# CS & IT ENGINEERING

## Theory of Computation

Turing Machine Recursively Enumerable

**Lecture No.** 1

By- DEVA Sir


01 Regulars Vs CFLs Vs RELs

02 FA Vs PDA Vs TM

03 Turing Machine

04

05

# Regulars

I) FA
- DFA
- NFA

II) Regular Grammars
- LLG
- RLG

III) Regular Expressions

# CFLs

I) PDA

II) CFG

# Recursively Enumerable Languages (RELs) (REs)

I) TM

II) Unrestricted Grammars

Regulars

⇓

These problems can be solved without using memory

CFLs

⇓

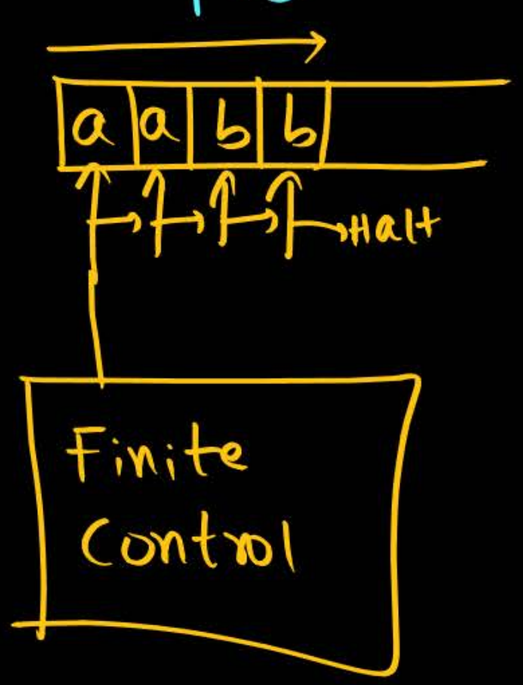
can be solved using 1 stacks with non-determinism

Recursively Enumerable Languages (RELs) (REs)

⇓

Can be solved without any restriction

# FA

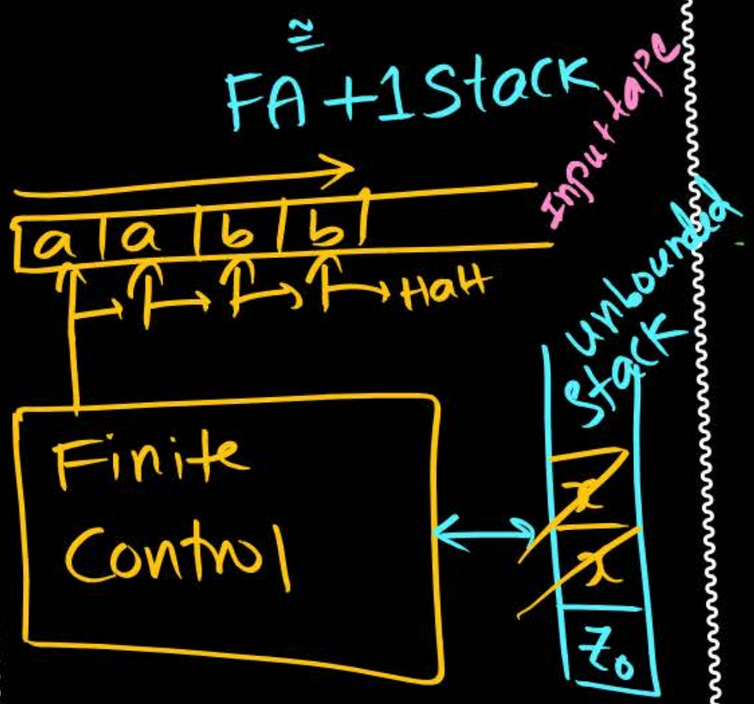



$FA = (Q, \Sigma, \delta, q_0, F)$

$\delta_{DFA} : Q \times \Sigma \to Q$

$\delta_{NFA} : Q \times \Sigma_\varepsilon \to 2^Q$

# PDA
## ≅
## FA + 1 Stack

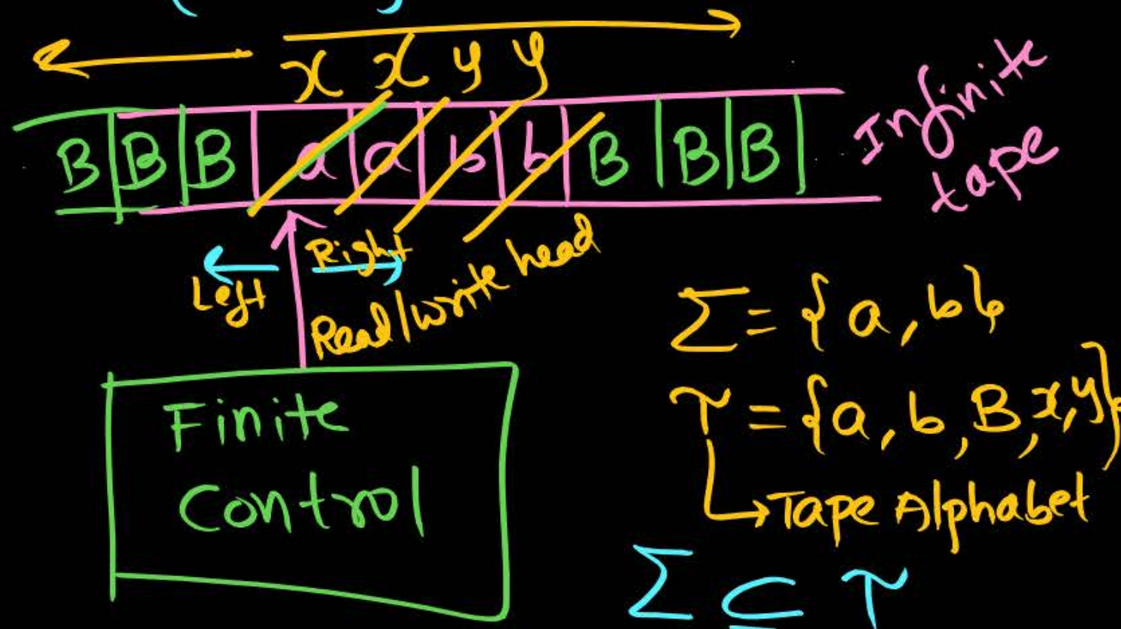



$PDA = (Q, \Sigma, \delta, q_0, F, Z_0, \Gamma)$

$\delta_{PDA} : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon^* \to 2^{Q \times \Gamma^*}$

$\delta_{DPDA} : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$

# TM
## (Turing Machine)




$\Sigma = \{a, b\}$

$\Gamma = \{a, b, B, X, Y\}$

↳ Tape Alphabet

$\Sigma \subseteq \Gamma$

$TM = (Q, \Sigma, \delta, q_0, F, B, \Gamma)$

↳ Blank Symbol

$\delta_{DTM} : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$

$\delta_{NTM} : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$

PDA that uses finite stack $\cong$ FA $\equiv$ Reg language

FA + R/W tape + Bidirectional Head + Infinite tape

$\cong$

Turing M/C

# FA

If we want to remember symbol 'a', How ?


