rightarrow \text{Output } 0, q_1 \rightarrow \text{Output } 1$	-

Equivalence & Applications

Moore और Mealy machines एक-दूसरे में convert हो सकती हैं (दोनों finite-state input-output behavior represent करती हैं)। **FA के Applications:** Lexical analysis (compiler), Pattern matching, Text searching, Regular expression processing, Digital circuit design, Protocol verification, Spell checking, Input validation।

Limitations of Finite Automata

Unlimited counting नहीं कर सकता; Nested structures recognize नहीं कर सकता; Balanced parentheses और $0^n 1^n$ जैसी languages recognize नहीं कर सकता। ऐसी problems के लिए ज़्यादा memory वाले models जैसे **PDA** चाहिए।

3 Context Free Grammar & Languages

CFG & CFL

CFG (Context Free Grammar)

Formally $G = (V, T, P, S)$ । Production form: $A \rightarrow \alpha$ (A एक single non-terminal)। Example: $S \rightarrow aSb \mid \epsilon \rightarrow$ language $\{a^n b^n \mid n \geq 0\}$ ।

CFL (Context Free Language)

किसी CFG द्वारा generated language। Example: $L = \{a^n b^n \mid n \geq 0\}$ । **Important:** Every Regular Language is CFL, but every CFL is not Regular.

Derivation, Parse Tree

Term	Meaning
Derivation	Grammar rules apply करके start symbol से terminal string प्राप्त करना। Example: $S \rightarrow aSb \rightarrow aaSbb \rightarrow aabb$
Leftmost Derivation	हर step में सबसे left non-terminal replace
Rightmost Derivation	हर step में सबसे right non-terminal replace
Parse / Derivation Tree	Grammar derivation को tree form में represent करता है (hierarchical structure दिखाता है)

Ambiguity

Ambiguous Grammar

यदि किसी string के लिए एक से अधिक **parse trees / leftmost derivations / rightmost derivations** possible हों।
Example: $E \rightarrow E + E \mid E * E \mid id$; expression **id + id * id** के multiple interpretations।

Inherently Ambiguous Language

यदि किसी CFL के लिए **कोई भी unambiguous CFG मौजूद नहीं** है। **Exam Point:** Ambiguous grammar $\neq$ necessarily inherently ambiguous language; grammar बदलकर ambiguity हटाई जा सकती है, inherently ambiguous language में नहीं।

Ambiguous $\rightarrow$ Unambiguous CFG

Ambiguity हटाने के लिए: **Operator precedence, Associativity, Grammar restructuring** (जैसे multiplication को addition से higher precedence)।

Simplification of CFG

Useless Symbols

जो किसी terminal string को derive करने में useful नहीं: **1) Non-generating symbols** (terminal string generate नहीं कर सकते) **2) Unreachable symbols** (start symbol से reach नहीं)।

Simplification Steps

1) ϵ -productions remove 2) Unit productions remove 3) Useless symbols remove। **Goal:** Equivalent और simpler grammar।

Normal Forms

Normal Form	Production Form	Exam Trick
CNF Chomsky Normal Form	$A \rightarrow BC$ या $A \rightarrow a$ (special case में start symbol के लिए ϵ production allowed)	CNF $\rightarrow$ Two variables
GNF Greibach Normal Form	$A \rightarrow a\alpha$ (production का right side terminal से शुरू)	GNF $\rightarrow$ Starts with terminal

Properties of CFL

Property	Details
Closure	Closed under: Union, Concatenation, Kleene Star Generally NOT closed under: Intersection, Complement, Difference (CFL $\cap$ CFL is not necessarily CFL)
Decision	Emptiness - Decidable • Finiteness - Decidable • Membership - Decidable • Equivalence (दो CFGs same language?) - Undecidable in general

Pumping Lemma for CFL

यह prove करने के लिए कि कोई language **NOT context-free** है। Common form: $z = uvwxy$, जहाँ $|vwx| \leq p$ और $|vx| > 0$, और $uv^iwx^iy \in L$ for suitable $i \geq 0$ । **Exam Point:** Regular language $\rightarrow$ Pumping Lemma with xyz ; CFL $\rightarrow$ Pumping Lemma with $uvwxy$ ।

4 Push Down Automata

PDA (Push Down Automata)

PDA = Finite Automaton + Stack। Stack additional memory देता है, इसलिए PDA $a^n b^n$ जैसी languages recognize कर सकता है। PDA मुख्यतः **Context Free Languages** को recognize करता है: **PDA $\leftrightarrow$ CFL**।

Formal Definition

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

Q = States • Σ = Input alphabet • Γ = **Stack alphabet** • δ = Transition function • q_0 = Initial state • Z_0 = **Initial stack symbol** • F = Final states

Instantaneous Description (ID)

PDA की current configuration: (q, w, γ) - q = Current state, w = Remaining input, γ = Stack contents।

Example: $(q_1, abb, AZ) \rightarrow$ current state q_1 , input में abb बाकी, stack contents AZ ।

Acceptance by PDA

Acceptance	Meaning
By Final State	Input पूरी तरह consume होने के बाद PDA final state में पहुँचे $\rightarrow$ accepted (Input consumed $\rightarrow$ Final State $\rightarrow$ Accept)
By Empty Stack	Complete input consume होने के बाद PDA का stack empty हो जाए $\rightarrow$ accepted

IMPORTANT General **nondeterministic** PDA के लिए acceptance by Final state और Empty stack **equivalent in expressive power** हैं।

DPDA vs NPDA

DPDA (Deterministic PDA)

हर configuration के लिए transition **uniquely determined**। **DPDA की power general NPDA से कम होती है। Exam**

