CS & IT ENGINEERING

THEORY OF COMPUTATION

Turing Machine

Lecture No.- 01

By- Venkat sir

Recap of Previous Lecture

Topic

60% F.A $\Leftrightarrow$ Regular $\Leftrightarrow$ Regular Expression

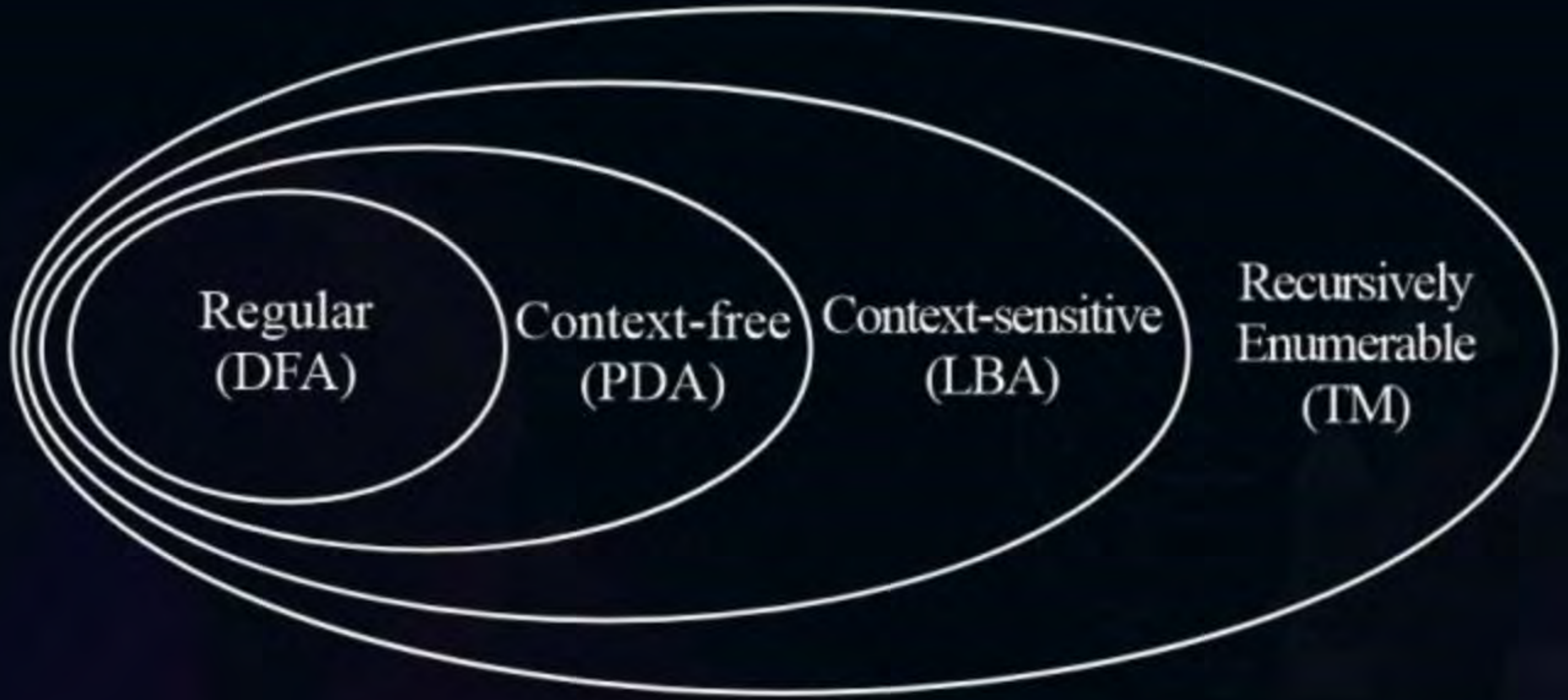
?? 30% Grammar, PDA, CFL

Topics to be Covered

Topic Turing Machine

Topic ?? Recursive Enumerable language

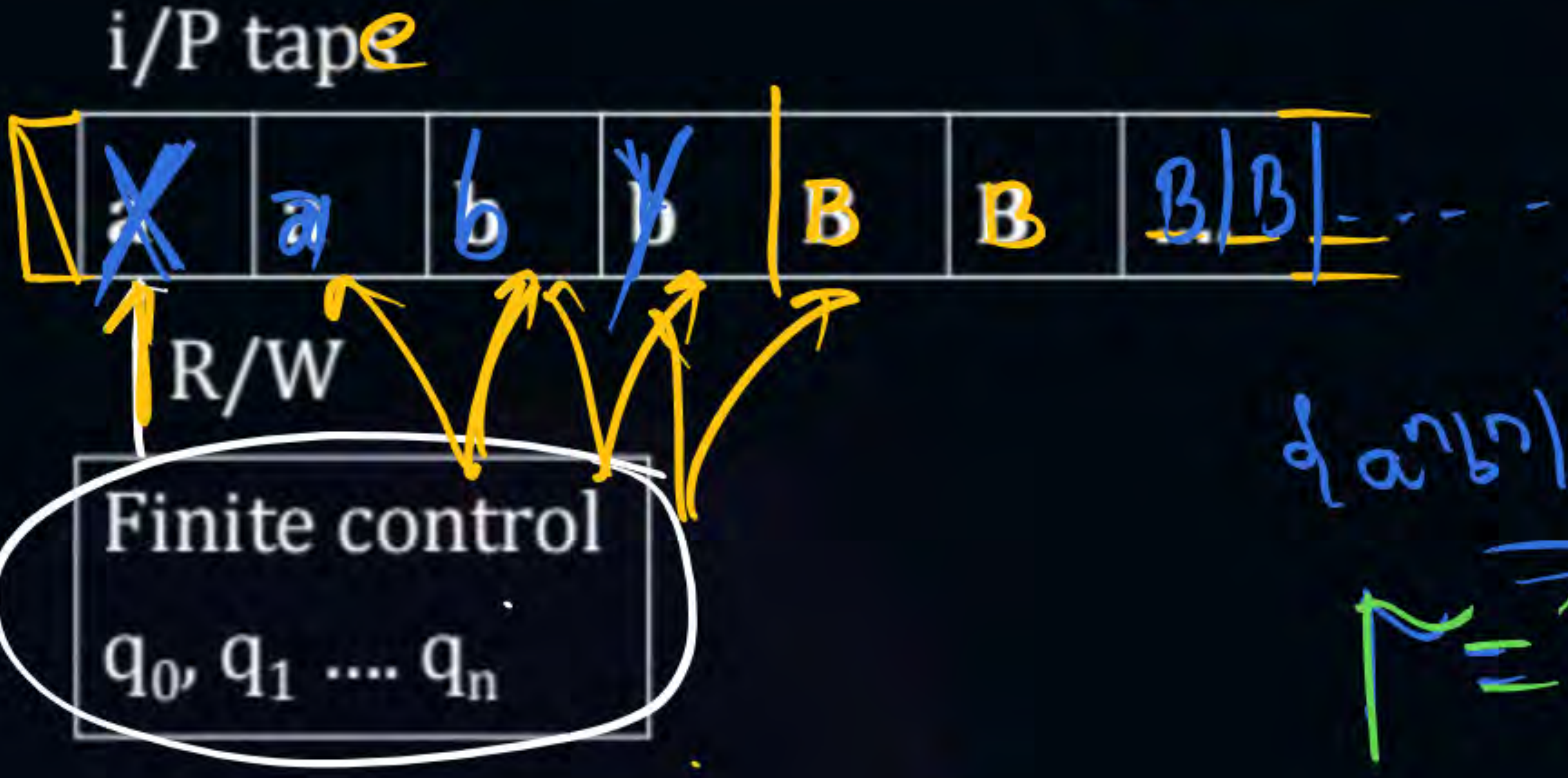
Topic ?? Undecidability


Regular
(DFA)
Context-free
(PDA)
Context-sensitive
(LBA)
Recursively
Enumerable
(TM)

Topic : Turing Machine

Diagram

$T.M > LBA > PDA > F.A$



$\{a^n b^n c^n\}$

$\{a^n b^n c^n d^n\}$

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

$\{a^n b^n \mid n \geq 1\}$

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

1. Infinite length tape ✓
2. Turn around capability ✓
3. Read write capability ✓

Topic : Turing Machine

Alam Turing

- → ① Turing machine is a mathematical model that represents general purpose computer.
- → ② The problem, not solved by Turing machine or not soluble by computer also.
- → ③ Hence Turing machine are used to study power of a compiler.

NOTE:

Computer to finite automata, PDA, Turing having additional property they are

1. **Infinite Length tape:** Turing machine is one side closed and one side infinite.
2. **Turnaround capability:** Turing machine to turn left as well as right side.
3. **Read-Write capability:** Turing machine can replace reading symbol by other or same symbol.

