Automata & Complexity

Short Notes based on ***Exit Exam***

Zemzem labs

1. Basic Language Theory

Alphabets, Strings, Languages

- Alphabet (Σ): finite set of symbols
- String (w): finite sequence of symbols
- Language (L): set of strings over Σ
- ε = empty string
- |w| = length of w

Language Operations

- Union
- Intersection
- Concatenation
- Kleene Star (L*)
- Complement

2. Finite Automata (FA)

Finite automata recognize **regular languages**.

Types

1. **DFA – Deterministic Finite Automaton**
    - One transition per symbol
    - No ε-moves
    - Always unambiguous
2. **NFA – Nondeterministic Finite Automaton**
    - Multiple transitions allowed
    - ε-transitions allowed
    - Equivalent in power to DFA

Key Notes

- DFA ⇿ NFA (same power)
- Convert NFA → DFA using **subset construction**
- Regular languages are **closed** under union, intersection, complement, reverse

Regular Expressions

- Describe the same class as DFA/NFA
- R → FA conversion possible

Limitations

Regular languages **cannot** handle:

- Nested structures
- Matching parentheses
- Palindromes
- $\{a^n b^n\}$

3. Regular Languages & Grammars

Regular languages are generated by:

- **Regular Grammars (Type-3)**
  - Right-linear
  - Left-linear

Regular languages closed under:

- Union
- Intersection
- Complement
- Difference
- Reverse
- Concatenation
- Kleene Star

4. CFG (Context-Free Grammars)

CFGs generate **context-free languages**, recognized by **Pushdown Automata (PDA)**.

Grammar Form

$G = (V, \Sigma, R, S)$

Derivations

- Leftmost derivation
- Rightmost derivation

Parse Trees

Show structure; ambiguity arises when multiple parse trees exist.

Ambiguous Grammar

- More than one parse tree
- Eliminated using precedence/associativity or rewriting grammar

Normal Forms

- **Chomsky Normal Form (CNF)**
  $A \rightarrow BC \mid a$
- **Greibach Normal Form (GNF)**
  $A \rightarrow a\alpha$

5. Pushdown Automata (PDA)

Recognizes **context-free languages (CFLs)**.

PDA Types

- **Deterministic PDA (DPDA)** → recognizes **LR(1) subset**, not all CFLs
- **Nondeterministic PDA (NPDA)** → recognizes all CFLs

Key Notes

- PDA uses **stack** for memory
- Languages like $a^n b^n$ require PDA
- CFLs **not closed** under intersection or complement
- CFLs **closed** under union & concatenation & star

6. Turing Machines (TM)

Most powerful model of computation.

Turing Machine Definition

$M = (Q, \Sigma, \Gamma, \delta, q_0, q\_accept, q\_reject)$

Capabilities

- Recognizes **recursively enumerable languages**
- Decides **recursive languages** (if halts on all inputs)

Variants

- Multi-tape TM
- Non-deterministic TM
- TM with storage
  All equivalent in computability power (same languages).

Universal Turing Machine

A TM that can simulate any other TM.

7. Chomsky Hierarchy

Type	Grammar	Machine	Language Class
0	Unrestricted	TM	Recursively Enumerable
1	Context-Sensitive	LBA	CSL
2	Context-Free	PDA	CFL
3	Regular	FA	RL

8. Decidability & Undecidability

Decidable Problems (Recursive)

TM halts for all inputs.

Examples:

- DFA emptiness
- DFA equivalence
- Regular expression → DFA conversion
- CFL membership (CYK Algorithm)

Undecidable Problems

No TM can solve.

Examples:

- **Halting Problem**
- **Post Correspondence Problem (PCP)**
- TM equivalence
- CFG equivalence
- Grammar ambiguity

Semi-decidable Problems

- TMs may loop on some inputs
- Languages are **recursively enumerable**

9. Reductions

Used to prove undecidability or complexity hardness.

Types

- Many-one reduction
- Turing reduction

If A reduces to B and A is hard, then B is at least as hard.

Usage

- Halting problem reduces to many other undecidable problems.

10. Complexity Theory

Asymptotic Notations

- **Big O** (upper bound)
- **Ω** (lower bound)
- **Θ** (tight bound)

11. Time & Space Complexity Classes

Big Picture

- **P**: solvable in polynomial time
- **NP**: verifiable in polynomial time

- **EXP**: exponential time
- **PSPACE**: polynomial space
- **L / NL**: log space / nondet log space

Relationships (known)

P ⊆ NP ⊆ PSPACE ⊆ EXP
L ⊆ NL ⊆ P

❗ **Unknown important question:**

- **Is P = NP?** (Open problem)

12. NP-Completeness

NP Problems

- Solutions can be checked in polynomial time.

NP-Complete Problems

- In NP
- Every NP problem reduces to them

Examples:

- SAT (first NP-complete problem)
- 3-SAT
- Traveling Salesman Problem (decision version)
- Clique
- Vertex Cover
- Subset Sum

NP-Hard

- At least as hard as NP
- Not necessarily in NP
- Does not require polynomial-time verification

🌟 Exam-Ready Ultra-Condensed Summary

- Regular → DFA/NFA/Regex

- Context-Free → CFG/PDA
- CS languages → LBA
- Recursive languages → TM (decidable)
- RE languages → TM (semi-decidable)
- Halting → Undecidable
- NP-Complete → SAT, 3-SAT, TSP (decision), Clique
- P vs NP → unknown