I) Change state

# PDA

I) Change state

OR

II) Push/Pop a onto stack

OR

III) Change state and Push/Pop on stack

# TM

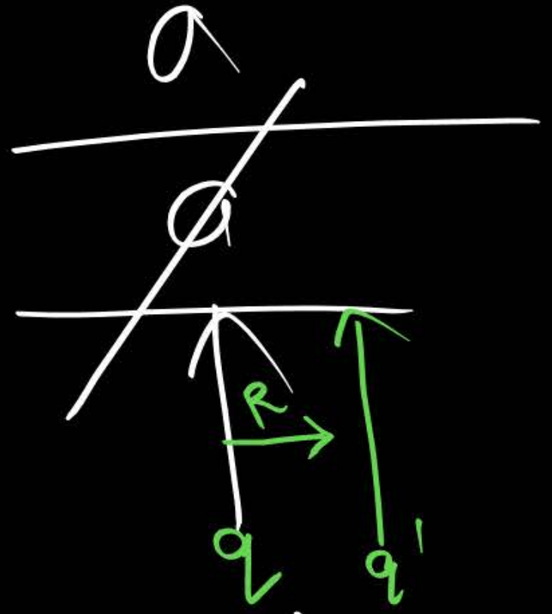
I) Change state

OR

II) Write with new symbol on tape

OR

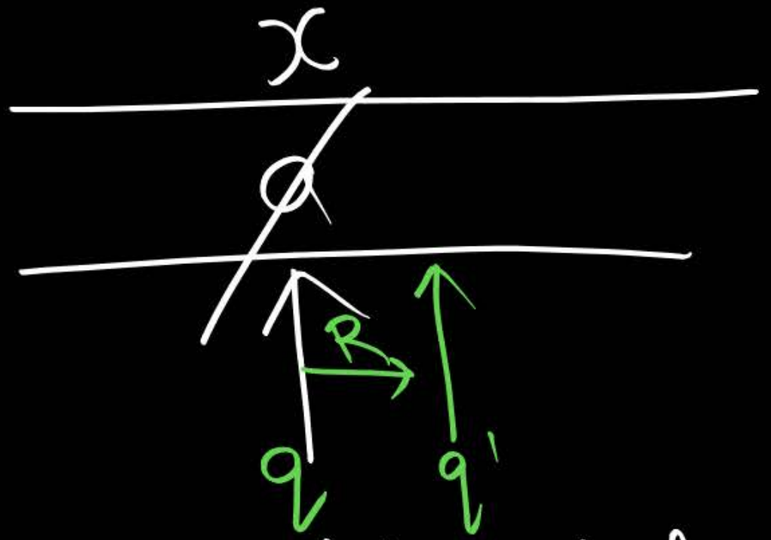
III) Change state & Write


Read 'a'

$$\delta(q, a) = (q', a, R)$$

PS: Present State
Read
NS: Next State
Write
Direction




Read 'a' but write will 'x'.

$$\delta(q, a) = (q', x, R)$$

Find meaning of transitions:

① $\delta(q, B) = (S, a, \leftarrow)$ (Left, L)

From State q, by reading B.

goes to S, by writing with a and moves Left.

② $\delta(q, a) = (q, a, \rightarrow)$ (Right, R)


③ $\delta(q, a) = (q', B, L)$

i/p = a

Replacing a with B

Left Direction

# Turing Machine

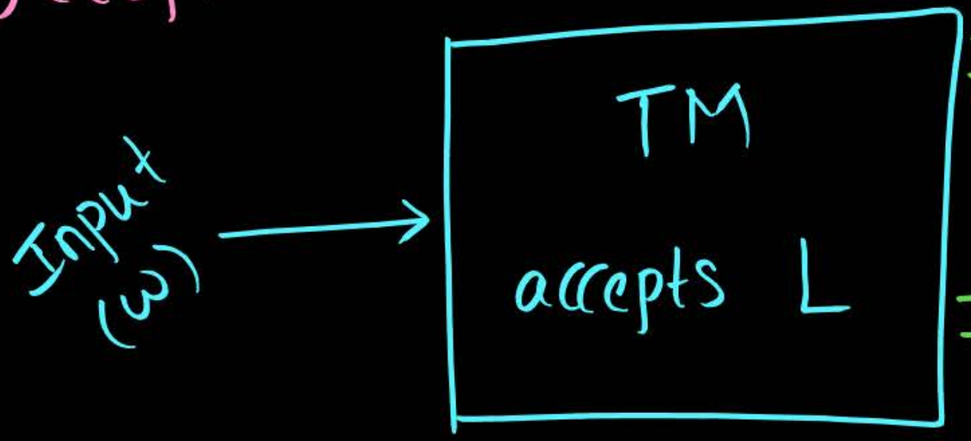
↳ It represents REL (Recursively Enumerable Lang) (Enumerable Language)

I) It can accept (Recognize) REL.

OR

II) It can enumerates REL.

