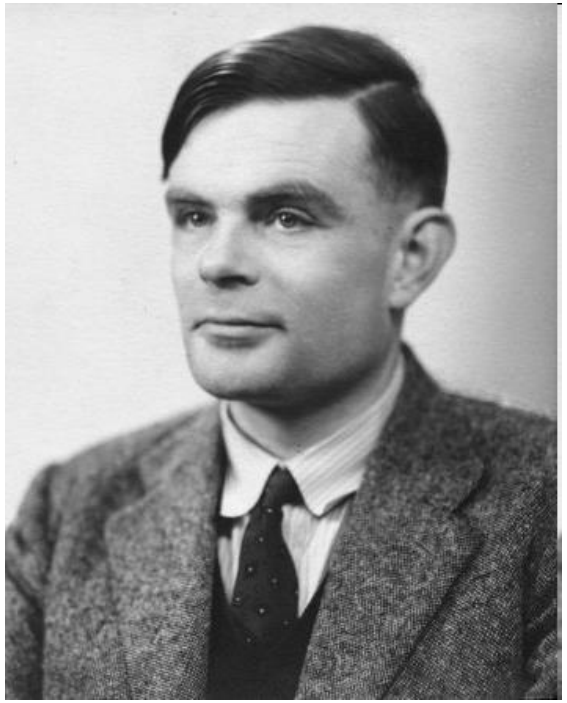
UNIT - I
WHAT IS AUTOMATA THEORY?

- *Study of abstract computing devices, or "machines"*
- Automaton = an abstract computing device
  - Note: A "device" need not even be a physical hardware!
- A fundamental question in computer science:
  - Find out what different models of machines can do and cannot do
  - The *theory of computation*
- Computability vs. Complexity

(A pioneer of automata theory)

ALAN TURING (1912-1954)

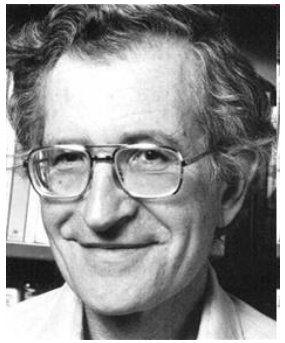
- Father of Modern Computer Science
- English mathematician
- Studied abstract machines called ***Turing machines*** even before computers existed
- Heard of the Turing test?



