Pumping Lemma

Pumping lemma and its application.

- Suppose L is an infinite language. Then there is an integer 'n'(pumping length) so that for any w∈ L with |w|>=n, then w can be divided into three parts w=xyz so that
  1. For each i>=0, $xy^iz \in L$
  2. |xy|<=n and
  3. |y|>0

Pumping lemma ‘s application

Application

- The pumping lemma is extremely useful in proving that certain sets are non-regular. The general methodology followed during its applications is
  - Select a string w in the language L.
  - Break the string w into x, y and z in accordance with the above conditions imposed by the pumping lemma.
  - Now check if there is any contradiction to the pumping lemma for any value of i.

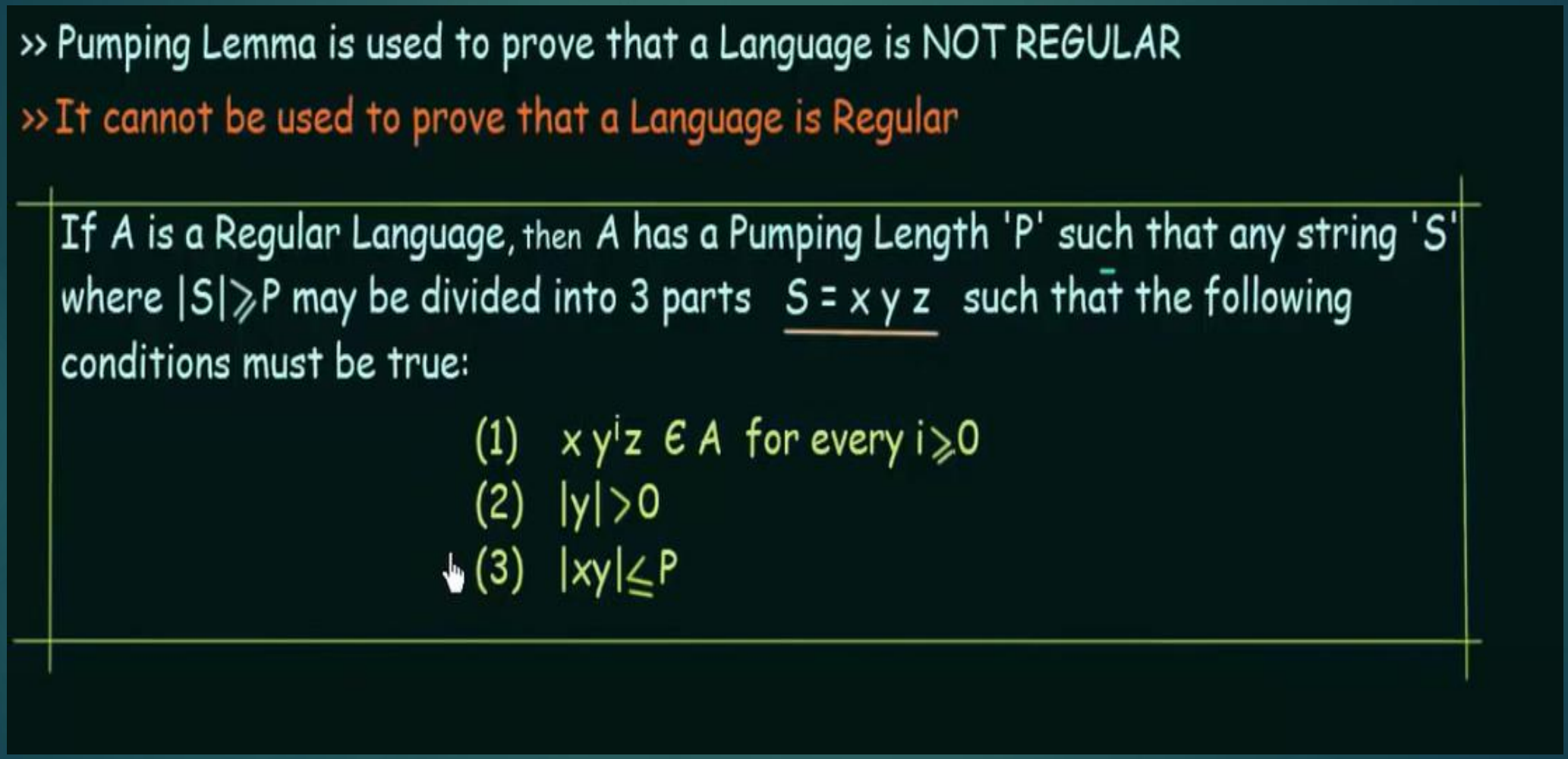
Example

- $a^n b^{2n}$ is regular or not?
- Say n=2
- W=aabbbb
- Divide w into 3 parts
- aa bb bb
- x=aa
- y=bb
- z=bb