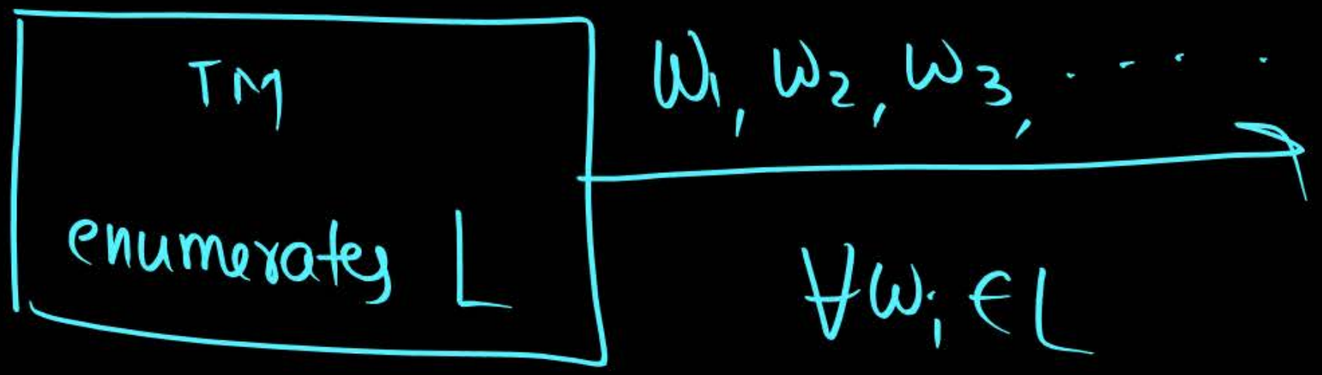
## Acceptor




If $w \in L$, TM halts at final state

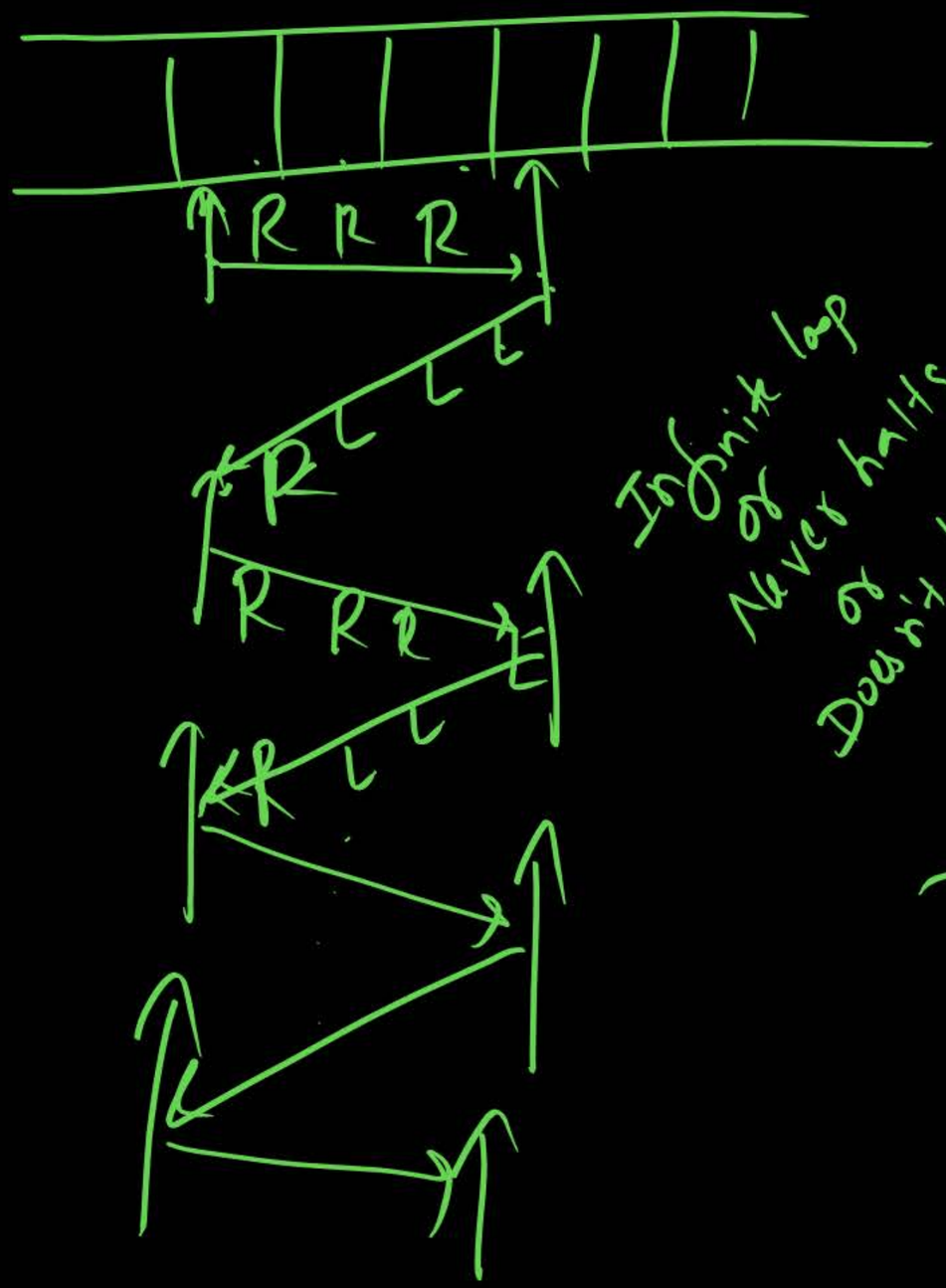
If $w \notin L$, either halts at not final
OR
Never halts