Topic : Turing Machine

Turing Machine = $\boxed{(Q, \Sigma, q_0, F, B, \Gamma, \delta)}$

Q : Finite number of state

Σ : I/p alphabet $\Sigma = \{a, b\}$

q_0 : Initial state → only one

F : Set of final states

B : Blank symbol

Γ : Tape alphabet

δ : Transition function.

Q	×	Γ	→	Q	×	Γ	×{L,R}

$$\boxed{Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}}$$

$$(q_1, X) = (q_2, Y, L)$$

$$(q_2, a) = (q_3, X, R)$$

Topic : Turing Machine

$$|Q| \times |\tau| \rightarrow |Q| \times |\tau| \times \{ L, R \}$$

Notaulus :

$\Rightarrow$ Transition diagram

$\Rightarrow$ Transition Table

Type of TM

(ii) Language Recognizer → yes / no

i/p → O/P generator $\Rightarrow$ OP

Language Generator $\Rightarrow$ Lans

Topic : Turing Machine

Type of Turing Machine

① i/p → Language Recognizer → accept / Reject

② i/p → O/P generator → output

③ → Language generator ⇓ Language

Topic : Turing Machine

Notations

- Transition diagram
- Transition table

Type of Turing Machine

- i/p ① → Language Recognizer
- i/p ② → O/P generator
- ③ → Language generator

Turing machine as a language recognizer-

→ By reading the string Turing machine may halt (or) may not halt (go to infinite loop)

→ By reading string 'X' Turing machine halts ~~as~~ in final state then X is accepted.

→ By reading string 'X' Turing machine halts non-final state then string is rejected

→ By reading string 'X' if Turing machine enters into infinite loop then don't knows about the i/p.

