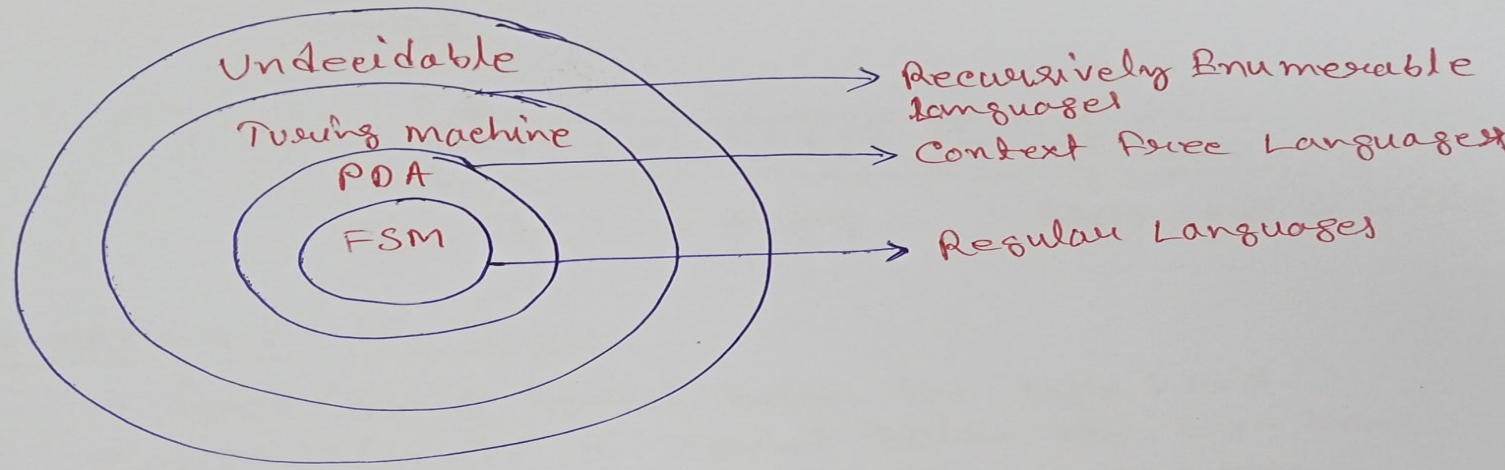
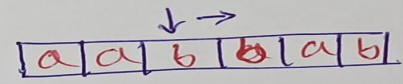
Turing Machine

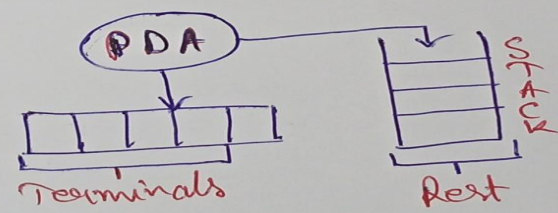
Turing Machine - Introduction



FSM: → The Input String

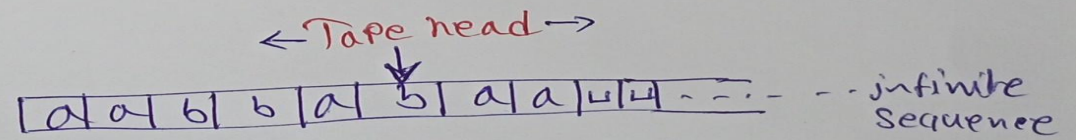


PDA: → The Input String
→ A Stack



TURING MACHINE:

→ A Tape



Tape Alphabets: $\Sigma = \{0, 1, a, b, x, z_0\}$

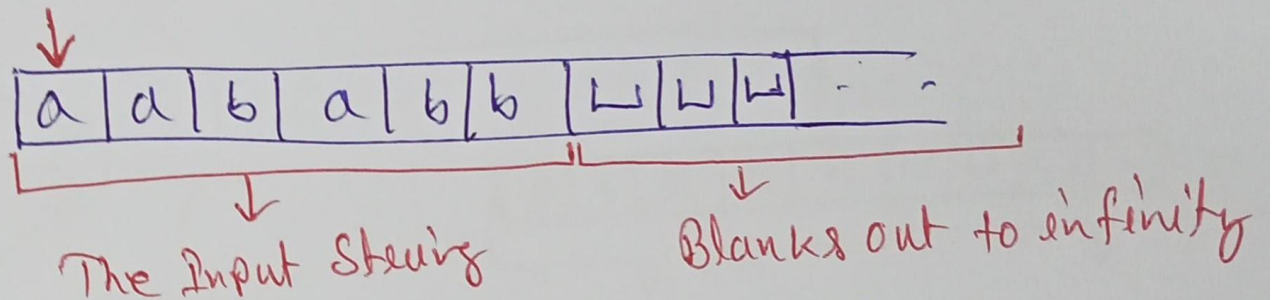
The Blank ⊔ is a special symbol, where $\sqcup \notin \Sigma$

↓

It is a special symbol used to fill the infinite tape.