## Enumerator



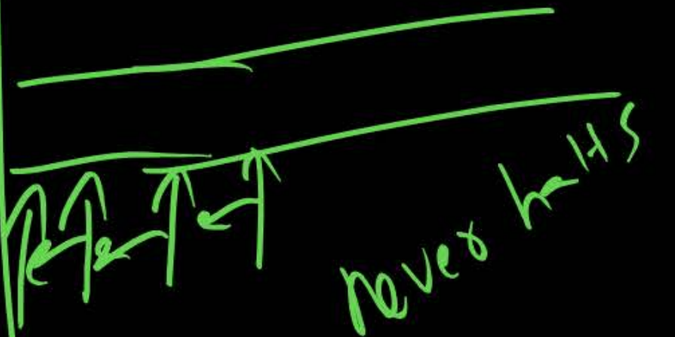
TM $\cong$ Program
C/C++/Java

$\cong$ Computer

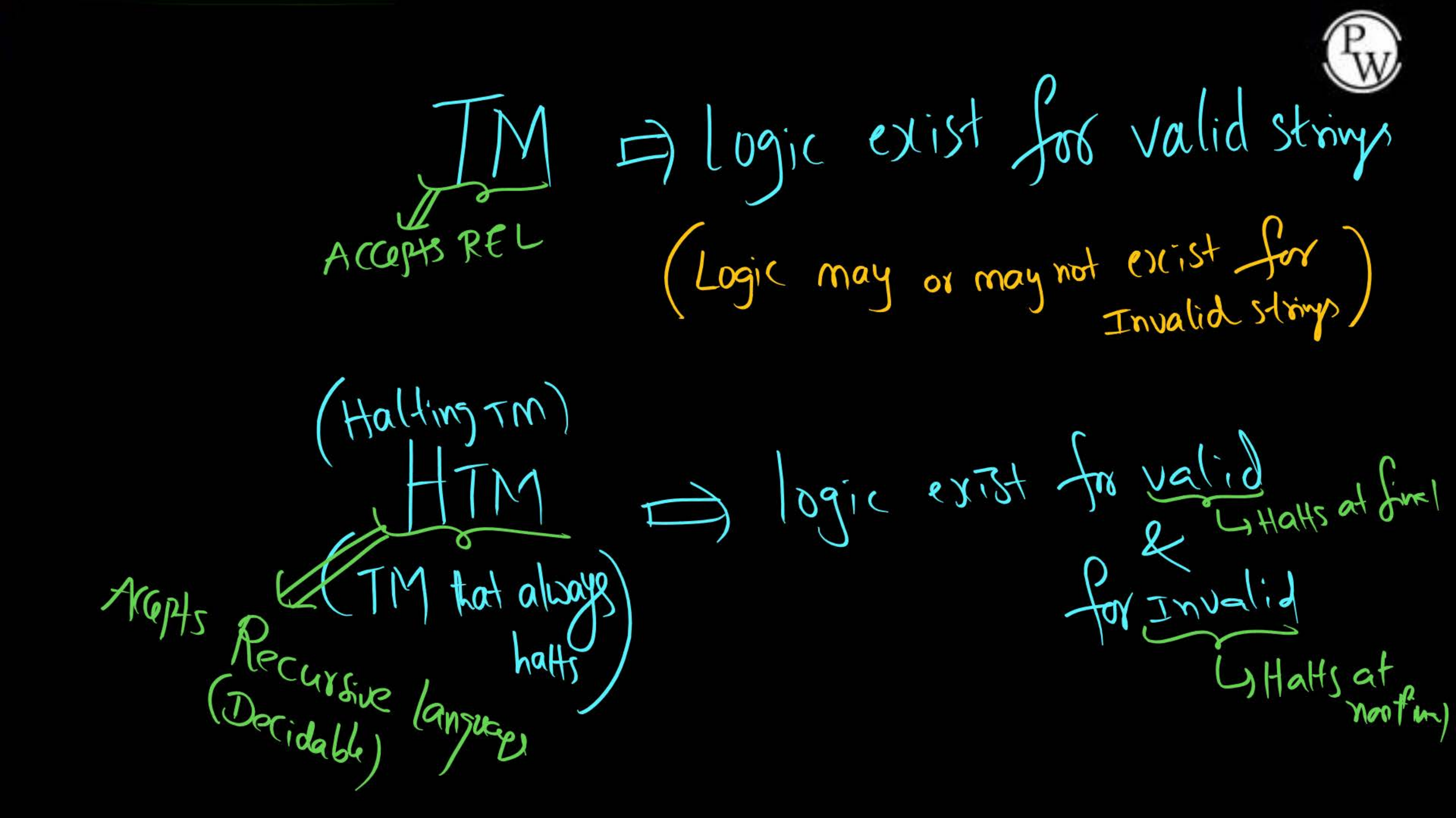


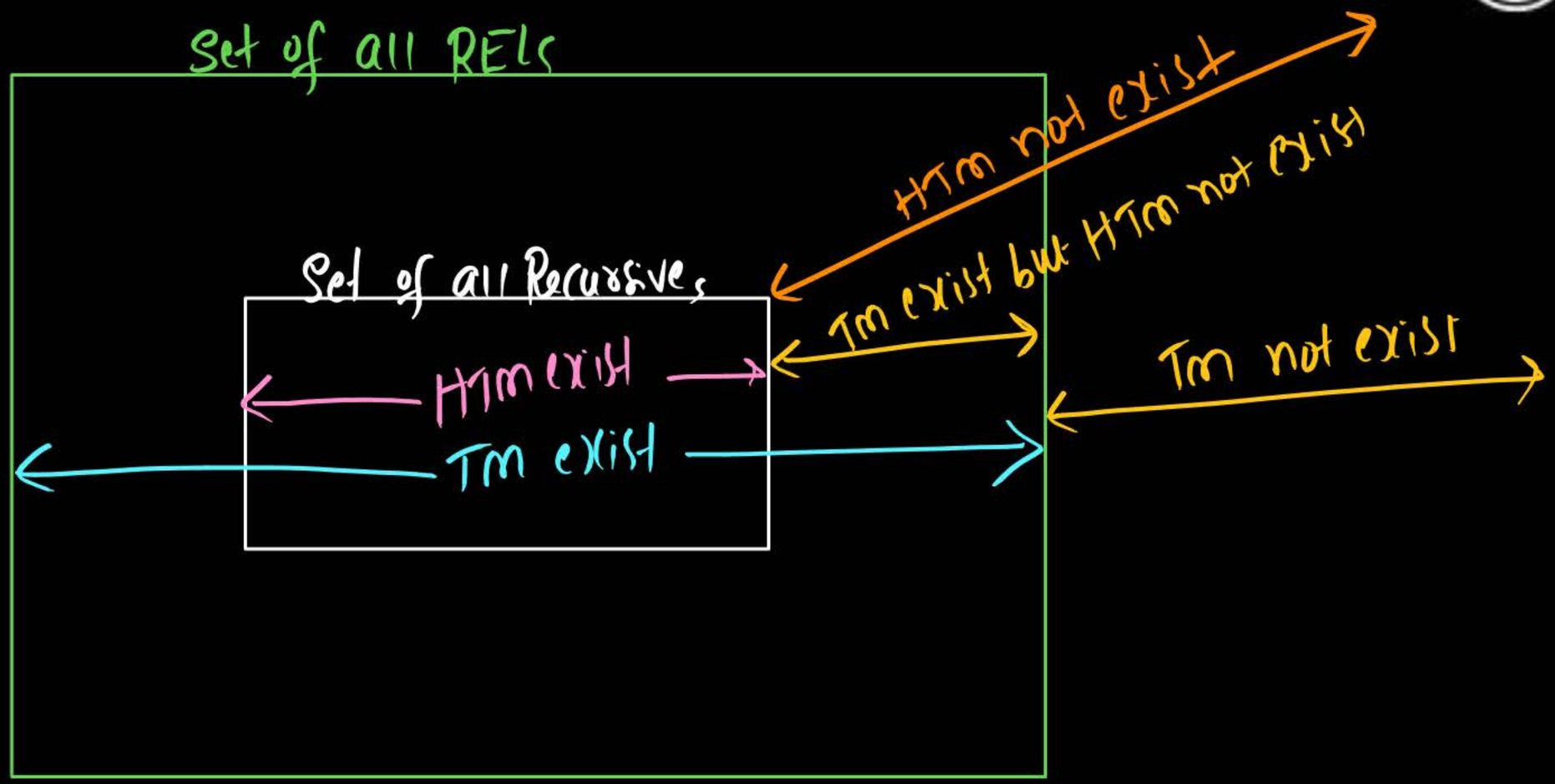

Infinitely moving Right

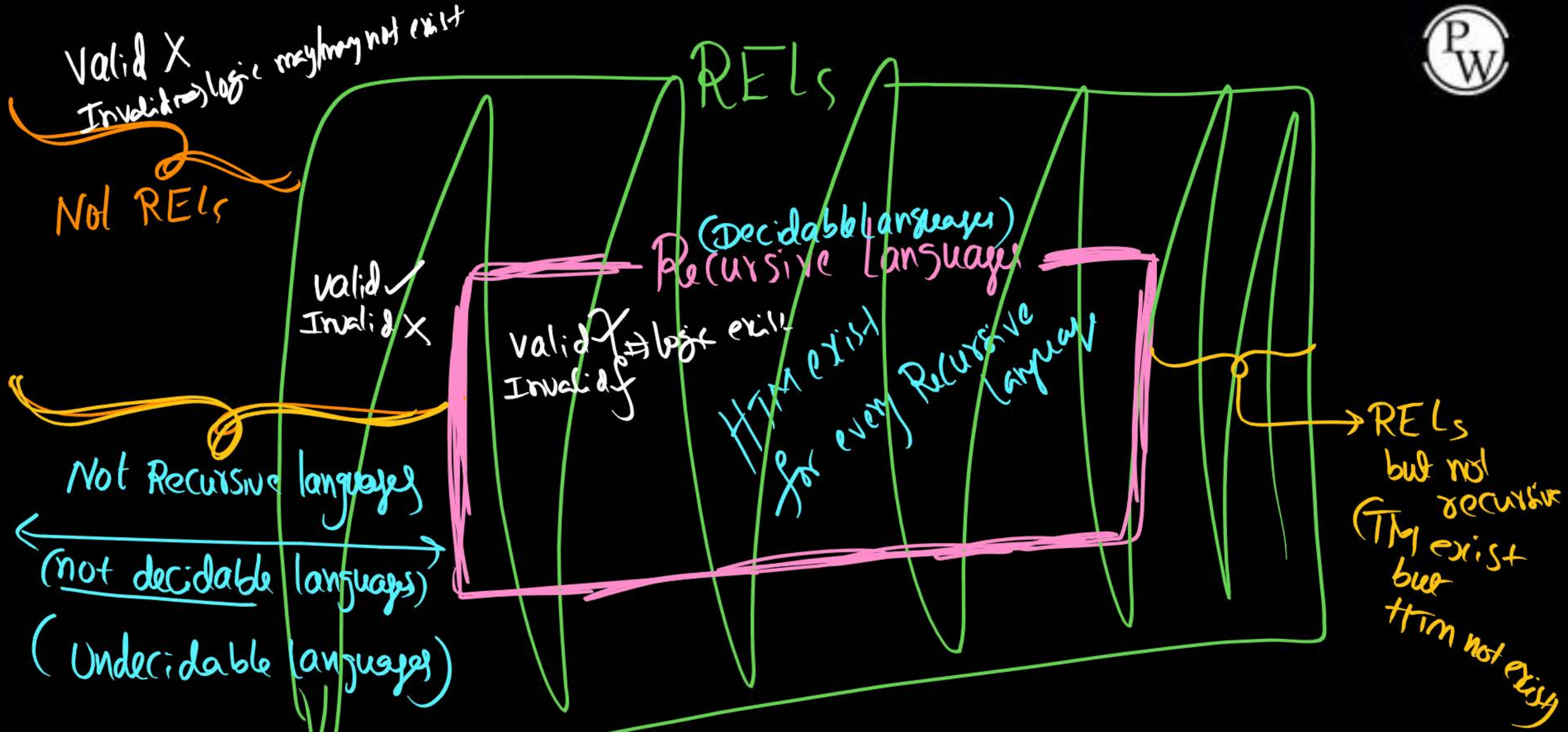
Never halts

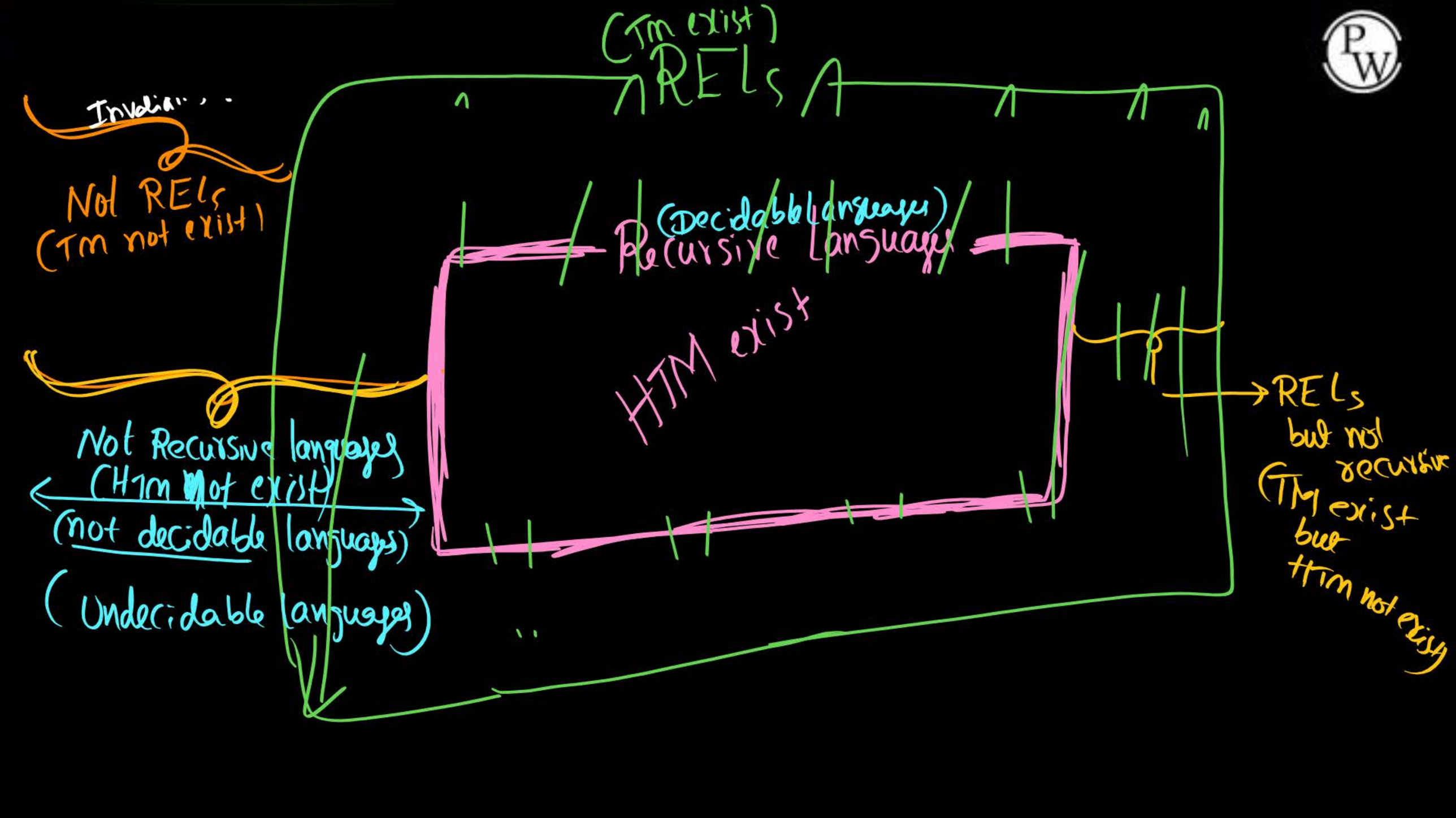


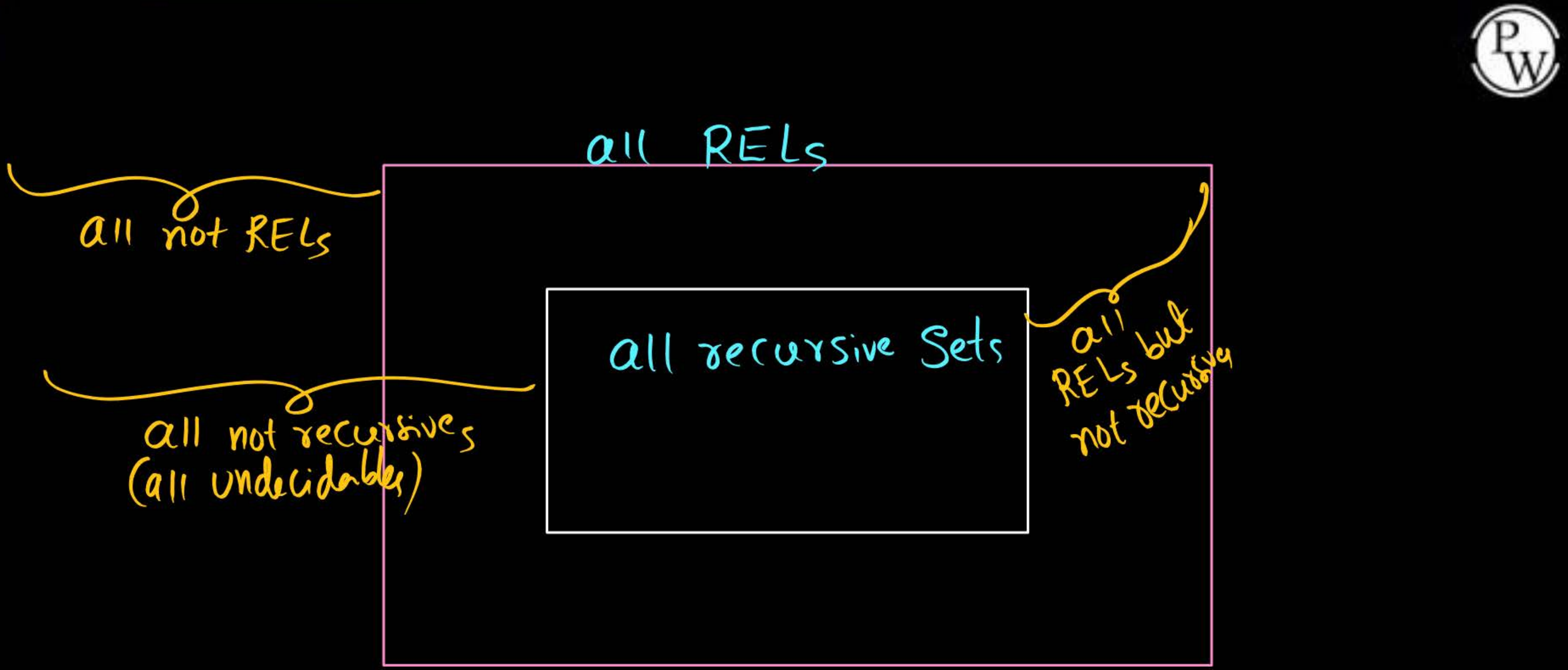
Never halts