(We can not say anything about whether it is accepted or not.)

Construct a Turing machine

(Q) Construct T.M for $L = \{a^n \mid n \geq 1\}$

a	a	a	B	B	...

transition diagram

T.T

	a	B
$\rightarrow q_0$	(q_1, a, R)	
q_1	(q_1, a, R)	(q_f, B, R)
q_f	final state	

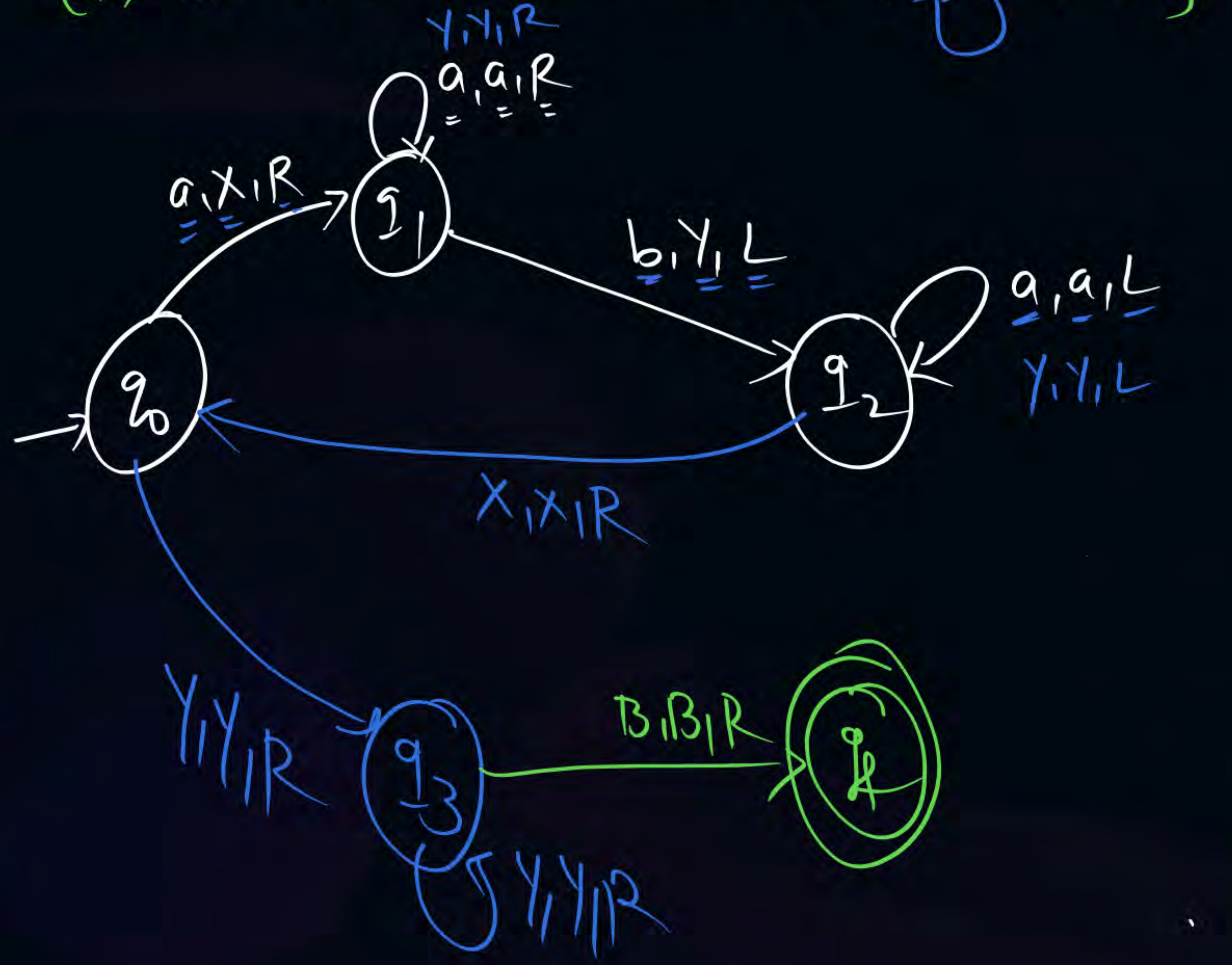
$$\boxed{Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}}$$