Point: DPDA recognizes **Deterministic CFLs (DCFLs)**; हर CFL, DCFL नहीं है।

Equivalence of PDA and CFG

Fundamental Theorem

CFGs and Nondeterministic PDAs are equivalent in expressive power: **CFG $\rightarrow$ PDA** और **PDA $\rightarrow$ CFG** दोनों conversions possible।

Conversion	Idea
CFG $\rightarrow$ PDA	CFG की productions को PDA transitions में convert; PDA stack में grammar symbols रखे जाते हैं और productions के अनुसार replace किए जाते हैं
PDA $\rightarrow$ CFG	PDA के states और stack behavior के आधार पर equivalent CFG construct

EXAM TRICK Regular Language $\leftrightarrow$ FA | Context-Free Language $\leftrightarrow$ PDA

Two Stack PDA

इसमें दो **stacks** होते हैं। **Important result:** A two-stack PDA is computationally **equivalent to a Turing Machine**, क्योंकि दो stacks मिलकर unbounded read/write storage की तरह काम कर सकते हैं।

5 Turing Machine & Decidability

Turing Machine (TM)

Definition

एक **powerful theoretical computational model** जो Read, Write, Move left/right और Change state कर सकती है।

Components: 1) Infinite tape 2) Read/write head 3) Finite control 4) Transition function। Tape cells में symbols होते हैं (... | 0 | 1 | 1 | 0 | B | B | ...); head symbol read/write और left/right move कर सकता है।

Formal Definition

M = (Q, Σ , Γ , δ , q_0 , B, F): Q = States, Σ = Input alphabet, Γ = **Tape alphabet**, δ = Transition function, q_0 = Initial state, B = **Blank symbol**, F = Final states।

ID of TM & Acceptance

Instantaneous Description: Current state + Tape contents + Head position। **Acceptance:** TM किसी string पर accepting state में पहुँचे तो string accepted। TM किसी string पर **Accept** कर सकती है, **Reject** कर सकती है, या **Forever loop** कर सकती है (यही distinction decidability में महत्वपूर्ण है)।

Variant of TM	Meaning
Multi-Tape TM	Multiple tapes
Multi-Track TM	एक tape में multiple tracks
Non-Deterministic TM	एक configuration से multiple possible transitions
Multi-Head TM	Multiple read/write heads
Universal TM	दूसरी Turing machines को simulate कर सकती है

IMPORTANT इनमें से कई variants standard TM के **equivalent computational power** रखते हैं।

TM as Computer of Integer Functions

TM algorithms के माध्यम से computable functions calculate कर सकती है। Example: Input 3, 4 → TM Addition perform करके 7 produce करती है।

Universal Turing Machine (UTM)

ऐसी TM जो किसी अन्य TM का **description और input** लेकर उस machine को **simulate** कर सकती है। (Description of M + Input → Universal TM → Simulates M) **Exam Point:** Universal TM = Machine that simulates other Turing machines.

Church's Thesis (Church-Turing Thesis)

जिसे कोई **effective algorithmically computable** method द्वारा calculate किया जा सकता है, उसे Turing Machine द्वारा भी compute किया जा सकता है। **Important:** यह एक **thesis** है, mathematical theorem नहीं।

Recursive vs Recursively Enumerable (RE) Languages

Feature	Recursive (Decidable)	Recursively Enumerable (RE)
Accept member	Yes	Yes
Reject non-member	Yes	May reject or loop
Always halts	Yes	No
Also called	Decidable	Turing-recognizable
Definition	TM हर input पर halt करती है: string language में है → Accept, नहीं है → Reject	Language में string होने पर eventually accept; न होने पर reject या forever loop

RELATIONSHIP Every Recursive Language is RE, लेकिन every RE Language is not necessarily Recursive। | Key: Recursive = Always halts.

Undecidability

Halting Problem

Given a program/Turing machine और input, क्या वह eventually halt करेगी या forever चलेगी? **Alan Turing** ने दिखाया कि कोई general algorithm सभी programs के लिए सही-सही determine नहीं कर सकता। **Halting Problem → Undecidable.**

Undecidable Problem

यदि किसी problem के लिए ऐसा algorithm मौजूद नहीं जो प्रत्येक valid input पर **finite time में सही answer** दे, तो वह undecidable है।

FAMOUS

Halting Problem

TM acceptance problem

TM equivalence problem

Post Correspondence Problem

Undecidable Problem (about TM)	Question
Acceptance Problem	क्या TM M input w को accept करेगी? (generally undecidable)
Halting Problem	क्या TM M input w पर halt करेगी?
Equivalence Problem	क्या दो TMs same language accept करती हैं?
Emptiness Problem	क्या TM की language empty है? (undecidable in general)

PCP, MPCP & Recursive Function Theory

Post Correspondence Problem (PCP)

Domino/tile pairs दिए जाते हैं: Top: $x_1 x_2 x_3 \dots$ / Bottom: $y_1 y_2 y_3 \dots$; देखना है कि क्या ऐसी sequence मौजूद है जिससे **Top string = Bottom string** हो जाए। **PCP is undecidable.**

MPCP (Modified PCP)

PCP का modified version जिसमें matching sequence को एक **specified first pair** से शुरू करना आवश्यक है। **MPCP is also undecidable**, और PCP की undecidability prove करने में MPCP का उपयोग होता है।

Recursive Function Theory

Computable functions और उनकी mathematical properties का अध्ययन। Computability theory यह study करती है: **"What can be computed?"**

CONCEPTS

Recursive functions

Partial recursive functions

Computability

Decidability

Recursion

Relation

Language Class	Machine
Regular Language	Finite Automata (FA)
Context-Free Language	Push Down Automata (PDA)
Recursive / RE Language	Turing Machine (two-stack PDA $\equiv$ TM)