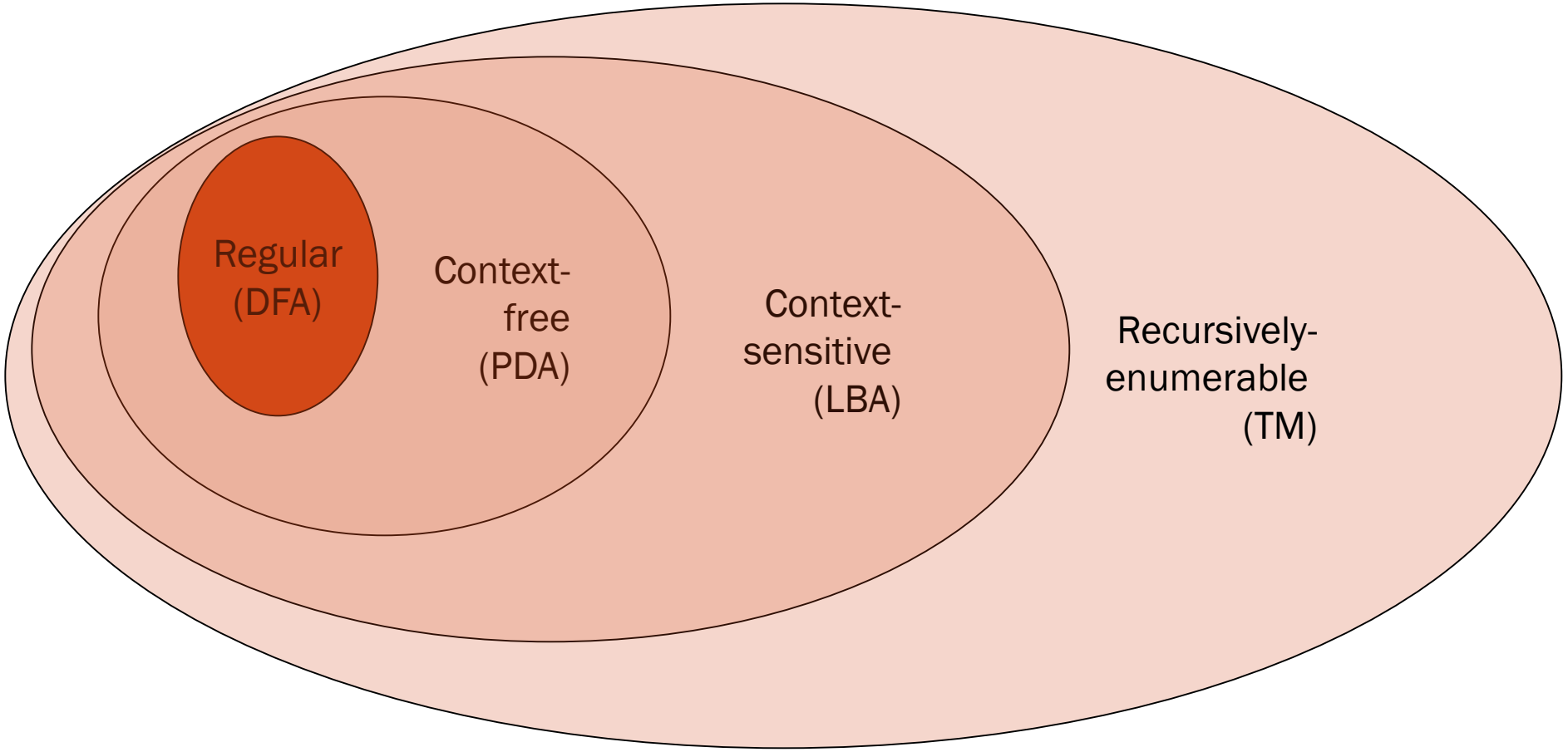
THEORY OF COMPUTATION: A HISTORICAL PERSPECTIVE

1930s	• Alan Turing studies Turing machines • Decidability • Halting problem
1940-1950s	• "Finite automata" machines studied • Noam Chomsky proposes the "Chomsky Hierarchy" for formal languages
1969	Cook introduces "intractable" problems or "NP-Hard" problems
1970-	Modern computer science: compilers, computational & complexity theory evolve

THE CHOMSKY HIERACHY



- A containment hierarchy of classes of formal languages



BRIEF HISTORY OF THEORY OF COMPUTATION

- 1936 Alan Turing invented the *Turing machine*, and proved that there exists an *unsolvable problem*.
- 1940's Stored-program computers were built.
- 1943 McCulloch and Pitts invented *finite automata*.
- 1956 Kleene invented *regular expressions* and proved the equivalence of regular expression and finite automata.

HISTORY OF THEORY OF COMPUTATION CONTD...

- 1956 Chomsky defined *Chomsky hierarchy*, which organized languages recognized by different automata into hierarchical classes.
- 1959 Rabin and Scott introduced *nondeterministic finite automata* and proved its equivalence to (deterministic) finite automata.
- 1950's-1960's More works on languages, grammars, and compilers

HISTORY OF THEORY OF COMPUTATION

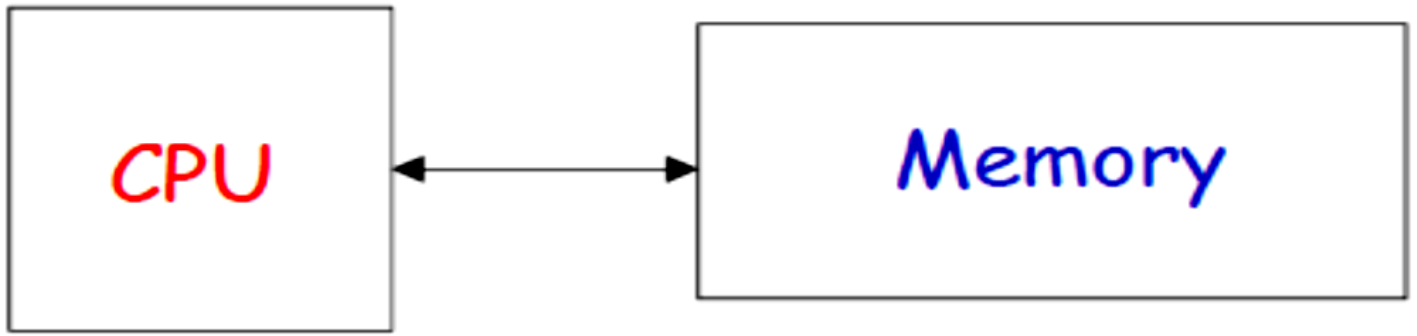
- 1965 Hartmantis and Stearns defined *time complexity*, and Lewis, Hartmantis and Stearns defined *space complexity*.
- 1971 Cook showed the first *NP-complete problem*, the *satisfiability* problem.
- 1972 Karp Showed many other NP-complete problems.