$(q_0, a) = (q_1, a, R)$ ✓

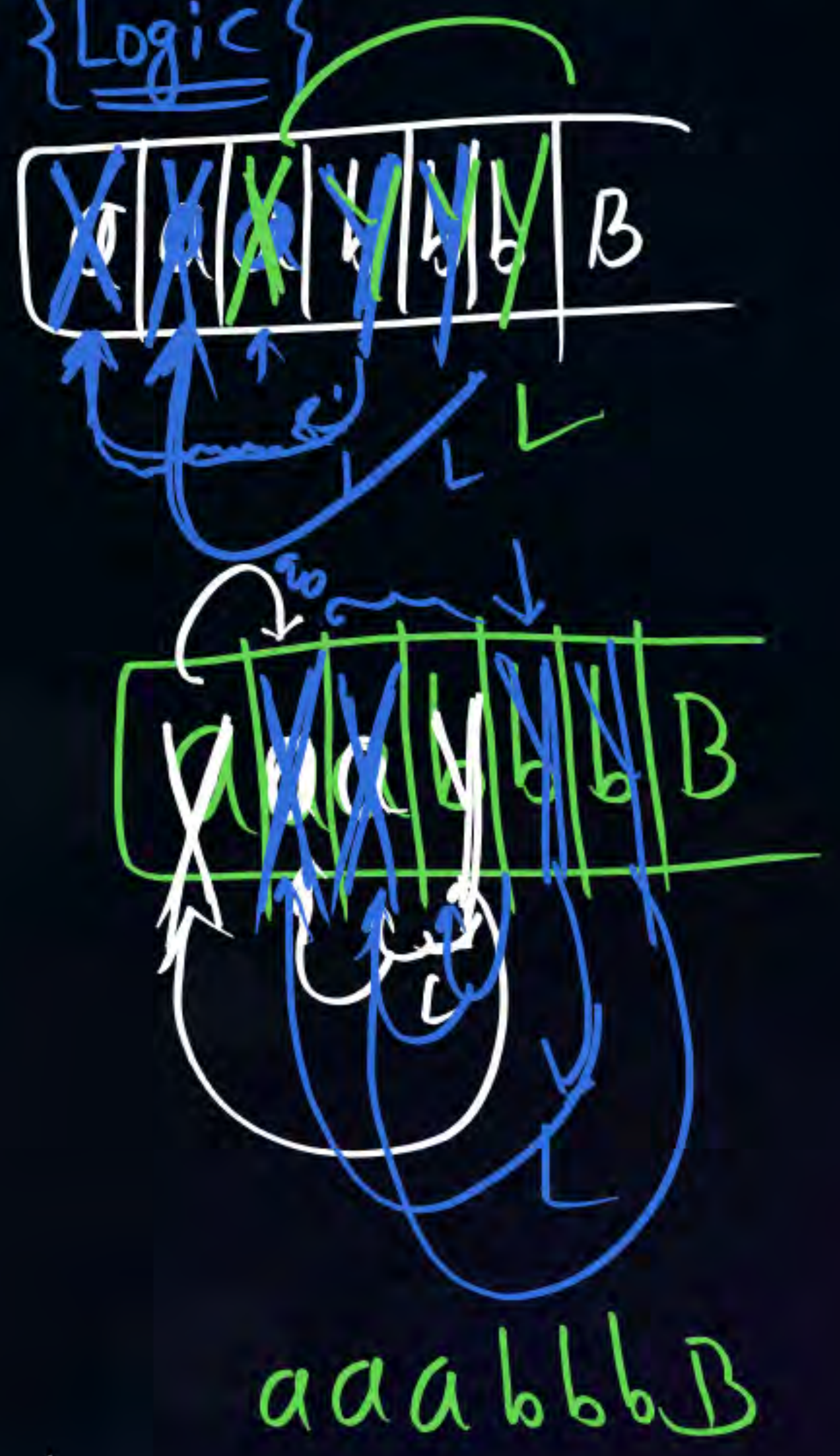
$(q_1, a) = (q_1, a, R)$

$(q_1, B) = (q_f, B, R)$

(Q) Construct T.M for $L = \{a^n b^n \mid n \geq 1\}$ $\{a^n b^n c^n\}$

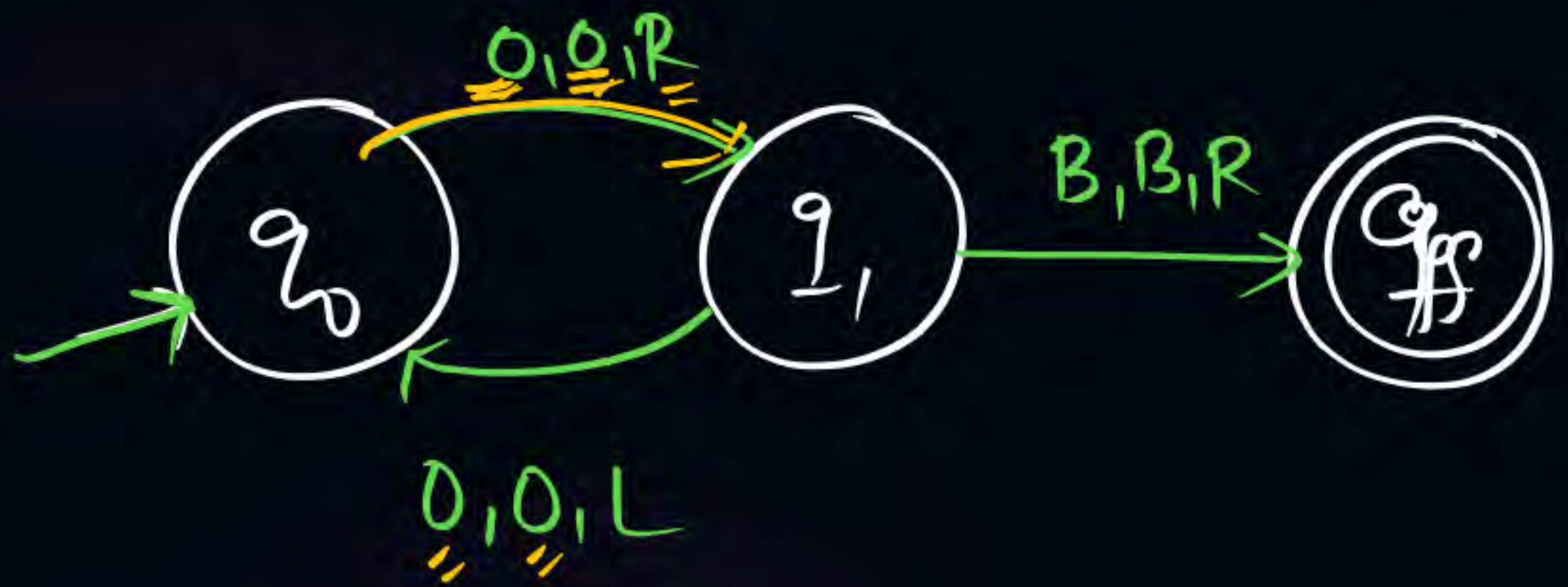


{Logic}



(Q) Construct T.M for $L = \{a^n b^n c^n \mid n \geq 1\}$

(Q) which of the following languge for given T.M?