TM - Introduction

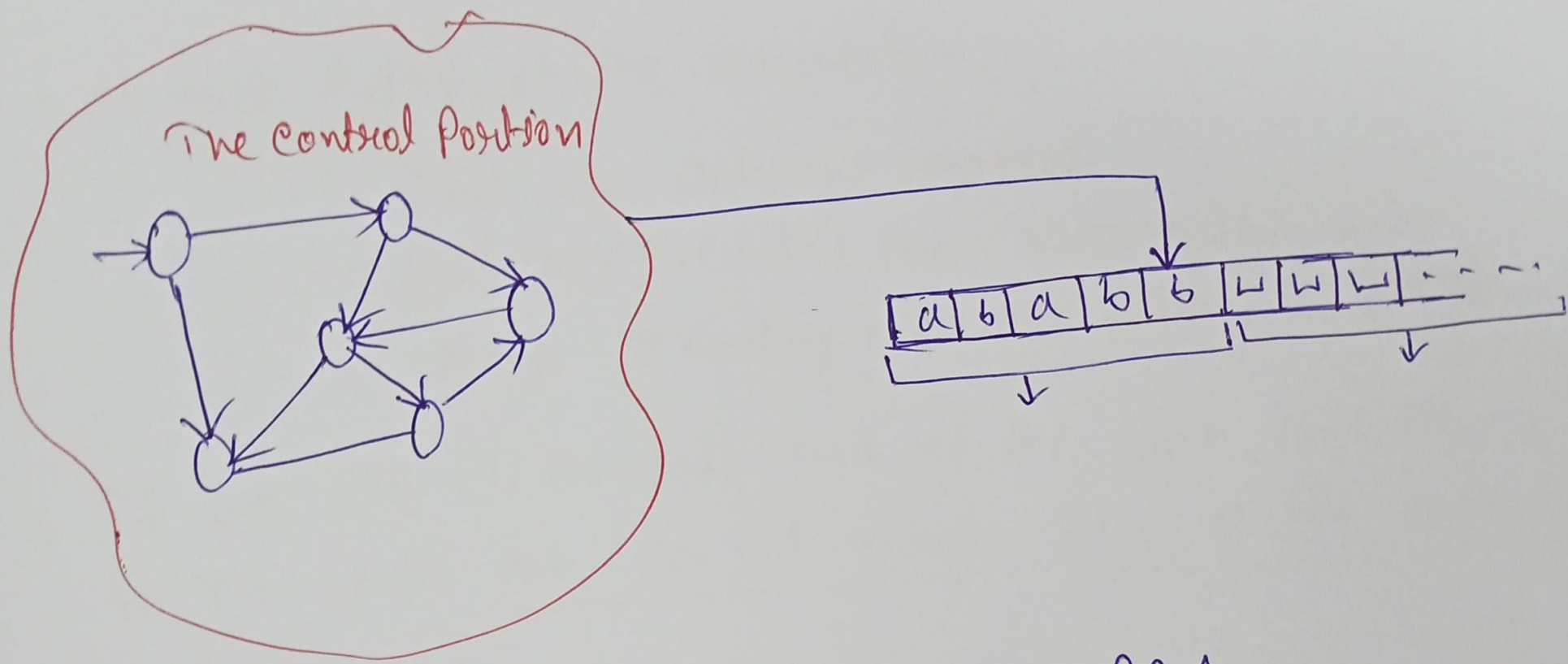
Initial Configuration:



Operations on the Tape:

- → Read/Scan symbol below the tape head.
- → Update/Write a symbol below the tape head.
- → Move the tape head one step LEFT
- → Move the tape head one step Right.