WHAT IS COMPUTATION ?

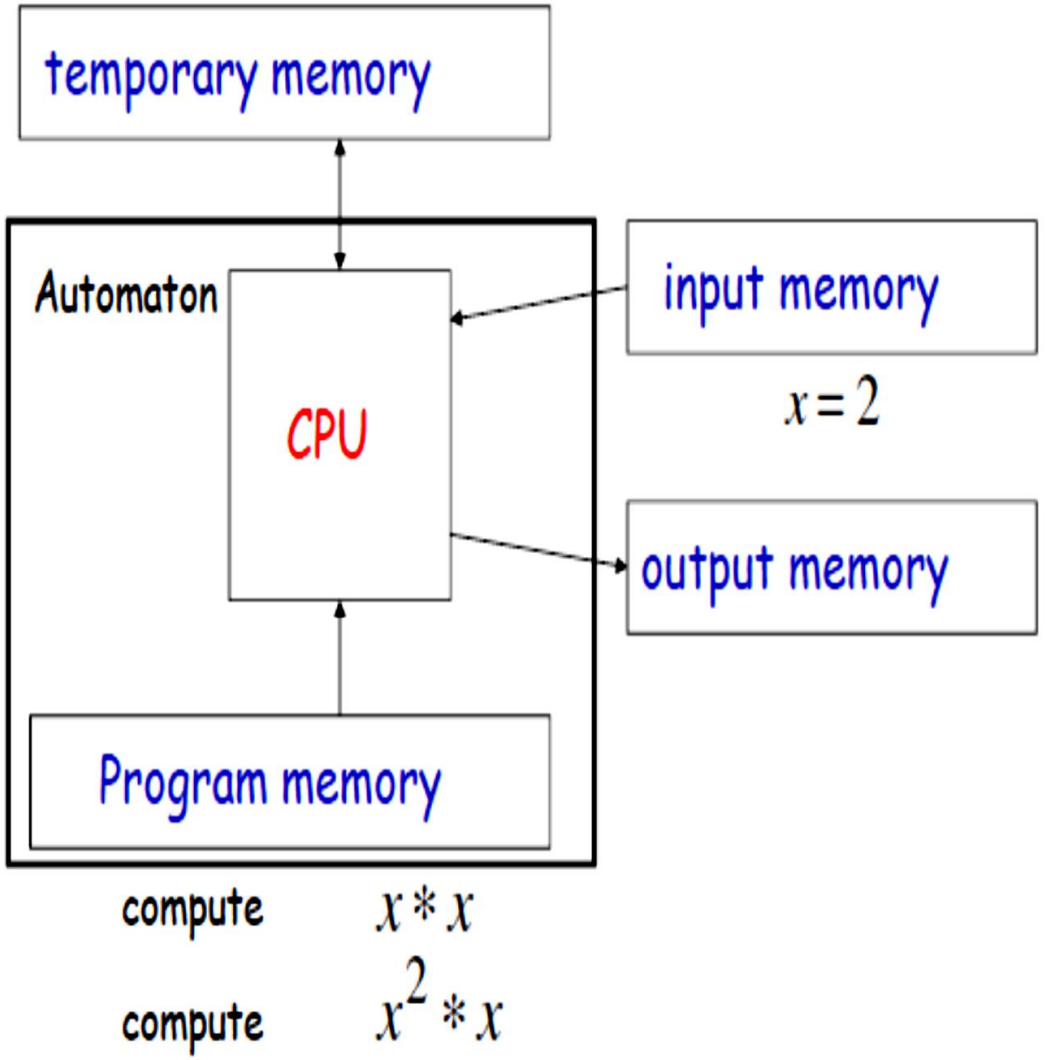
- Computation is nothing but any task that can be performed by a computer or calculator.
- **TOC:- The study of mathematical representation of computing system and their capabilities is TOC.**
  - Sequence of mathematical operations ?
    - What are, and are not, mathematical operations?
  - Sequence of well-defined operations
    - How many operations ?
      - The fewer, the better.
    - Which operations ?
      - The simpler, the better.

