Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

Turing Machines

Study Guide

Prof. Riddhi Atulkumar Mehta
CSE, PIT
Parul University

4.1 The basic model for Turing machines (TM)

Turing Machine

- A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.
- Turing Machines (TM) play a crucial role in the Theory of Computation (TOC). They are abstract computational devices used to explore the limits of what can be computed.
- Turing Machines help prove that certain languages and problems have no algorithmic solution.
- A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given.
- It consists of a head which reads the input tape. A state register stores the state of the Turing machine.
- After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left.
- If the TM reaches the final state, the input string is accepted, otherwise rejected.
- In the context of automata theory and the theory of computation, Turing machines are used to study the properties of algorithms and to determine what problems can and cannot be solved by computers.
- They provide a way to model the behavior of algorithms and to analyze their computational complexity, which is the amount of time and memory they require to solve a problem.
- A TM can be formally described as a 7-tuple (Q, X, ∑, δ, q0, B, F)

Definition of a Turing Machine

A TM can be formally described as a 7-tuple (Q, X, ∑, δ, q_0, B, F) where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- **∑** is the input alphabet
- **δ** is a transition function; $\delta : Q \times X \rightarrow Q \times X \times$ {Left_shift, Right_shift}.
- **q_0** is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Tape and Head

- The tape is infinite in one or both directions
- The head moves left (L) or right (R)

- Can read, write, and move

TM vs Finite Automata

Feature	Finite Automaton	Turing Machine
Memory	Finite	Infinite Tape
Rewrite Tape	No	Yes
Move Head	Right only	Left & Right
Recognizes	Regular Langs	RE Languages

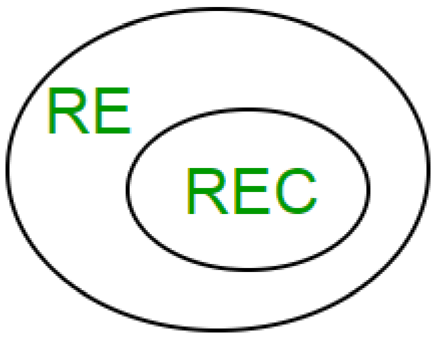
4.2 Turing-Recognizable and Turing-Decidable Languages

Recursively Enumerable languages

- If any Turing Machine can be designed to accept all string of the given language, then the language is called recursively enumerable language.
- Recursively enumerable languages are the formal languages that can be decide-able, (fully or partially).
- According to the Chomsky hierarchy of formal languages, we can see the recursively enumerable languages as type 0 languages.
- An RE language can be accepted or recognized by Turing machine which means it will enter into final state for the strings of language and may or may not enter into rejecting state for the strings which are not part of the language.
- It means TM can loop forever for the strings which are not a part of the language. RE languages are also called as Turing recognizable languages.

Recursive Language (REC)

- A recursive language (subset of RE) can be decided by Turing machine which means it will enter into final state for the strings of language and rejecting state for the strings which are not part of the language.
- e.g.; L= $\{a^nb^nc^n|n>=1\}$ is recursive because we can construct a turing machine which will move to final state if the string is of the form $a^nb^nc^n$ else move to non-final state.
- So the TM will always halt in this case. REC languages are also called as Turing decidable languages.



Recognizable vs Decidable

Feature	Turing-Recognizable (RE)	Turing-Decidable (Recursive)
Halts on all inputs?	No	Yes
Accepts members?	Yes	Yes
Rejects non-members?	Not guaranteed	Yes
Example	Halting problem (RE)	Palindromes (Decidable)

Closure Properties: Turing-Decidable Languages

Closed under:

- Union
- Intersection
- Complement
- Concatenation
- Kleene star

Closure Properties: Turing-Recognizable Languages

Closed under:

- **Union**
- **Intersection**
- **Concatenation**
- **Kleene star**

  ✗ **Not closed under: Complement**

4.3 Variants of Turing Machines & Nondeterminism

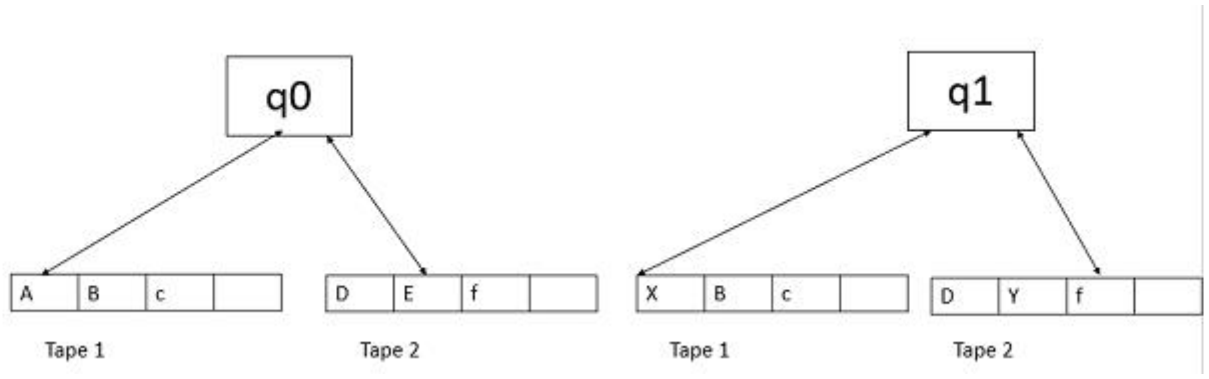
Variations of Turing Machine

- Turing machines are powerful computational models that can simulate any algorithmic process.
- A standard Turing machine consists of a single tape and a single read-write head. However, there are variations of Turing machine that have been developed to address different computational challenges.

- These variations differ mainly in structure and operation, but they all have the same computational power as the standard Turing machine.

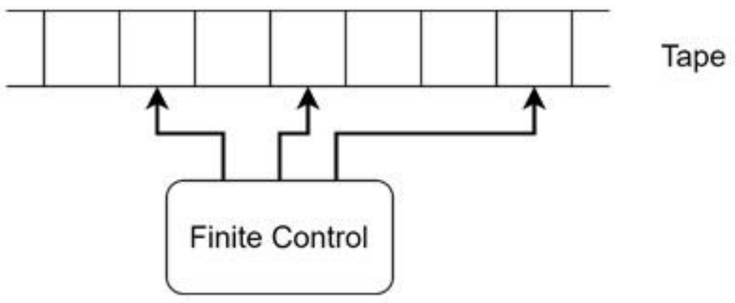
Multi-tape Turing Machine

- As the name suggests, a multi-tape Turing machine is an extension of the standard Turing machine where multiple tapes are available for input, output, and computation.
- Each tape has its own read-write head, and the machine's transition function is based on the current state and the symbols read by each head.



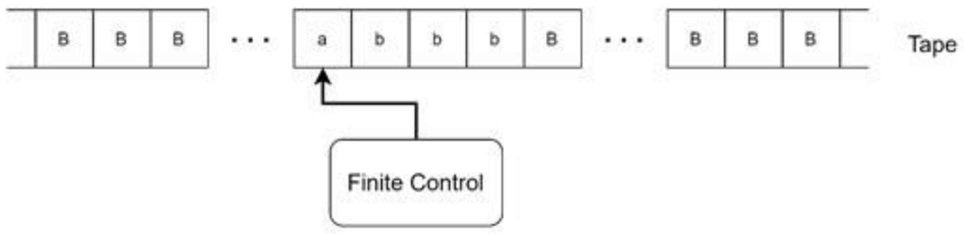
Multi-head Turing Machine

- In a multi-head Turing machine, a single tape is used, but it has multiple read-write heads.
- These heads can independently read and write symbols, enabling the machine to perform complex tasks more efficiently.



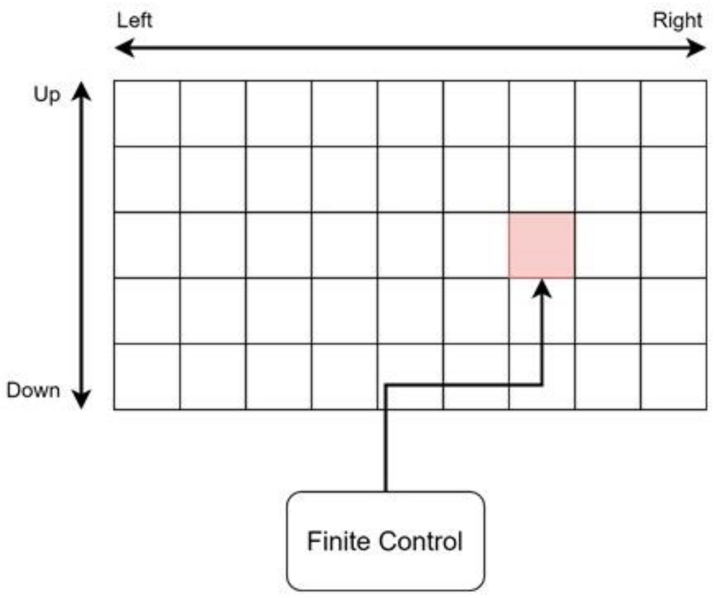
Two-way Infinite Tape Turing Machine

- A two-way infinite tape Turing machine allows the tape to extend infinitely in both directions, unlike the standard machine where the tape extends infinitely in only one direction.
- This removes the boundary on the left side of the tape.