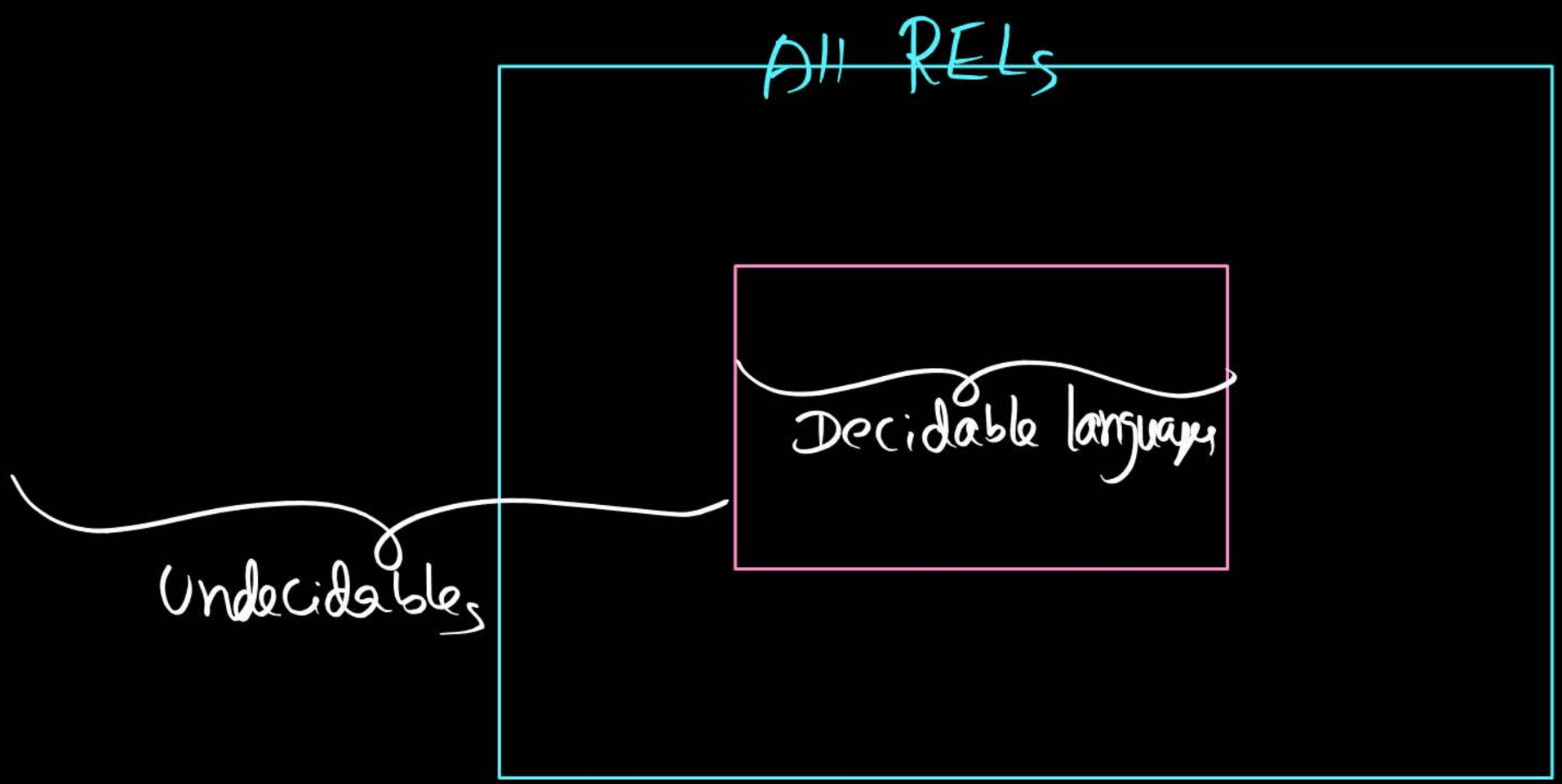

TM
Accepts REL
⇒ logic exist for valid strings
(Logic may or may not exist for Invalid strings)
(Halting TM)
HTM
(TM that always halts)
Accepts Recursive language (Decidable)
⇒ logic exist for valid
↳ Halts at final
&
for Invalid
↳ Halts at non final


Set of all RELs
Set of all Recursives
HTM exist
TM exist
HTM not exist
TM exist but HTM not exist
TM not exist


Valid X
Invalid may/may not exist
Not RELs
RELs
Recursive Languages (Decidable languages)
Valid ✓
Invalid X
Valid ✓ TM exist
Invalid ✗
HTM exist for every Recursive language
Not Recursive languages
(not decidable languages)
(Undecidable languages)
RELs but not recursive
(TM exist but HTM not exist)


(TM exist)
RELs
Invalid
Not RELs
(TM not exist)
(Decidable languages)
Recursive Languages
HTM exist
Not Recursive languages
(HTM not exist)
(not decidable languages)
(Undecidable languages)
RELs
but not
recursive
(TM exist
but
HTM not exist)


all RELs
all not RELs
all recursive Sets
all not recursives
(all undecidables)
all RELs but not recursive


All RELs
Decidable languages
Undecidables

REL

↳ Recursive (Decidable)