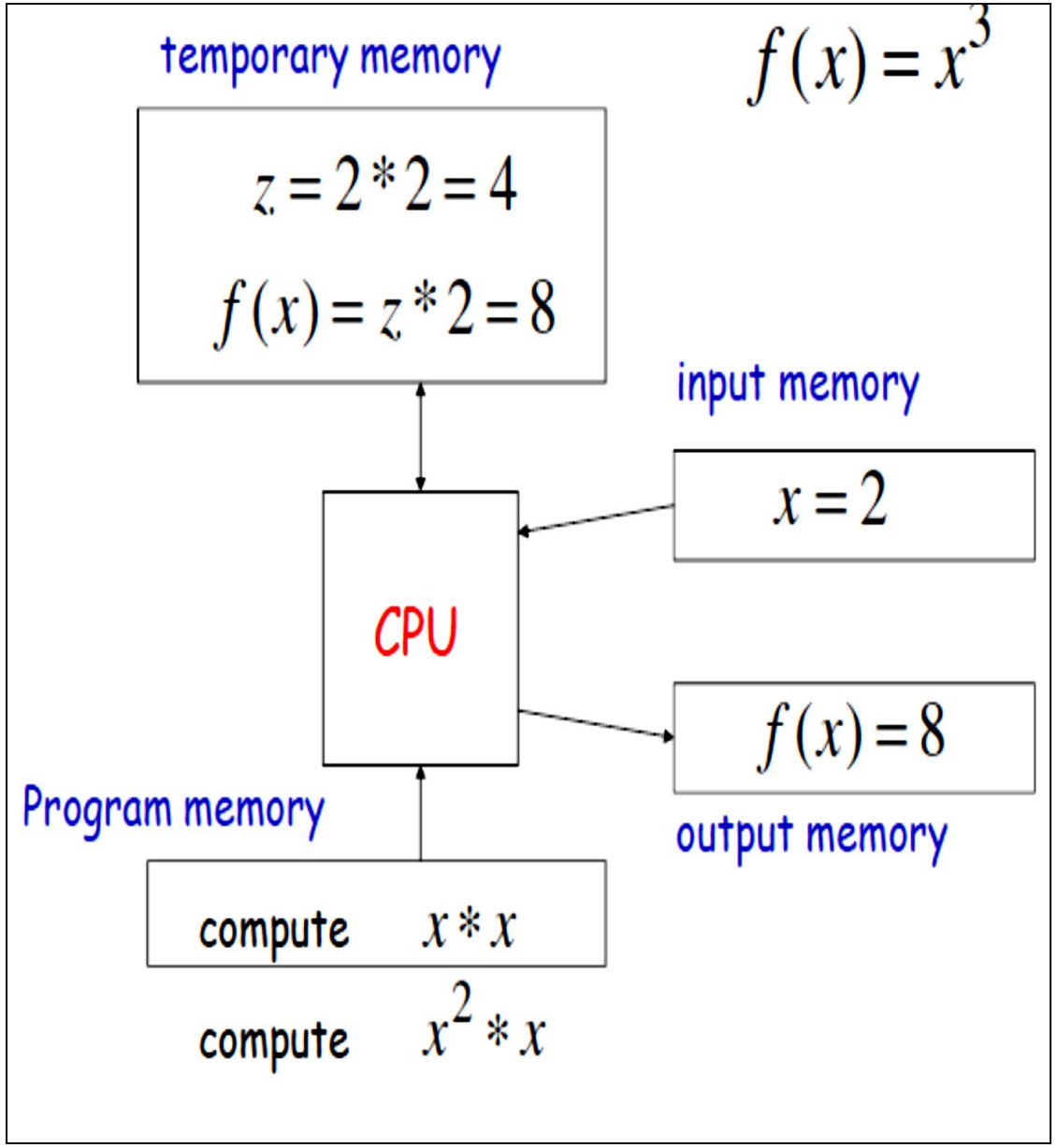
COMPUTATION

Solving problems through the mechanical, preprogrammed execution of a series of small, unambiguous steps.