Example

- xy^iz
- $aa(bb)^1bb$
- i=i+1
- i=2
- $(aa)(bb)^2(bb)$
- New w=aa bb bb bb
- This new W does not belong to L because number(a)=2 and number(b)=4 here number(b)=6

Pumping Lemma For Regular Languages:



Pumping Lemma For Regular Languages:

To prove that a language is not Regular using PUMPING LEMMA, follow the below steps:

(We prove using Contradiction)

- -> Assume that A is Regular
- -> It has to have a Pumping Length (say P)
- -> All strings longer than P can be pumped $|S| \geq P$
- -> Now find a string 'S' in A such that $|S| \geq P$
- -> Divide S into x y z
- -> Show that $x y^i z \notin A$ for some i
- -> Then consider all ways that S can be divided into x y z
- -> Show that none of these can satisfy all the 3 pumping conditions at the same time
- -> S cannot be Pumped == CONTRADICTION

Pumping Lemma For Regular Languages:

Using Pumping Lemma prove that the language $A = \{a^n b^n \mid n \geqslant 0\}$ is Not Regular

Pumping Lemma For Context Free Languages:

Pumping Lemma (for CFL) is used to prove that a language is NOT Context Free

Context Free Language

In formal language theory, a Context Free Language is a language generated by some Context Free Grammar.

The set of all CFL is identical to the set of languages accepted by Pushdown Automata.

Context Free Grammar is defined by 4 tuples as $G = \{ V, \Sigma, S, P \}$ where

V = Set of Variables or Non-Terminal Symbols

Σ = Set of Terminal Symbols

S = Start Symbol

P = Production Rule

Context Free Grammar has Production Rule of the form

$$A \rightarrow \alpha$$

where, $\alpha = \{V \cup \Sigma\}^*$ and $A \in V$

Pumping Lemma For Context Free Languages:

Pumping Lemma (for CFL) is used to prove that a language is NOT Context Free

If A is a Context Free Language, then, A has a Pumping Length 'P' such that any string 'S', where $|S| \geq P$ may be divided into 5 pieces $S = uvxyz$ such that the following conditions must be true:

(1) $u\,v^i x\,y^i z$ is in A for every $i \geq 0$

(2) $|v\,y| > 0$

(3) $|v\,x\,y| \leq P$

Pumping Lemma For Context Free Languages:

To prove that a Language is Not Context Free using Pumping Lemma (for CFL) follow the steps given below: (We prove using CONTRADICTION)