TM - Introduction



The control portion similar to FSM or PDA

The program

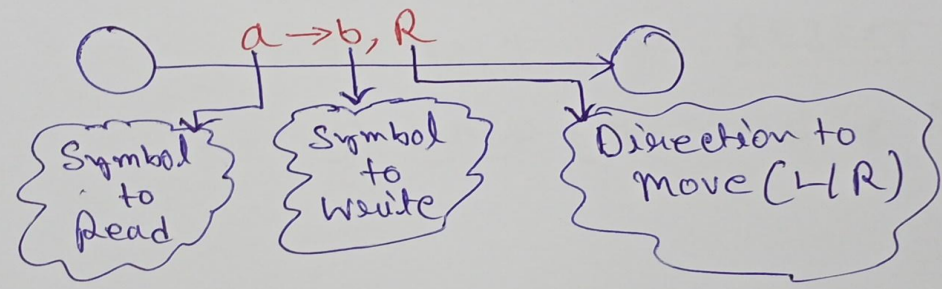
It is deterministic

Rules of Operation - 1 (TM)

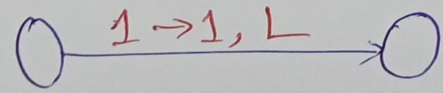
⇒ At each step of the computation:

- → Read the current symbol
- → Update (i.e. write) the same cell
- → move exactly one cell either L or R

⇒ If we are at the left end of the tape, and trying to move left, the do not move. Stay at the left end.



⇒ If you don't want to update the cell, just write the same symbol,



Rules of operation - 2 (TM)

→ Control is with a sort of FSM

→ Initial State

→ Final States: (there are two final states)

1) The ACCEPT state
2) The REJECT state

→ Computation can either

1) HALT and ACCEPT
2) HALT and REJECT
3) LOOP (the machine fails to HALT)

Turing machine (Formal Definition)

⇒ A Turing machine (TM) can be defined as a set of 7 tuples

$$(Q, \Sigma, \Gamma, \delta, q_0, b, F)$$

Q → Non empty set of States

Σ → Non empty set of Symbols

Γ → Non empty set of Tape symbols

δ → Transition function defined as

$$Q \times \Sigma \rightarrow \Gamma \times (R/L) \times Q$$

q_0 → Initial State

b → Blank Symbol

F → Set of final states (ACCEPT & REJECT States)

⇒ Thus, the production rule of TM will be written as

$$\delta(q_0, a) \rightarrow (q_1, Y, R)$$

Turing Thesis:

⇒ It states that any computation that can be carried out by mechanical means can be performed by some Turing Machine.

Few arguments for accepting this thesis are:

1) Anything that can be done on existing digital computer can also be done by TM.

2) No one has ~~yet~~ yet been able to suggest a problem solvable by what we consider an algorithm, for which a TM program cannot be written.

Recursively Enumerable Language:

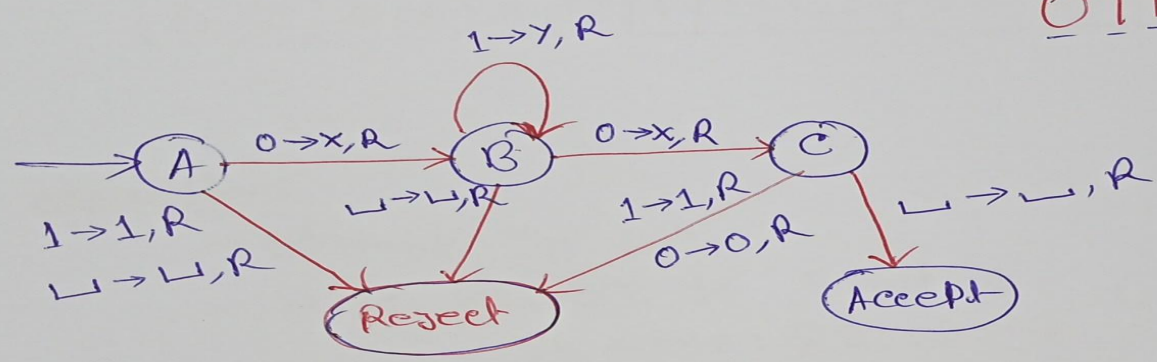
A Language L and Σ is said to be Recursively Enumerable if there exists a TM that accepts it.