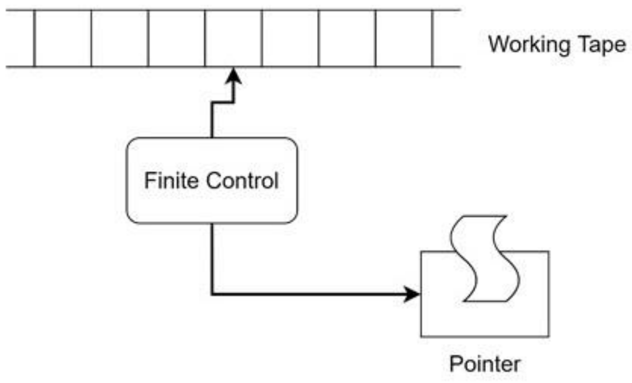
K-dimensional Turing Machine

- A K-dimensional Turing machine extends the concept of the tape to multiple dimensions.
- For example, a two-dimensional Turing machine as given in the following diagram, has a tape that extends infinitely in both the X and Y directions.



Enumerator Turing Machine

- An enumerator Turing machine is designed to generate strings of a language. It is equipped with a work tape and an output tape.
- The machine writes symbols to the output tape, which is then printed.



Non-deterministic Turing Machine

- In a Non-Deterministic Turing Machine, for every state and symbol, there are a group of actions the TM can have.
- So, here the transitions are not deterministic. The computation of a non-deterministic Turing Machine is

a tree of configurations that can be reached from the start configuration.

- An input is accepted if there is at least one node of the tree which is an accept configuration, otherwise it is not accepted.
- If all branches of the computational tree halt on all inputs, the non-deterministic Turing Machine is called a Decider and if for some input, all branches are rejected, the input is also rejected.

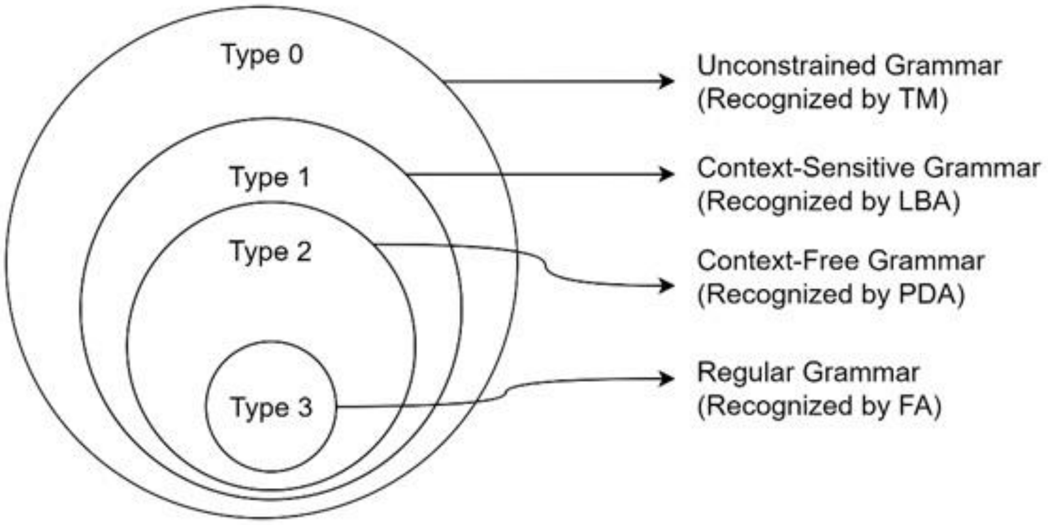
A non-deterministic Turing machine can be formally defined as a 6-tuple (Q, X, ∑, δ, q_0, B, F) where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- **∑** is the input alphabet
- **δ** is a transition function;
- δ : Q × X → P(Q × X × {Left_shift, Right_shift}).
- **q_0** is the initial state
- **B** is the blank symbol
- **F** is the set of final states

4.4 Unrestricted Grammars & Turing Machine Equivalence

Unrestricted Grammar

- Unrestricted grammar is a type of formal grammar that is defined without any restrictions on the form of its production rules.
- Formally, a grammar G = (V_N, Σ, P, S) is called an unrestricted grammar if all its productions are in the form LS → RS, where LS is a string of non-terminal and terminal symbols, and RS is a string of non-terminal and terminal symbols or the empty string.
- This form of grammar is known as Type 0 grammar in the Chomsky hierarchy, and it is the most general form of grammar.


Type 0
Type 1
Type 2
Type 3
Unconstrained Grammar (Recognized by TM)
Context-Sensitive Grammar (Recognized by LBA)
Context-Free Grammar (Recognized by PDA)
Regular Grammar (Recognized by FA)

- The lack of restrictions allows for a more flexible and powerful method of string generation, which leads to the capabilities of a Turing machine.
- Every language that can be generated by an unrestricted grammar can be recognized by a Turing machine, and vice versa.
- This states the idea that the set of languages generated by unrestricted grammar is equivalent to the set of recursively enumerable languages.