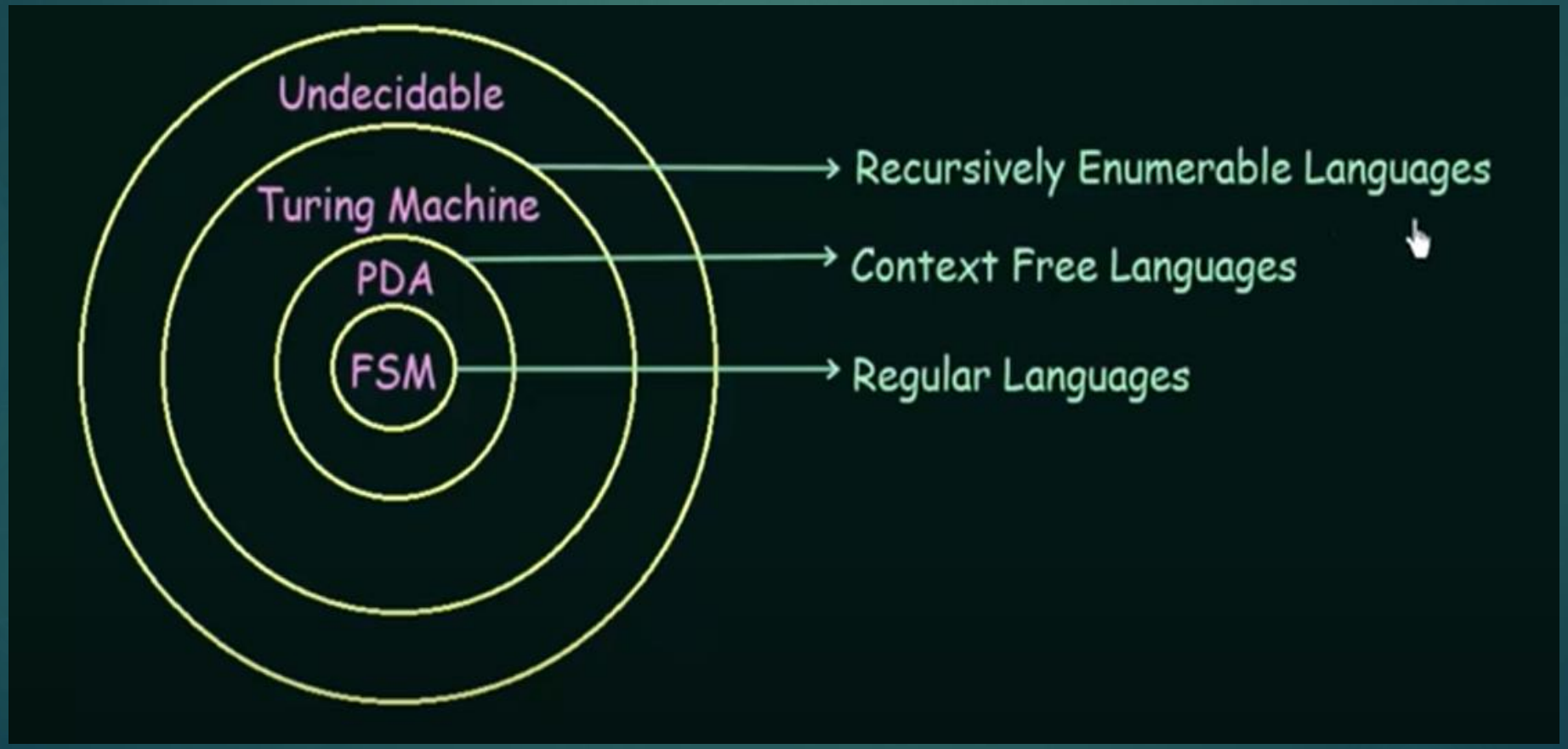
-> Assume that A is Context Free
-> It has to have a Pumping Length (say P)
-> All strings longer than P can be pumped $|S| \geqslant P$
-> Now find a string 'S' in A such that $|S| \geqslant P$
-> Divide S into uvxyz
-> Show that $u\, v^i x\, y^i z \notin A$ for some i
-> Then consider the ways that S can be divided into uvxyz
-> Show that none of these can satisfy all the 3 pumping conditions at the same time
-> S cannot be pumped == CONTRADICTION

Pumping Lemma For Context Free Languages:

- Examples need to be consider from the textbook or online resources.

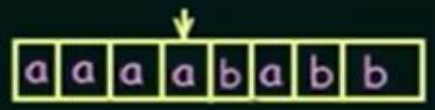
Turing Machine

Introductions:

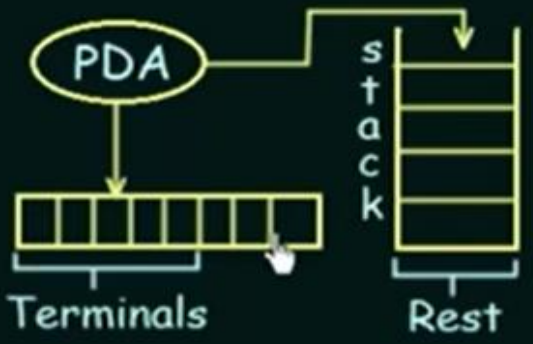


Introductions:

FSM: The Input String

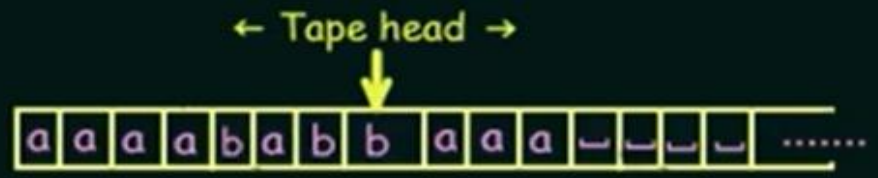


PDA: -> The Input String
-> A Stack



TURING MACHINE:

-> A Tape



Tape Alphabets: $\Sigma = \{ 0,1,a,b,x,Z_0 \}$

The Blank ⌴ is a special symbol. $\text{⌴} \notin \Sigma$

The blank is a special symbol used the fill the infinite tape

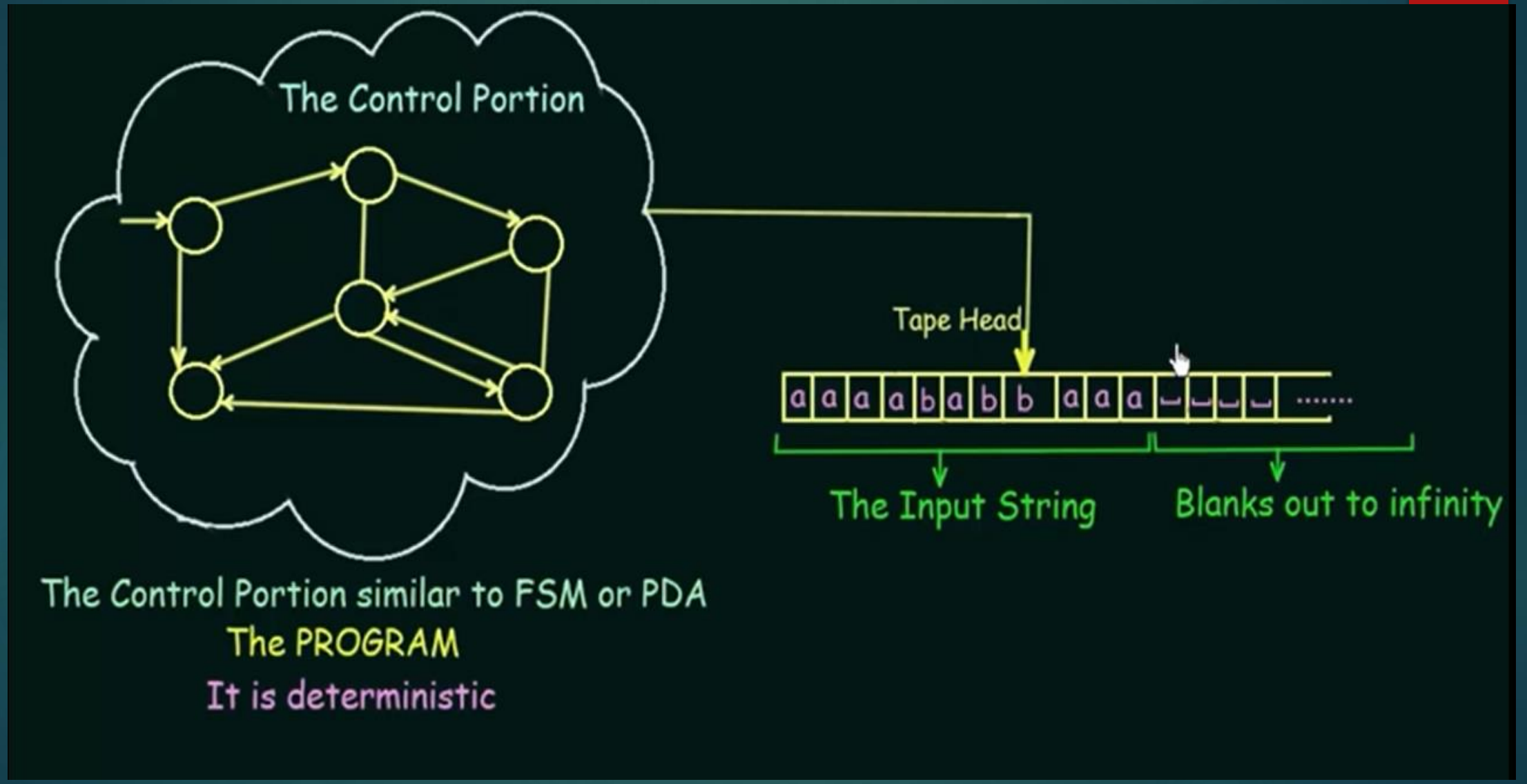
Introductions:

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

Introductions:



Introductions:

Rules of Operation - 1

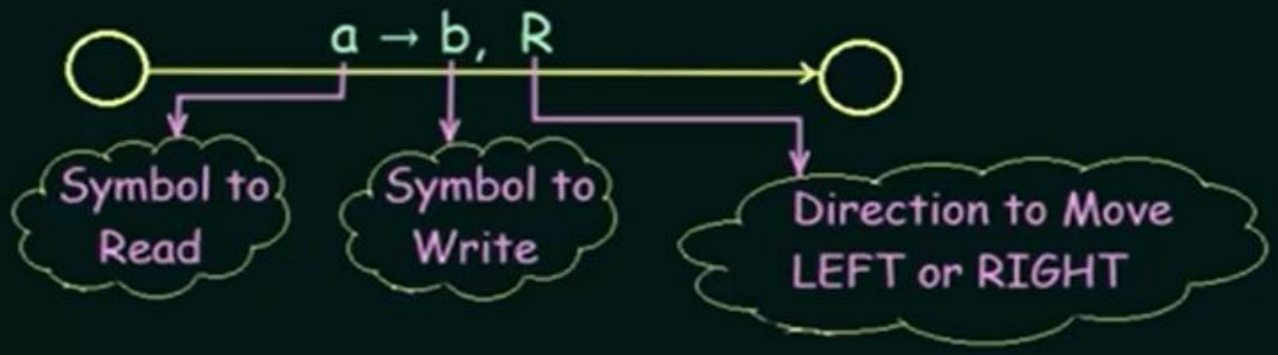
At each step of the computation:

-> Read the currect symbol

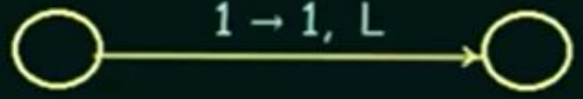
-> Update (i.e. write) the same cell

-> Move exactly one cell either LEFT or RIGHT

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



If you dont want to update the cell,
JUST WRITE THE SAME SYMBOL



Introductions:

Rules of Operation - 2

-> 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 Hypothesis:

Turing's Thesis:

Turing's Thesis 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:

i. Anything that can be done on existing digital computer can also be done by Turing Machine.

ii. No one has yet been able to suggest a problem solvable by what we consider on algorithm, for which a Turing Machine Program cannot be written.

Church Turing Hypothesis:

What does COMPUTABLE mean?

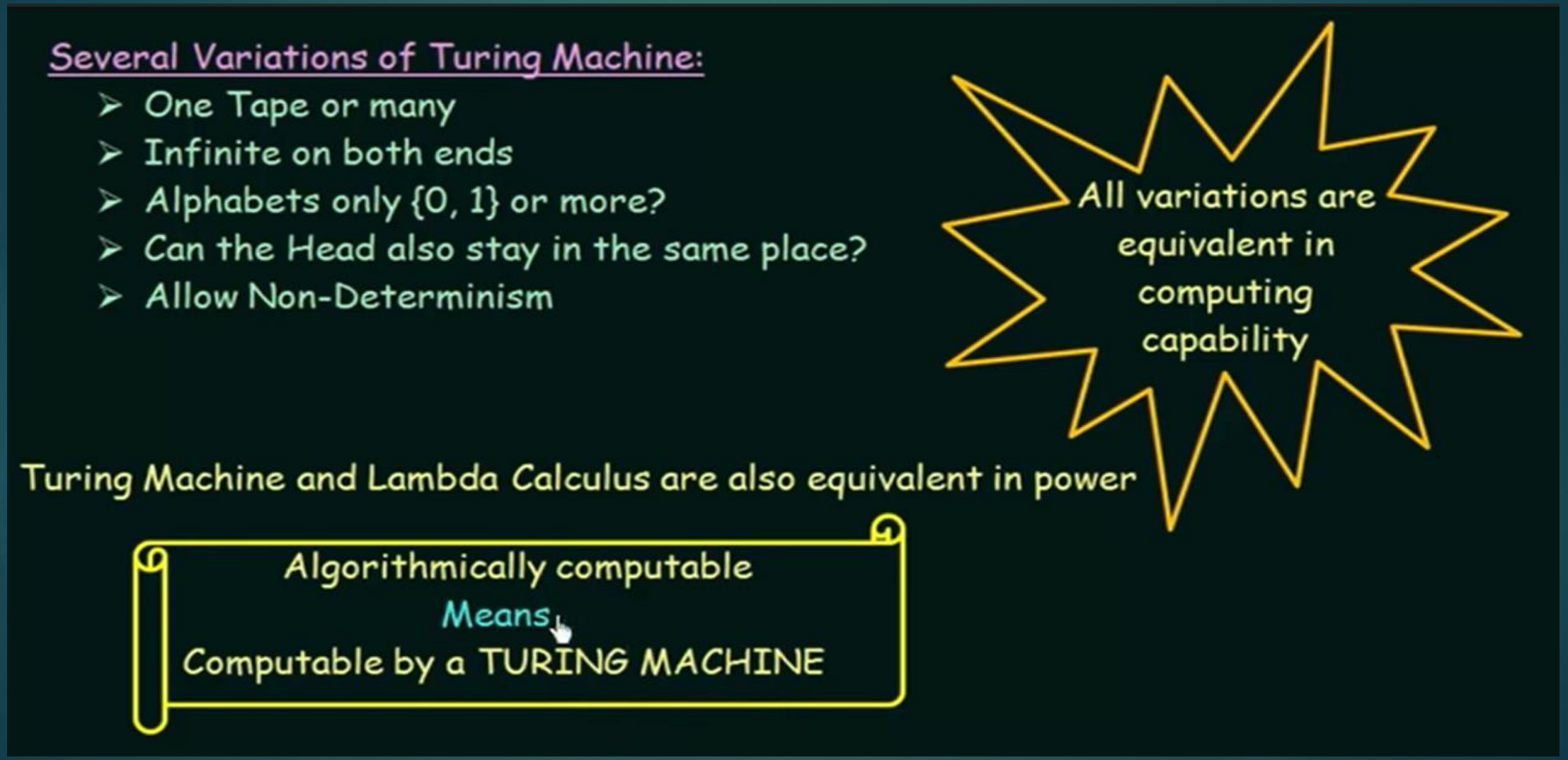
Alonzo Church - LAMBDA CALCULUS

(June 14, 1903 - August 11, 1995)
American mathematician and logician who made major contributions to mathematical logic and the foundations of theoretical computer science

Allen Turing - TURING MACHINE

(23 June 1912 - 7 June 1954)
English computer scientist, mathematician, logician, cryptanalyst, philosopher and theoretical biologist.

Church Turing Hypothesis:



Turing Machine Formal Definition:

A Turing Machine can be defined as a set of 7 tuples

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

$Q \rightarrow$ Non empty set of States

$\Sigma \rightarrow$ Non empty set of Symbols

$\Gamma \rightarrow$ Non empty set of Tape Symbols

$\delta \rightarrow$ Transition function defined as

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

$q_0 \rightarrow$ Initial State

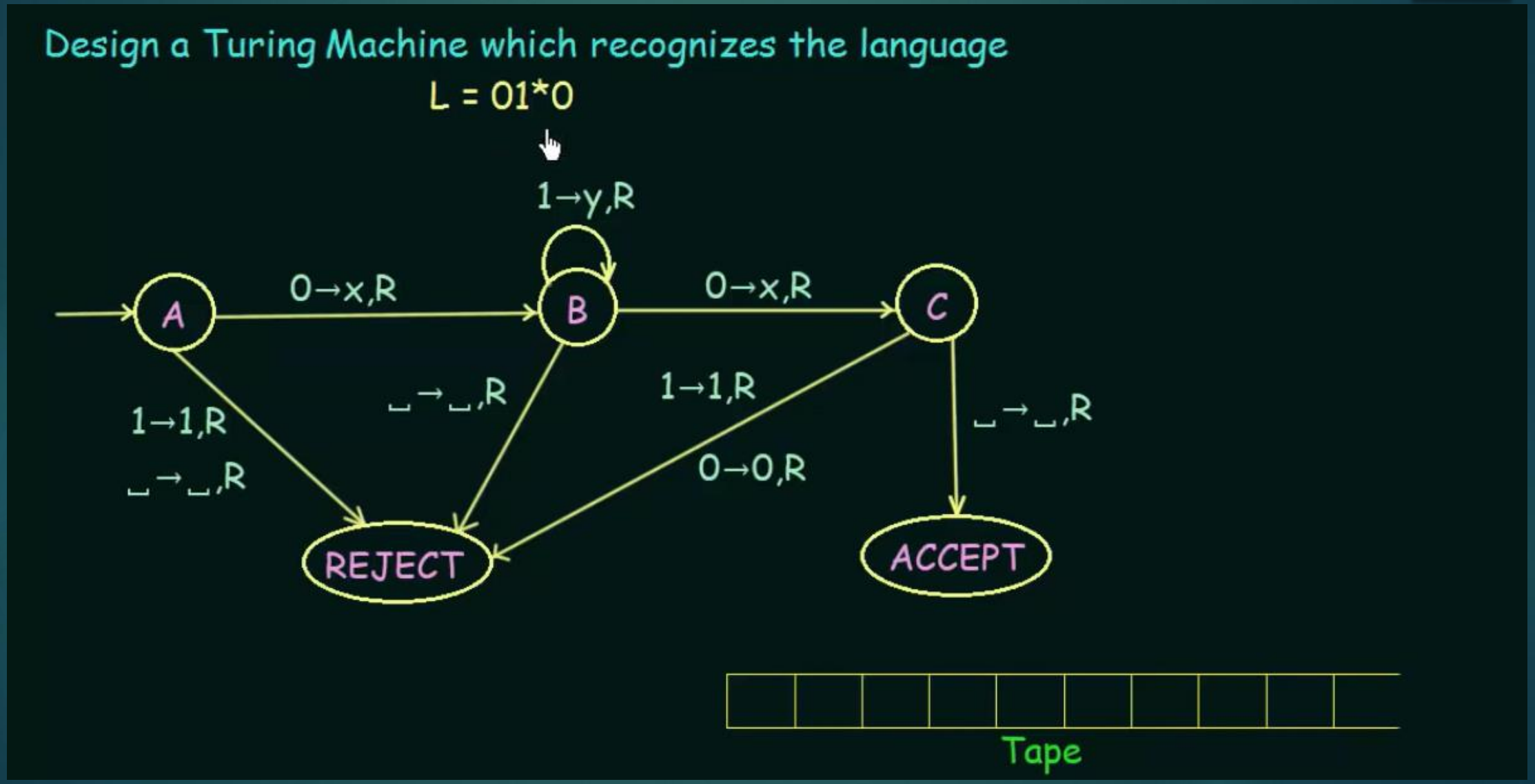