Turing Machine - Example

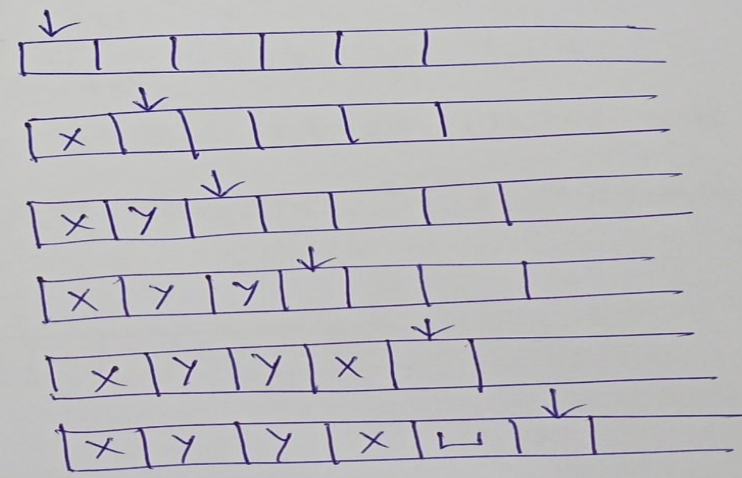
⇒ Design a TM which recognizes the language

$$L = 01^*0$$

0 1 1 0 ␣



Tape →



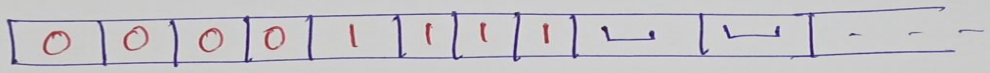
⇒ Deterministic TM [$\Sigma = \{0, 1\}$, $b = ␣$]

⇒ TM is more powerful than PSM and PDA

Turing Machine - Example

Design a TM which recognises the language

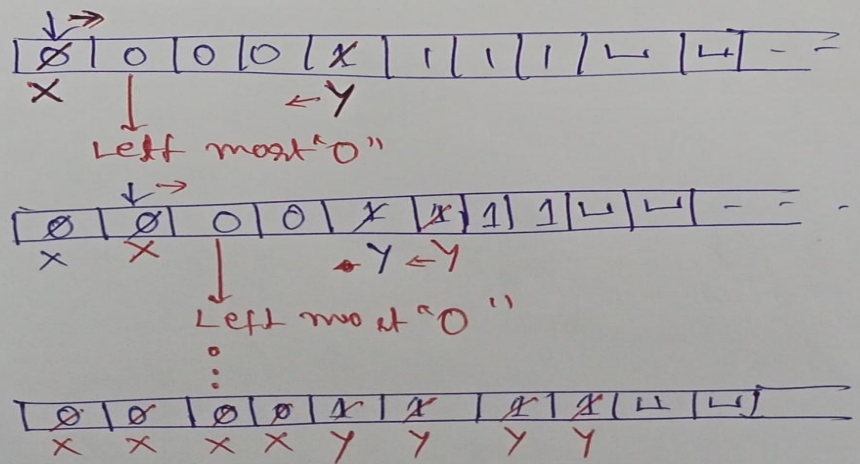
$L = 0^N 1^N$



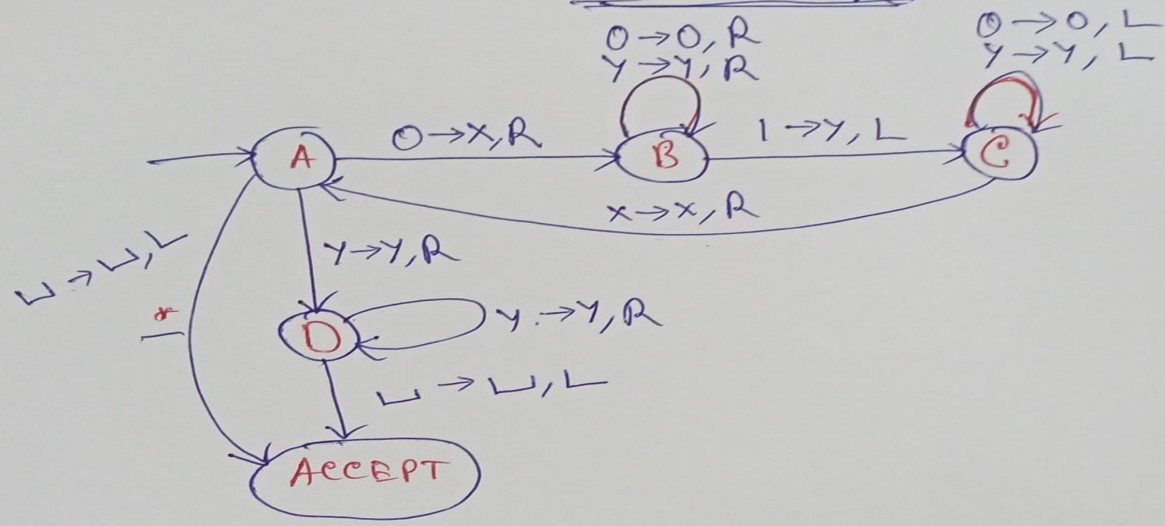
Algorithm:

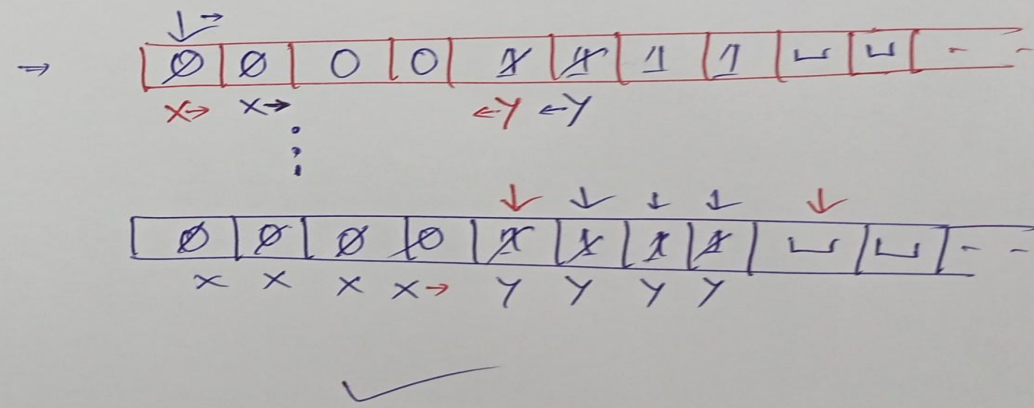
→ Change "0" to "x"
→ move RIGHT to first "1"
If none: REJECT

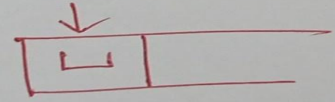
→ Change "1" to "y"
→ move LEFT to Leftmost "0"
→ Repeat the above steps until no more "0"s
→ make sure no more "1"s remain.



