

C.A.T 2

ICS 3103 Automata Theory And Computability

1. **Turing Machines**: Introduction to Turing Machine, Formal Description, Instantaneous description, The language of a Turing machine

**Types of Turing machine:** Turing machines and halting

Introduction to Turing Machine

A **Turing Machine (TM)** is an abstract mathematical model introduced by Alan Turing in 1936. It is used to formalize the concept of **algorithmic computation**. Turing Machines help define what problems can or cannot be solved using algorithms, making them a foundational concept in **computability theory** and **automata theory**. They are more powerful than finite automata and pushdown automata because they can simulate any computation that a real computer can perform.

Formal Description

A Turing Machine is defined as a 7-tuple:  
**M = (Q, Σ, Γ, δ, q₀, q-accept, q-reject)**

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Q** | **Finite set of states** |
| **Σ (Sigma)** | **Input alphabet (does not include the blank symbol □)** |
| **Γ (Gamma)** | **Tape alphabet (includes □ and all symbols in Σ)** |
| **δ (delta)** | **Transition function: Q × Γ → Q × Γ × {L, R}** |
| **q₀** | **Start state (q₀ ∈ Q)** |
| **q-accept** | **Accepting state (q\_accept ∈ Q)** |
| **q-reject** | **Rejecting state (q\_reject ∈ Q, and q\_accept ≠ q\_reject)** |
| **Component** | **Description** |
| **Q** | **Finite set of states** |
| **Σ (Sigma)** | **Input alphabet (does not include the blank symbol □)** |
| **Γ (Gamma)** | **Tape alphabet (includes □ and all symbols in Σ)** |
| **δ (delta)** | **Transition function: Q × Γ → Q × Γ × {L, R}** |
| **q₀** | **Start state (q₀ ∈ Q)** |
| **q-accept** | **Accepting state (q\_accept ∈ Q)** |
| **q-reject** | **Rejecting state (q\_reject ∈ Q, and q\_accept ≠ q\_reject)** |

**Simple Example:**

**Let M accept strings of the form a^n b^n (We want a Turing Machine that accepts strings like:ab, aabb, aaabbb – where the number of as is the same as the number of bs, and all as come before the bs.)**

**What is the Machine Doing?**

**The Turing Machine has this:**

* A tape (like a very long piece of paper with symbols).
* A head that reads and writes symbols on the tape and can move left or right.
* A set of rules (called states) that tell it what to do depending on what it reads.
* **Q = {q₀, q₁, q₂, q-accept, q-reject}**
* **Σ = {a, b}**
* **Γ = {a, b, X, Y, □}-**X and Y are **markers** the machine uses to mark which as and bs have already been checked and □ is a **blank symbol**, used at the end of the input.
* **q₀ = q₀**
* **δ includes:**

**δ(q₀, a) = (q₁, X, R)**

1. If you're in state q₀ and see an a, **change it to X** to mark it as used.
2. **Move right** to find a b.
3. Go to state q₁.

**δ(q₁, b) = (q₂, Y, L)**

1. n state q₁, if you see a b, **change it to Y** to mark it.
2. **Move left** to go back.
3. Go to state q₂.

The Turing Machine:

* Matches each a with a b
* Uses X and Y to keep track of matched characters
* Moves back and forth on the tape
* Accepts if all a’s and b’s are correctly paired

**Instantaneous Description (ID)**

An **Instantaneous Description (ID)** shows the **current state**, **tape content**, and the **position of the head** at a particular moment.

**Notation:**  
u q v

* u = left of the head
* q = current state
* v = symbol under the head + rest of the tape

**Example:**  
Tape: a a b □ □ ...  
Current state: q₀, head is on first b  
ID: aa q₀ b□□

This helps in tracing the step-by-step computation of the Turing Machine.

**Language of a Turing Machine**

The **language accepted** by a Turing Machine is the set of input strings that lead it to the **accepting state**.

**Key Points:**

* A TM accepts input w if it eventually enters q\_accept when started on w.
* The language accepted is **recursively enumerable** (RE).
* Some TMs **halt** on accepted strings, but may loop forever on others.

**Example:**  
Let L = {aⁿbⁿ | n ≥ 1}. A TM for L:

* Replaces an a with X, finds the corresponding b, replaces with Y, repeats.
* Accepts if all a’s and b’s match and no extra characters remain.

**Types of Turing Machines**

| **Type** | **Description** | **Example Use Case** |
| --- | --- | --- |
| **Standard TM** | Single tape, deterministic transitions. | Basic computability proofs. |
| **Multi-tape TM** | Multiple tapes, head moves independently. | Simulating real-world computers. |
| **Non-deterministic TM** | Multiple possible transitions per step. | Solving NP problems theoretically. |
| **Universal TM** | Simulates other TMs given their description. | Basis for modern computers. |
| **Alternating TM** | Combines non-determinism with universal states. | Complexity theory (e.g., PSPACE). |

**Turing Machines and Halting**

**The halting problem asks whether a Turing Machine will halt on a given input.**

**Key Facts:**

* **Proven undecidable by Alan Turing.**
* **No general algorithm can decide for every machine and input whether it halts.**
* **This limits our ability to fully automate program correctness checking.**

**Example of Halting Problem:  
Input: TM M and string w  
Problem: Will M halt when given w?  
Answer: No Turing Machine can solve this for all inputs.**

**Turing Machines are essential in understanding computation. They help define:**

* **What can be computed (computability)**
* **How problems are classified (decidability and recognizability)**
* **Theoretical limitations of algorithms (halting problem)**