An **unrestricted grammar** (Type-0) is a 4-tuple:
G = (V, Σ, R, S) where:

- V: Variables (non-terminals)
- Σ: Terminals
- R: Rules of the form **α → β**, where:
    - α ∈ (V ∪ Σ)^+ (α ≠ ε)
    - β ∈ (V ∪ Σ)^*
- S: Start symbol

Equivalence of Unrestricted Grammars and Turing Machines

- **Theorem:** A language is generated by an unrestricted grammar if and only if it is recursively enumerable (i.e., it is semidecided by some Turing machine M).
- **Proof:**
- Only if (grammar → TM): by construction of a nondeterministic Turing machine.
- If (TM → grammar): by construction of a grammar that mimics backward computations of M.

Direction 1 – Grammar ⇒ TM

Given an unrestricted grammar **G**, we can construct a TM **M** such that:

- M simulates leftmost derivation of G on input string w
- If derivation leads to w, M accepts

🛠 **Method:**

- Encode derivations on the TM tape
- Simulate rule application step-by-step

Direction 2 – TM ⇒ Grammar

Given a TM M, we can construct an unrestricted grammar G such that:

- G generates all strings accepted by M

🛠 **Idea:**

- Simulate TM configurations as strings

- Use productions to mimic TM transitions

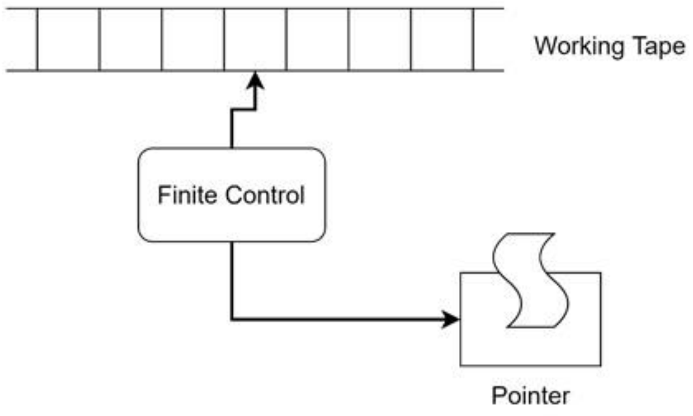
4.5 Turing Machines as Enumerators

What is an Enumerator?

- An enumerator is a Turing machine with an attached printer (output device)
- It prints (enumerates) strings of a language L possibly in infinite sequence
- Strings may appear in any order, possibly with repetition
- There are different variations of Turing Machines, which are quite powerful and useful in several cases.
- We have a variation of the Turing machine called the Enumerator, which plays a different but equally important role.
- In this type of machine, instead of simply determining whether a string is in a language, an enumerator generates or lists all the strings that belong to a language.

An **Enumerator** is similar to a Turing machine in structure but with a distinct purpose.

- Like a Turing machine, an enumerator has a tape that extends infinitely and a finite state control that guides its operations.
- The key difference is that an enumerator also has a **printer**.
- This printer allows the enumerator to produce strings, effectively generating a sequence of strings that make up a language.



Key Characteristics of Enumerator Turing Machine

- **Tape –** The tape of an enumerator is initially empty, and unlike a standard Turing machine, there is no input string provided to it.
- **Finite State Control –** The control unit of an enumerator functions similarly to that of a Turing machine, guiding the machine's operations based on its current state and the symbols on the tape.
- **Printer –** The printer is a unique feature of the enumerator, enabling it to output strings that belong to the language it is enumerating.
- **Operations –** The operation of an enumerator is straightforward. It begins with an empty tape and produces strings by writing them onto the tape and then printing them out. The enumerator lists all the strings in a language, effectively defining that language through enumeration.
- **Halting and Looping –** An enumerator can either halt after producing a certain number of strings or it may continue to loop indefinitely, generating more strings. For infinite languages an enumerator will run forever, printing out an endless list of strings.

Formal Definition

An enumerator is a Turing machine E that:

- Has no input
- Prints strings to an output tape
- The set of strings printed is L(E) — the language enumerated by E

References:

1. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation* (3rd ed.). Pearson Education. — Chapter 9: Introduction to Turing Machines (Sections on Turing Machines, Variants, and Unrestricted Grammars).
2. GeeksforGeeks. (n.d.). *Turing Machine*. Retrieved from https://www.geeksforgeeks.org/turing-machine/
3. TutorialsPoint. (n.d.). *Turing Machine*. Retrieved from https://www.tutorialspoint.com/automata_theory/turing_machine.htm

Parul® University
Vadodara, Gujarat
NAAC GRADE A++

https://paruluniversity.ac.in/