TM - Example

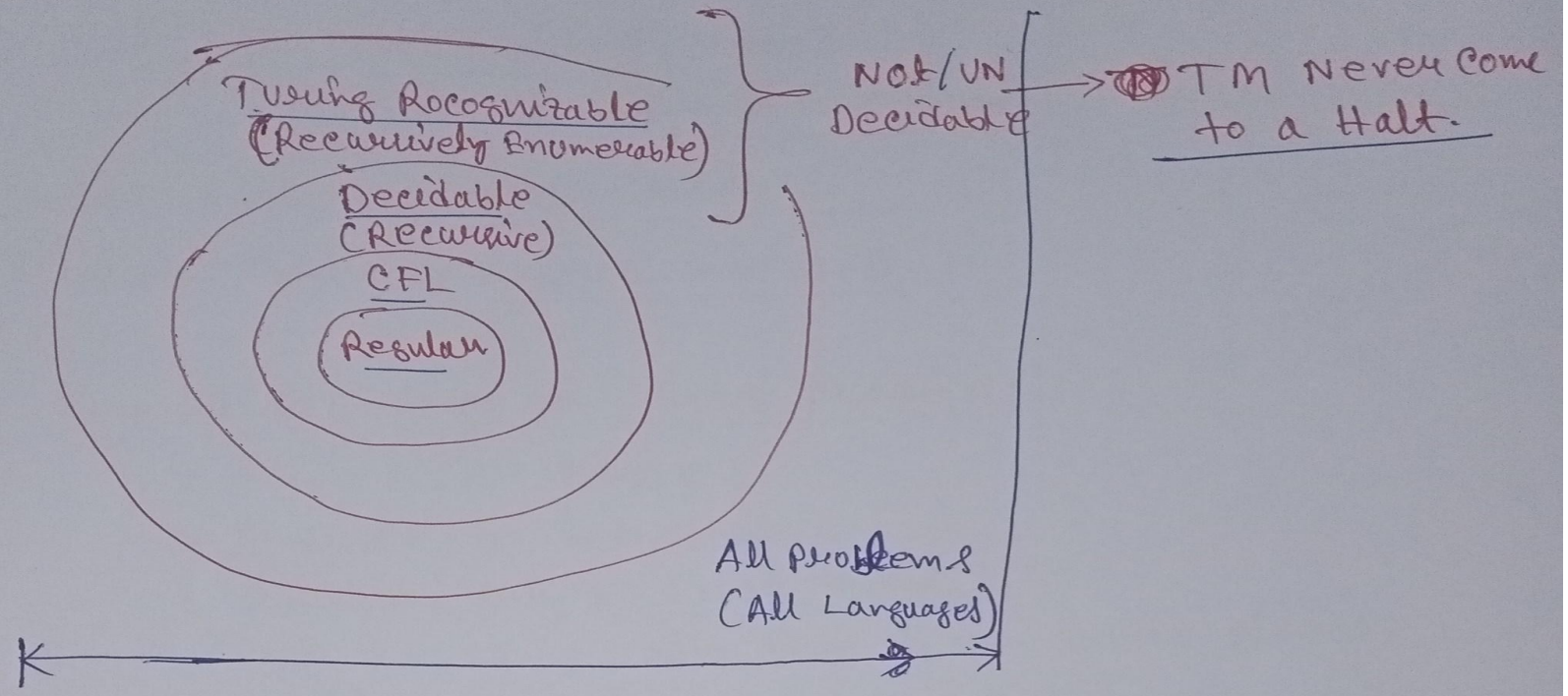




* 

NOTE! Any edge that has a missing transition that will go to the REJECT State

Different Classes of Languages



Turing machine Languages:

⇒ Decidable (Recursive) → that can be decided by a TM.

① Accept and Halt.
② Reject and Halt.

→ Always come to a Halt.

⇒ Turing Recognizable → (Recursively Enumerable)

① Accept and Halt
② Loop (may not Halt)

The CHURCH-TURING Thesis

⇒ What does COMPUTABLE mean?

→ FUZZY Term

① Alonzo Church - LAMDA CALCULUS

② Allen Turing - TURING MACHINE

Several Variations of Turing Machine:

- → One Tape or many
- → Infinite on both ends
- → Alphabets only {0, 1} or more?
- → Can the Head also stay in the same place?
- → Allow Non-Determinism

All variations are equivalent in Computing Capability

Turing machine and Lambda calculus are also equivalent in power

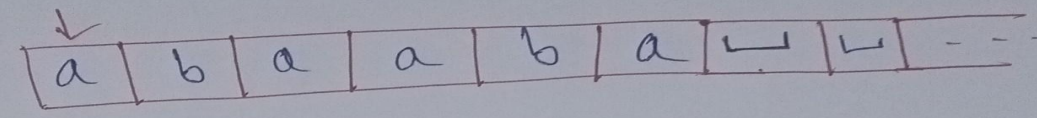
Algorithmically Computable means Computable by Turing machine

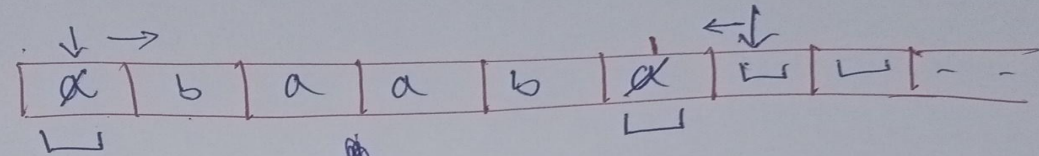
NOTE: TURING TEST ≠ TURING MACHINE

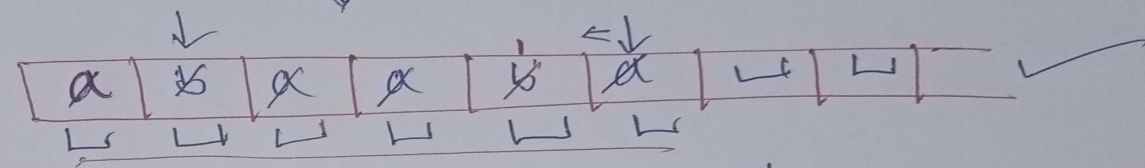
Turing machine for Even Palindromes

⇒ Design a TM that accepts Even palindromes over the alphabet $\Sigma = \{a, b\}$

Ex. ①







②