(a) ~~$L = \{0\}$~~

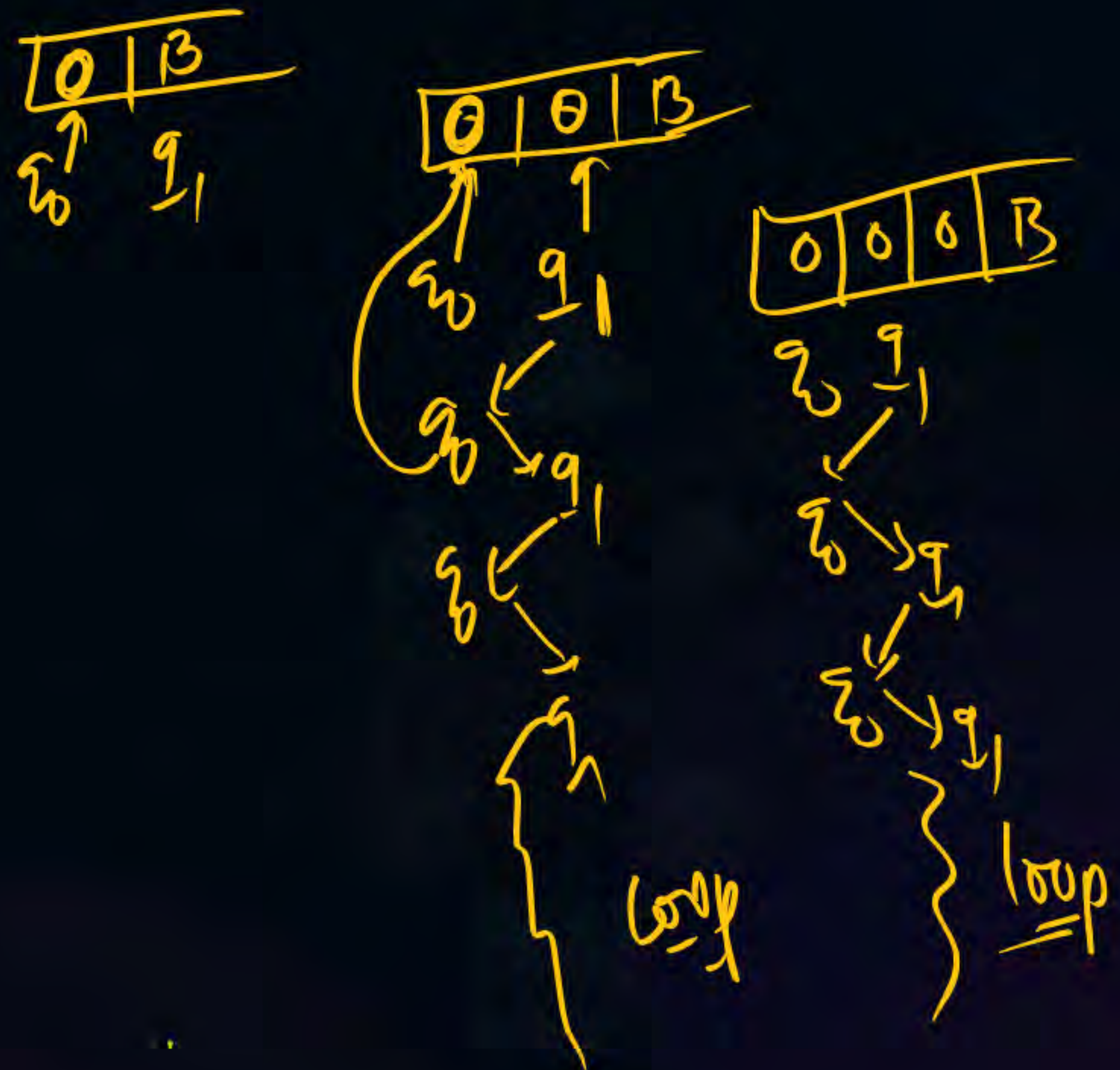
(b) $L = 0^*$

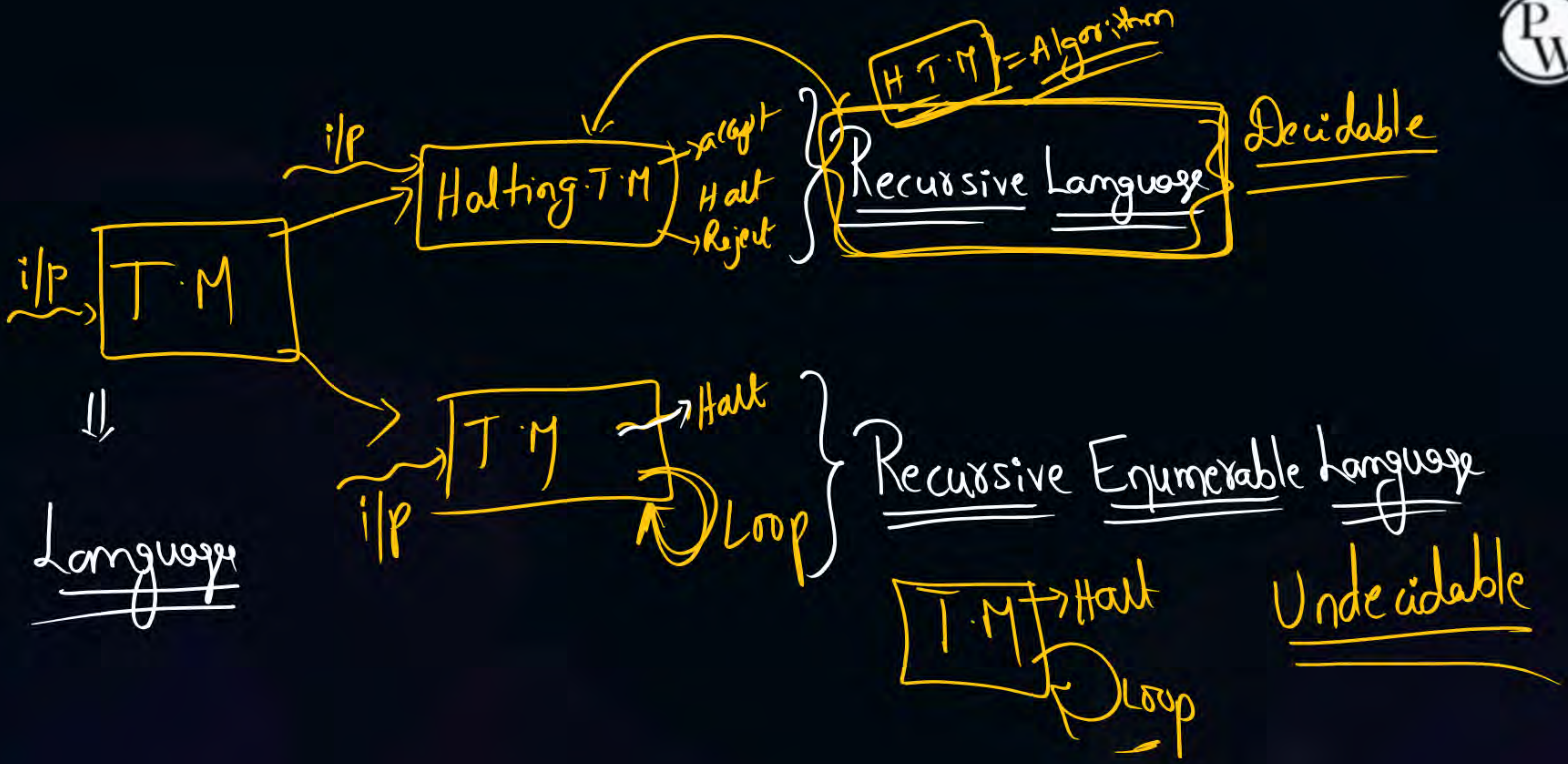
(c) $L = 0(00)^*$

(d) $L = (00)^*$

T.M can enter into infinite loop

0 ✓





i/p
T.M
⇓
Language
i/p
Halting T.M
accept
Halt
Reject
H.T.M = Algorithm
Recursive Language
Decidable
i/p
T.M
Halt
Loop
Recursive Enumerable Language
T.M
Halt
Loop
Undecidable

Recursive Enumerable (RE) or Type-0 Language

RE languages or type-0 languages are generated by type-0 grammars.

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^b^c^]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 b c else move to non-final state.

- So the TM will always halt in this case. REC Languages are also called as Turing decidable languages.

Topic : Turing Machine

$\boxed{T.M} \rightarrow$ Halt } Undecidable

R.E.L

A language 'L' is said to be REL if there exist a Turing machine for that language, that Turing machine may halft on some i/p (or) may not halt on some i/p

→ I.e if the string is valid string of the languages then Turing Machine halts in final state and it says string is accepted.

→ If the string is not belongs to the language in the enter into infinite loop or halt in non final state

→ REL are called as Turing recognizable language

→ If any languages REL then it is undecidable

(number halting Turing machine not exits)

Topic : Turing Machine

NOTE:

All recursive language are R.E.L., but R.E.L. need not be recursive languages.

Hence recursive language are subclass of R.E.L.

→ By Default Turing Machine is may or may not halting Turing Machine.

→ By default Turing recognizable language are recursive enumerable language.