or

REL but not recursive

Undecidable Language

↳ REL but not rec

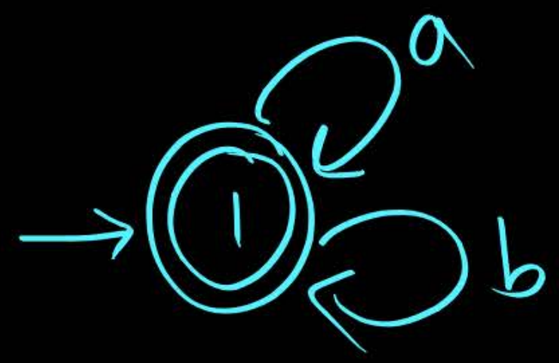
or

Not REL

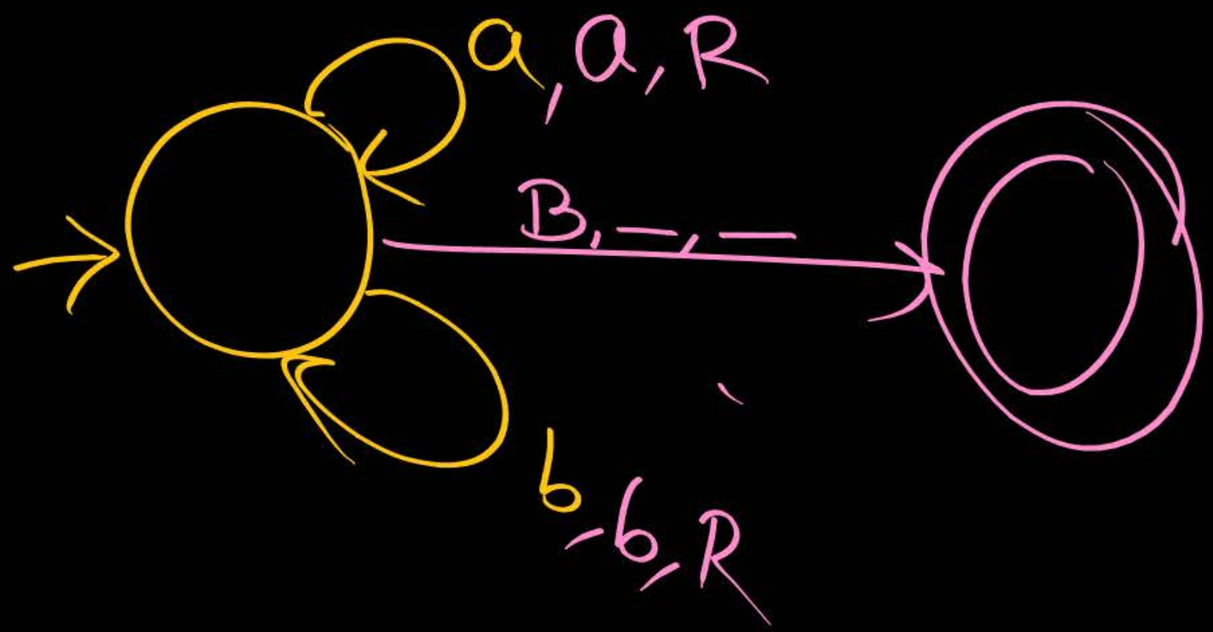
| | $T_M$ | $\overline{H_M}$ |
|---|---|---|
| ① Recursive Language (Decidable Language) | ✓ | ✓ |
| ② REL (Semidecidable Language) | ✓ | Don't know |
| ③ REL but not recursive [SD ∩ UD] | ✓ | ✗ |
| ④ Not REL | ✗ | ✗ |
| ⑤ Undecidable Language | Don't know | ✗ |

# Construction of TM :

① $L = (a+b)^*$



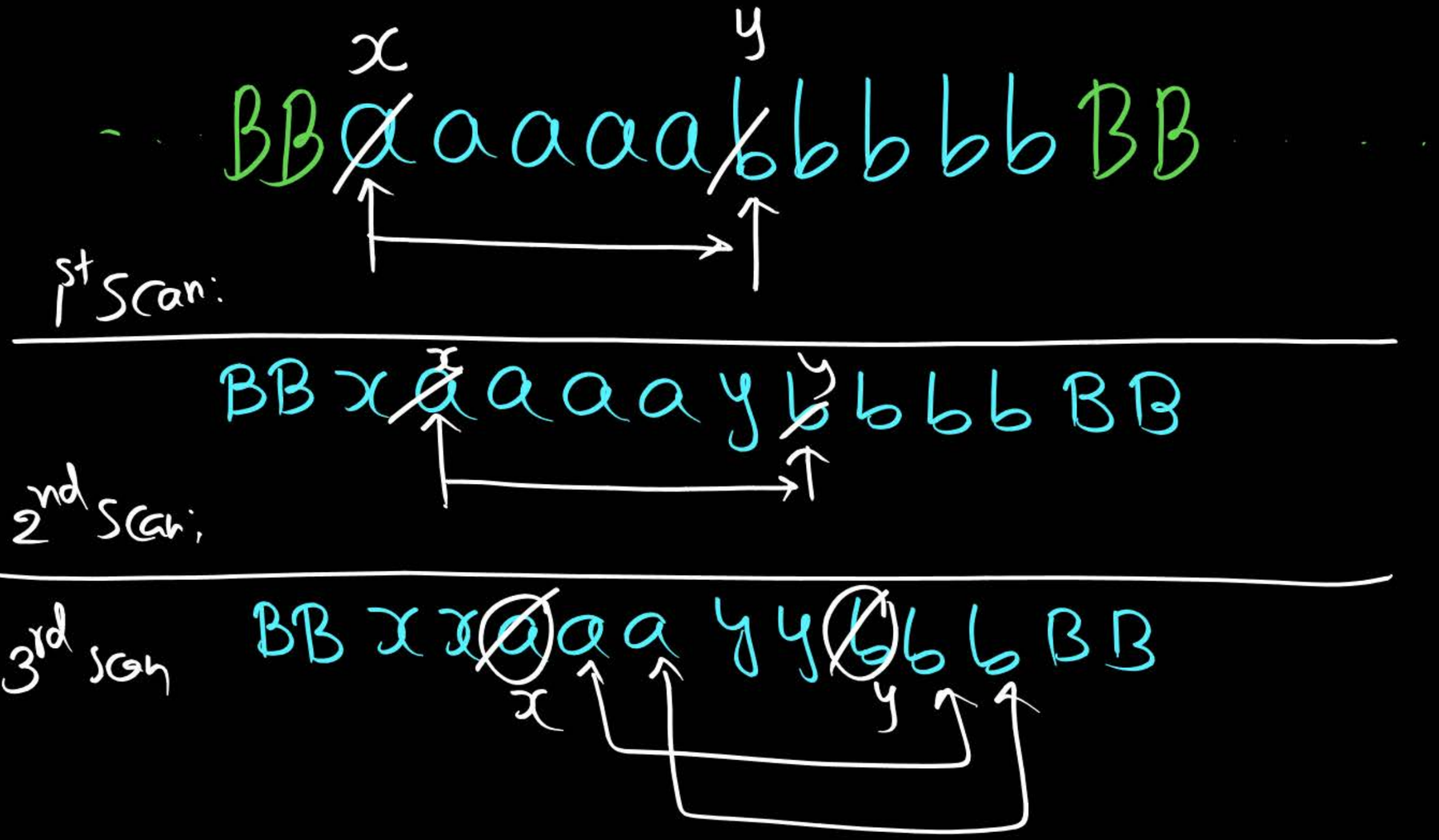

FA



Note : Every FA is convertible to TM / PDA / DPDA / LBA / HTM

(2) $L = \{a^n b^n \mid n \geq 0\}$

H.W.



## Summary

- TM ✓
- TM construction


THANK YOU
GW
SOLDIERS !