Example: $f(x)=x^3$





WHAT DO WE STUDY IN THEORY OF COMPUTATION ?

- What is computable, and what is not ?
- What a computer can and can not do
- Can you make your program more efficient?

- Basis of
  - Algorithm analysis
  - Complexity theory

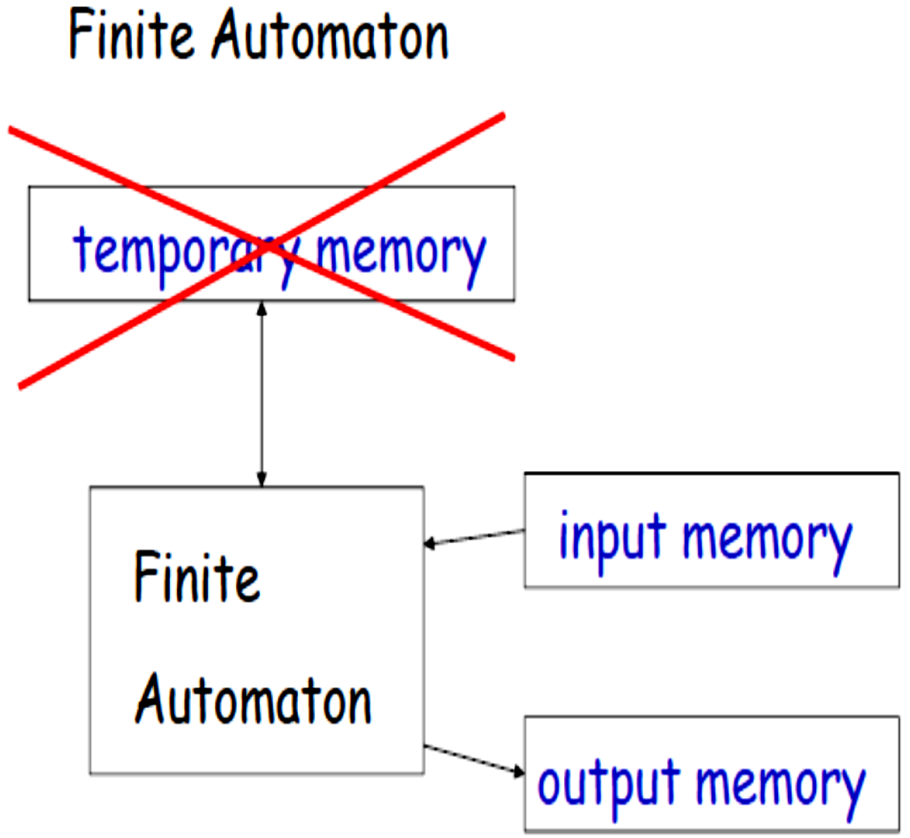
WHAT DO WE STUDY IN COMPLEXITY THEORY ?

- What is easy, and what is difficult, to compute ?
- What is easy, and what is hard for computers to do?

Different Kinds of Automata

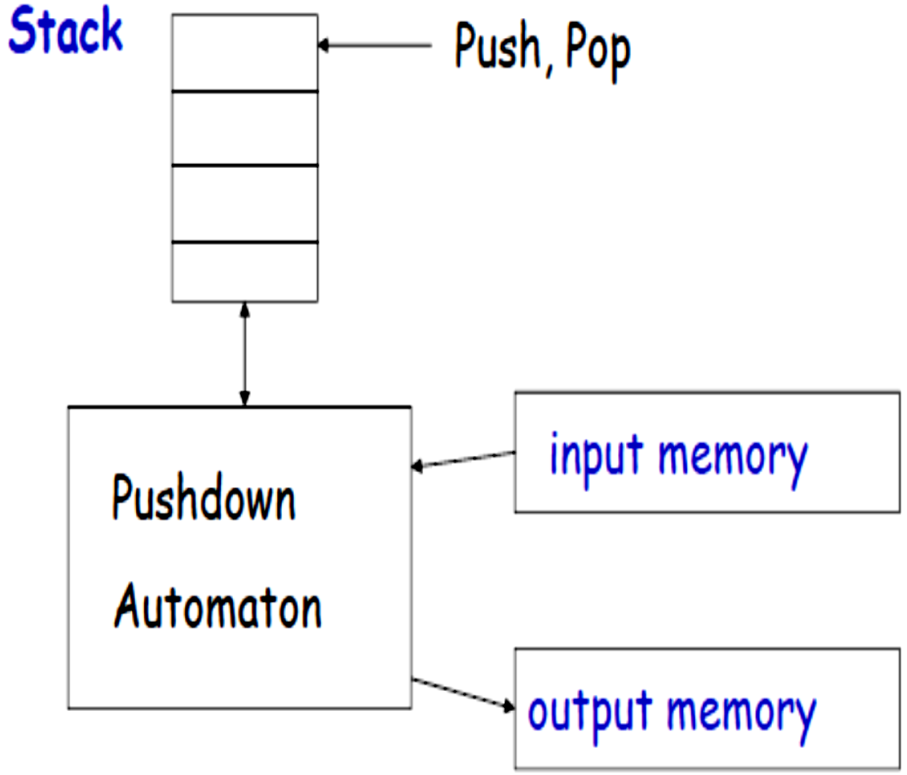
Automata are distinguished by the temporary memory