Recursive = T.M → Always Halt   Decidable

non REL = no T.M = Undecidable

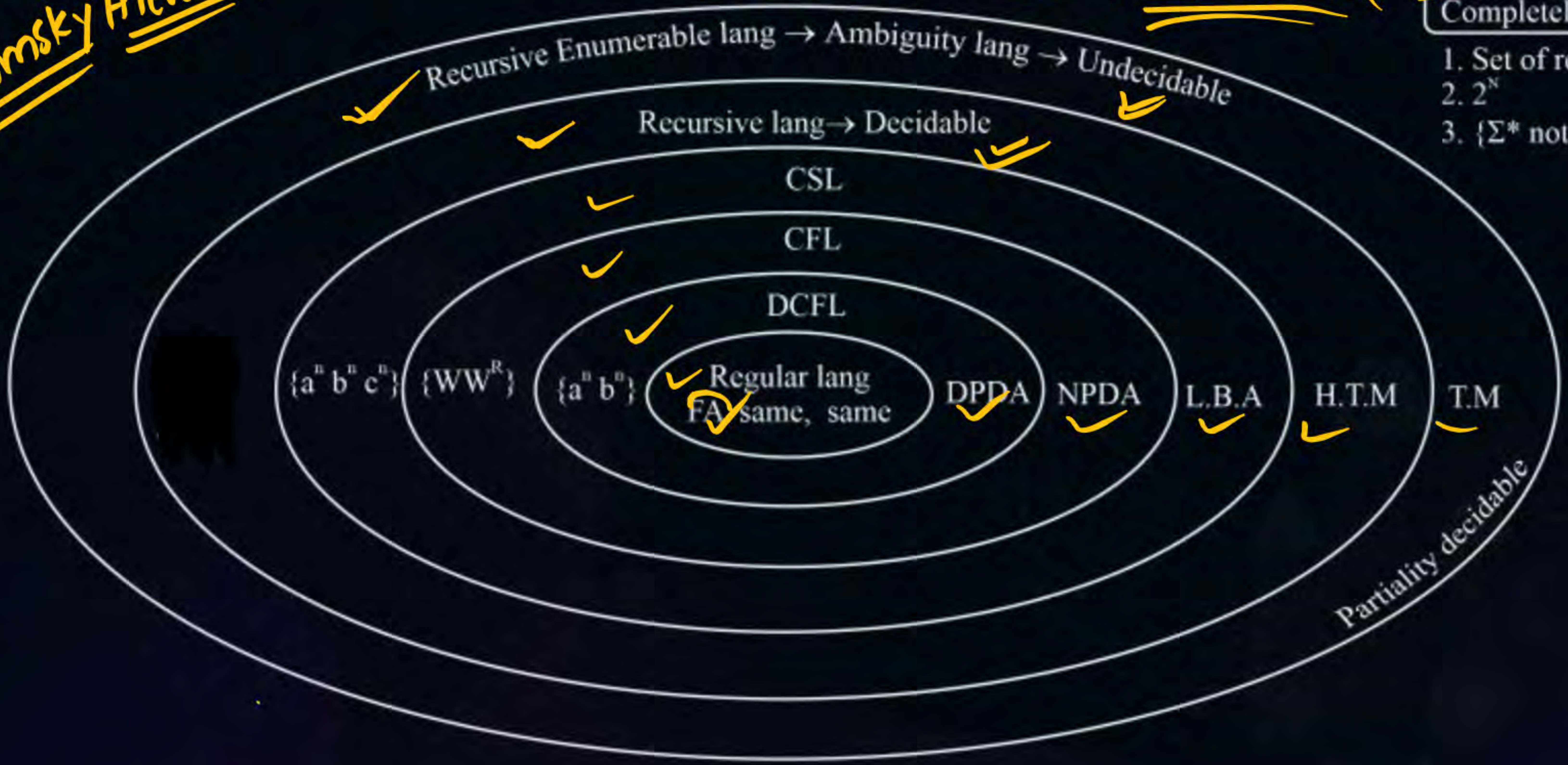
R.E.L = T.M → Halt, → loop } Semi decidable

non REL = no T.M } UD

Topic : Turing Machine

chomsky Hierarchy

non REL (not TM) } Undecidable non REL



Completely un-decidable

1. Set of real number
2. 2^N
3. {Σ* not R.E.L}

Note :

After Modification, the Expressive power of T.M Remains same.
(computing speed may increases).

Topic : Undecidability

• <u>DECIDABLE PROBLEM:::</u>

A problem is set to be decidable if there exist halting I.M. solve the problem.

(or)

There exist Algorithm to solve this problem.

<u>UNDECIDABLE PROBLEM::</u>

- A problem is said to be undecidable if there is NO halting M/e (or) no turtling M/C for that problem (or) No Algorithm exist for that problem.
- To prove a problem 'X' is undecidable, we can use truing machine technique (or) reduction technique.