$b \rightarrow$ Blank Symbol

$F \rightarrow$ Set of Final states (Accept state & Reject State)

Thus, the Production rule of Turing Machine will be written as

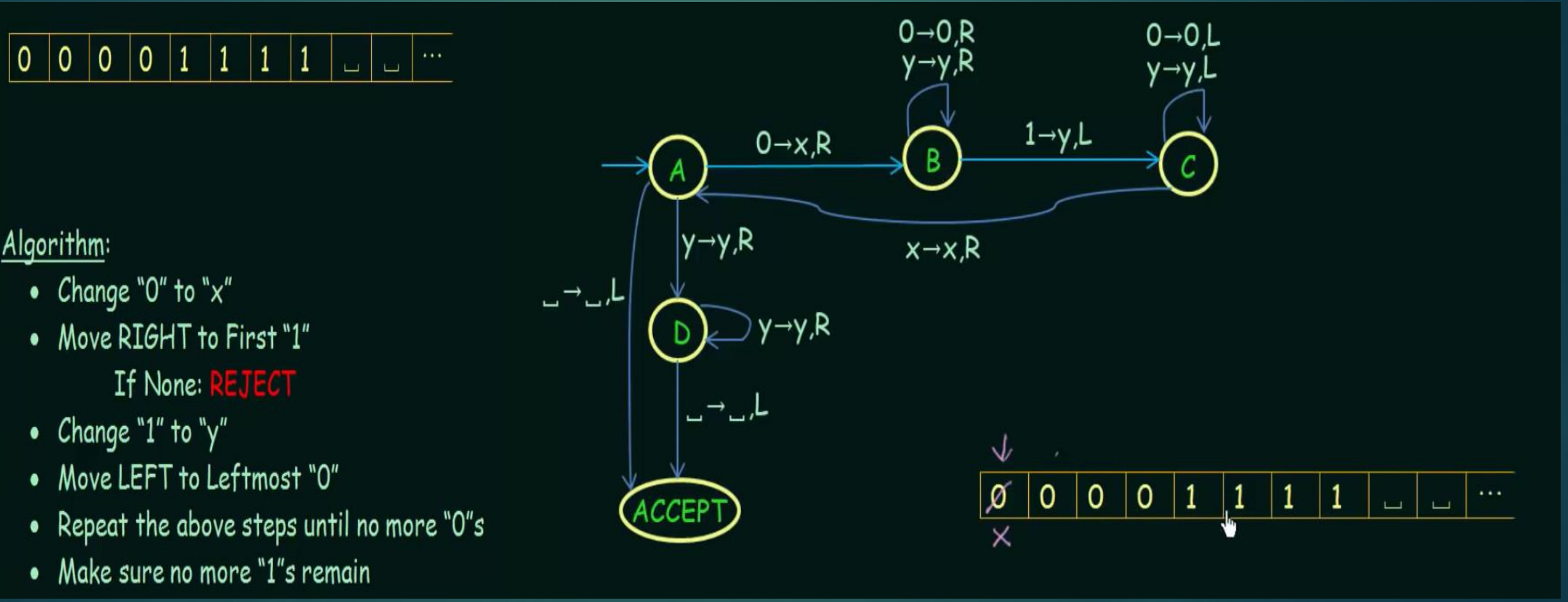
$$\delta (q_0, a) \rightarrow (q_1, y, R)$$

Example:


Design a Turing Machine which recognizes the language
L = 01*0
1→y,R
0→x,R
0→x,R
A
B
C
1→1,R
␣→␣,R
␣→␣,R
1→1,R
0→0,R
REJECT
ACCEPT
Tape

Example:

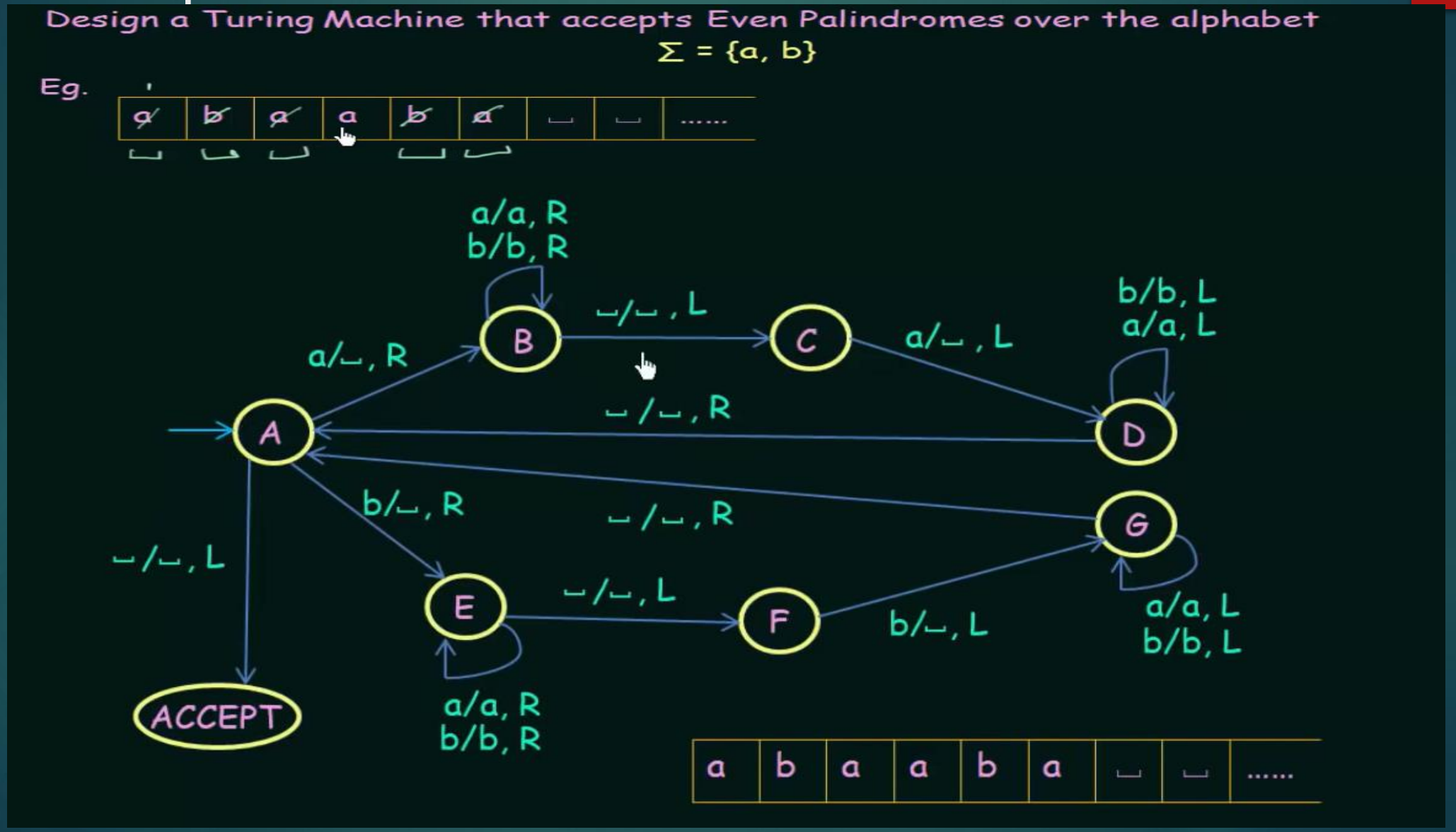
Design a Turing Machine which recognizes the language $L = 0^N 1^N$



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

Example:


Design a Turing Machine that accepts Even Palindromes over the alphabet
Σ = {a, b}
Eg.
a b a a b a ␣ ␣ .....
a/a, R
b/b, R
␣/␣, L
b/b, L
a/a, L
a/␣, R
B
C
a/␣, L
␣/␣, R
A
D
b/␣, R
␣/␣, R
G
␣/␣, L
E
␣/␣, L
F
b/␣, L
a/a, L
b/b, L
ACCEPT
a/a, R
b/b, R
a b a a b a ␣ ␣ .....

Thank You.