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory



Example: String Search, Decision Making, etc

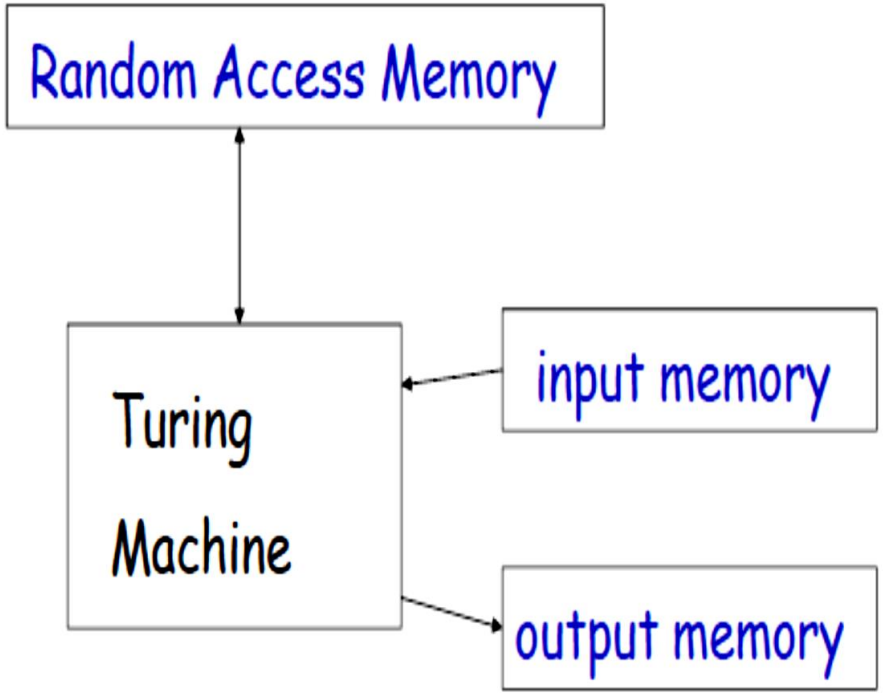
Pushdown Automaton



Example: Compilers for Programming Languages

(medium computing power)

Turing Machine



Examples: Any Algorithm (highest computing power)

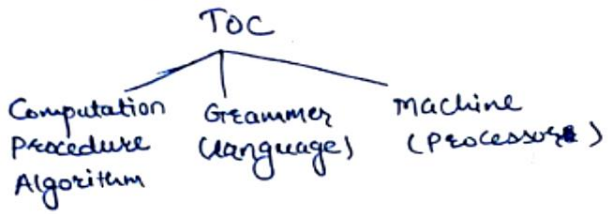
1) Theory of Automata.
2) Theory of computability
3) Theory of Complexity (DAA)

Reference Book:

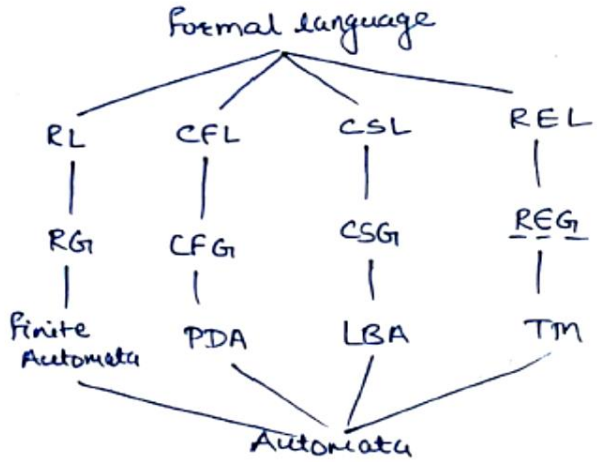
① TOC by J.D. Ullman

② TOC by Dr. K.V.N. Sunitha

③ TOC by ACE Engg. Academy



GATE/ PSV



TOC:- The study of mathematical representation of computing system and their capabilities is TOC.

Formal Language:- The language which has proper alphabet, grammer and model to recognize the language is called as formal language.

Ex:- $\Sigma = \{0, 1, 1\}$

① $L = \{01, 10, 11, 00\}$

② $L = \{0^n 1^n \mid n \geq 1\}$

③ {C, C++, java, pascal etc}

Note:- In the formal language we need to give importance only for the formation of string rather than meaning of string

Grammer:- The collection of rules which are used to generate the string is called as grammer.

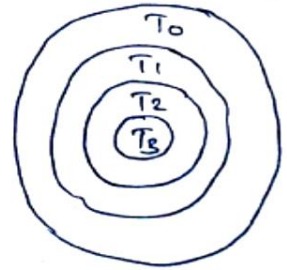
Ex:- $S \to ABC$, $A \to a$, $B \to b$, $C \to c$

$\therefore S \to aBC \to abC \to abc$

$\therefore L(G) = \{abc\}$

Note:- Grammer is generating device

Classification of Grammer:- Grammer can be classified into 4 types
① Type 0 or REG
② Type 1 or CSG
③ Type 2 or CFG
④ Type 3 or RG



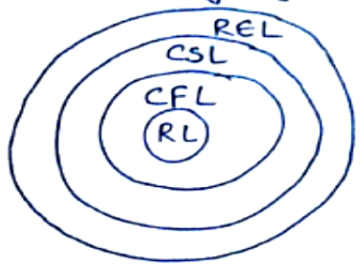
$T_3 < T_2 < T_1 < T_0$

Types of formal language :-

Classified into 4 types

① Type 0 or REL
② Type 1 or CSL
③ Type 2 or CFL
④ Type 3 or RL

Formal language can be



$$CSL = CSL \cup \{\epsilon\}$$

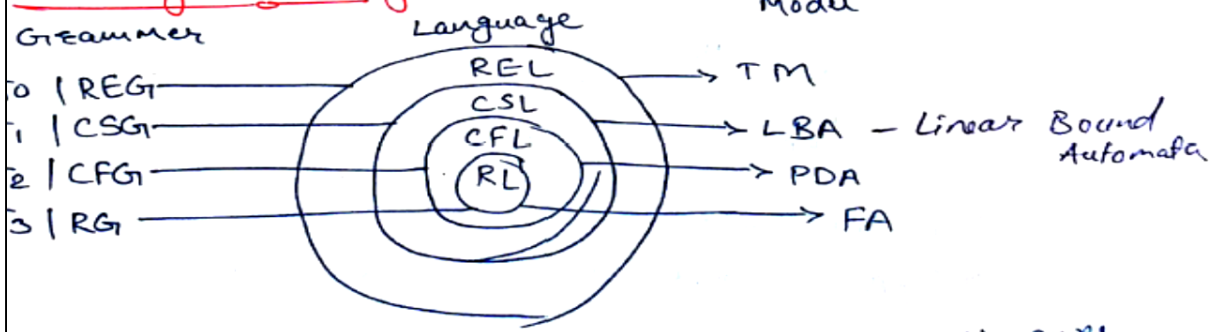
Automata :- Mathematical representation of formal language is called as automata.

Note :- Automata is recognising or accepting device

Types of Automata :-

① FA (RL) ✓
② PDA (CFL) Context Free Lang.
③ LBA (CSL) Context sensitive Language
④ TM (REL) Recursively enumerable language.

Chomsky Hierarchy :-



FA – 1 → Accepting power
PDA – 2
LBA – 3
TM – 4