Topic : Undecidability

Closure Properties of All language

	Regular	DCFL	CFL	CSL	Rec-Lang	REL
1. UNION	✓	X	✓	✓	✓	✓
2. Concatenation	✓	X	✓	✓	✓	✓
3. Intersection	✓	X	X	✓	✓	✓
4. Compliment	✓	✓	X	✓	✓	X
5. Difference $(L_1 - L_2 = L_1 \cap L_2^c)$	✓	X	X	✓	✓	X
6. L ∧ Reg.	✓	✓	✓	✓	✓	✓
7. L – Reg.	✓	✓	✓	✓	✓	✓
8. Kleene closure	✓	X	✓	X	✓	✓
9. Positive closure	✓	X	✓	✓	✓	✓
10. Substitution	✓	X	✓	✓	X	✓
11. Homeomorphism	✓	X	✓	X	X	✓
12. I.H.M.	✓	✓	✓	✓	✓	✓
13. Reverse	✓	X	✓	✓	✓	✓

Recursive Complement is Recursive

R.E.L Complement is may (or) may not REL

Decidable } $\boxed{\text{H.T.M}}$ = Algo exist

Undecidable } { no H.T.M, no T.M } no algo

Whether $W \in T.M$ within $\boxed{\text{1000 steps}}$ Decidable

Topic : Undecidability

Undecidability Table

Problem	Regular	DCFL	CFL	CSL	Rec-Lang	REL
1. is $W \in L$? (membership problem)	D	D	D	D	D	UD
2. is $L = \phi$? (Emptyness problem)	D	D	D	UD	UD	UD
3. is L finite or Not ? (finiteness Problem)	D	D	D	UD	UD	UD
4. is $L_1 = L_2$? (Equivalence Problem)	D	D	UD	UD	UD	UD
5. $L_1 \cap L_2 = \phi$? (Intersection empty)	D	UD	UD	UD	UD	UD
6. is $L = \Sigma^*$ (Completeness problem)	D	D	UD	UD	UD	UD
7. is $L_1 \subseteq L_2$ (Subset Problem) ($L_1 \cap L_2 = L_1$)	D	UD	UD	UD	UD	UD
8. is $(\Sigma^* - L)$ finite or not	D	D	UD	UD	UD	UD
9. is $L_1 \wedge L_2$ = finite or not ?	D	UD	UD	UD	UD	UD
10. is L is regular (Regularity Problem)	D	D	UD	UD	UD	UD
11. Complement of Language is same type or not?	D	D	UD	D	D	UD
12. Intersection of two languages is same type or not?	D	UD	UD	D	D	D

⑥ if $L = \Sigma^*$?

complement $\left(\Sigma^* - L\right) \Leftarrow$ emptyness

[MCQ]

#Q. Context-free languages are

A closed under union

B closed under complementation

C closed under intersection

D closed under Kleene closure

[MCQ]

#Q. If L_1 and L_2 are context free languages and R a regular set, one of the languages below is not necessarily a context free language. Which one?

A. L_1L_2

B. $L_1 \cap L_2$

C. $L_1 \cap R$

D. $L_1 \cup L_2$

[MCQ]

#Q. Let R_1 and R_2 be regular sets defined over the alphabet then

A. $R_1 \cap R_2$ is not regular

B. $R_1 \cup R_2$ is not regular

C. $\Sigma^* - R_1$ is regular

D. R_1^* is not regular

[MCQ]

#Q. If $L_1 = \{a^n \mid n \geq 0\}$ and $L_2 = \{b^n \mid n \geq 0\}$, consider

I. $L_1 \cdot L_2$ is a regular language

II. $L_1 \cdot L_2 = \{a^n b^n \mid n \geq 0\}$

Which one of the following is CORRECT?

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

[MCQ]

#Q. Let L_1 be a recursive language. Let L_2 and L_3 be language that are recursively enumerable but not recursive. Which of the following statements is not necessarily true? false

REL

A. True $L_2 - L_1$ is reclusively enumerable

B. $Rec \wedge REL^c$

$L_1 - L_3$ is reclusively enumerable false

C. T $L_2 \cap L_3$ is reclusively enumerable

D. T $L_2 \cup L_3$ is reclusively enumerable

$\boxed{L_1 - L_2 = L_1 \wedge L_2^c}$

$L_2 - L_1 = L_2 \wedge L_1^c$

$REL \wedge Rec^c$

$REL \wedge Rec$

$REL \wedge REL = REL$

[MSQ]

MSQ

#Q. Which of the following problems are undecidable

(B, D)

A Membership problem in context-free languages → D

B Whether a given context-free language is regular → UD

C Whether a finite state automation halts on all inputs } Decidable

D Membership problem for type 0 languages → UD

[MCQ]

#Q. Which of the following statements is false?

D

$\boxed{T \cdot M}$

A The halting problem for Turing machine is undecidable } true

B Determining whether a context free grammar is ambiguous is undecidable } true

true

C Given two arbitrary context free grammars G_1 and G_2, it is undecidable whether $L(G_1) = L(G_2)$? $\Rightarrow$ UD

D Given two regular grammars G_1 and G_2, it is undecidable whether $L(G_1) = L(G_2)$?

false

Decidable

[MCQ]

#Q. Consider the following problems L(G) denotes the language generated by a grammar G. L(M) denotes the language accepted by a machine M.

I. For an unrestricted grammar G and a string w, whether $w \in L(G.)$? membership $\Rightarrow$ UD

II. Given a Turing Machine M, whether L(M) is regular. $\rightarrow$ UD

III. Given two grammars G_1 and G_2, whether $L(G_1) = L(G_2)$. $\rightarrow$ equival $\Rightarrow$ UD

D IV. Given an NFA N, whether there is a deterministic PDA P such that N and P accept the same language. DCFL equivalent Decidable

Which one of the following statements is correct?

A. Only I and II are undecidable

B. Only III is undecidable

C. Only II and IV are undecidable

D. Only I, II and III are undecidable

THANK - YOU