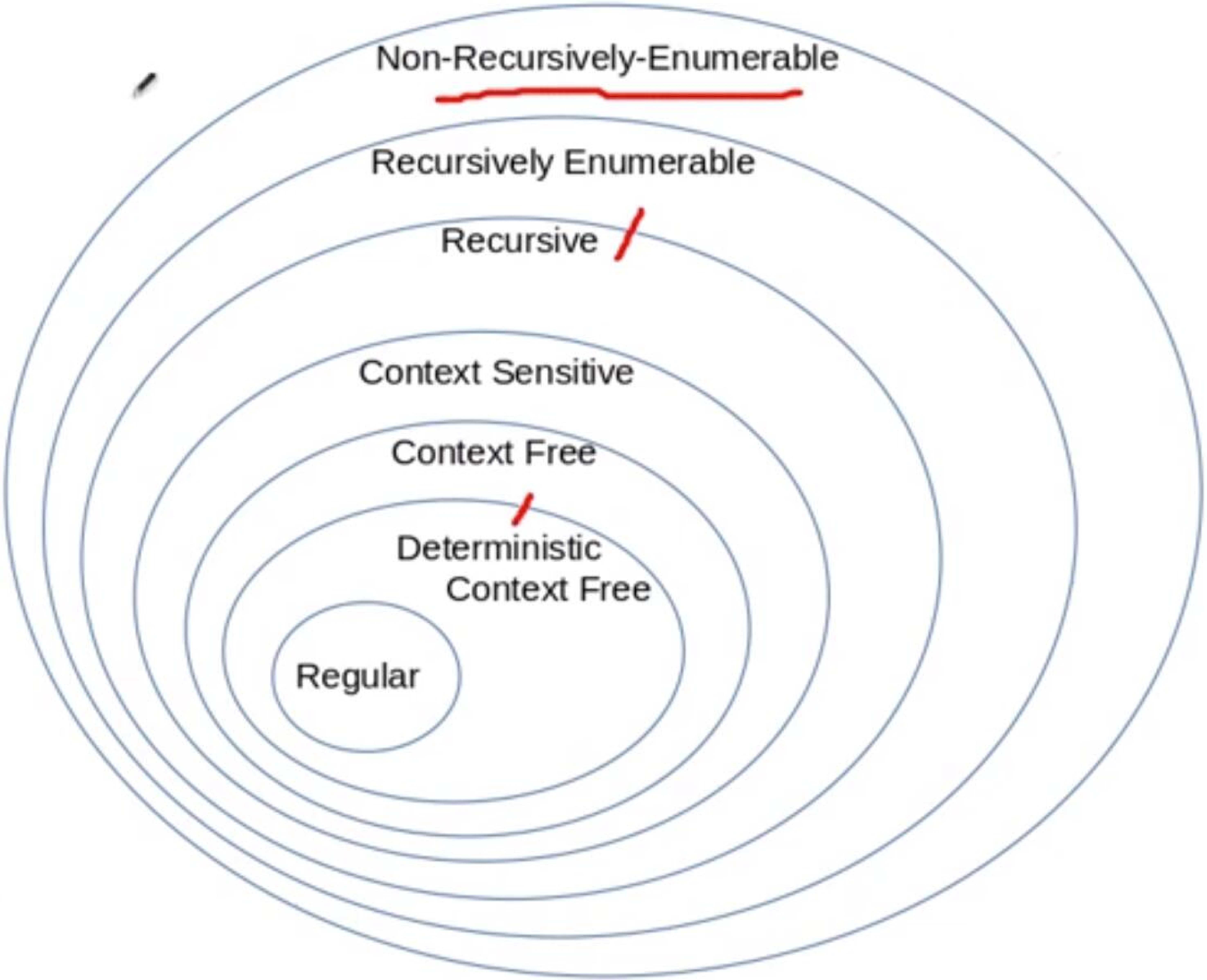
Hierarchy of Language Classes



Typical Questions in Theory of Computation - 4

- Suppose some languages are given
  and we know whether the given languages do or do not belong to certain classes.
- From the given languages, construct new languages using
  union / intersection / inversion / Keene's closure / quotient / homomorphism etc

For newly constructed languages,
determine whether they do or do not belong to certain language classes.

Example : Let $L_1 = \{w \in \{0,1\}^* \mid w \text{ constains even no. of 0s }\}$

$L_2 = \{w \in \{0,1\}^* \mid w \text{ constains even no. of 1s }\}$

(Suppose we already know that L_1 and L_2 are regular)

Prove that $L_1 \cap L_2$ is regular.

We shall prove theorems like - *Intersection of any two regular language is also regular*

These sort of properties are called **closure properties** of a language class.

Typical Questions in Theory of Computation - 3

Prove equivalence of different types of computational models.

Example : Prove that, for all NFA there is an equivalent DFA
for all CFG there is an equivalent PDA
... etc

Prove that certain language cannot be accepted by any model of specific type.

Example : Prove that, there does not exist any finite automata which can accept the language, $\{0^i 1^i \mid i \in \mathbb{N}\}$

Prove that certain type of models are more powerful than other type of models

Example : Prove that PDAs are more powerful than DFAs.
(Notice that, it is sufficient to prove that $\{0^i 1^i \mid i \in \mathbb{N}\}$ is accepted by some PDA)

Typical Questions in Theory of Computation - 2

Given a language, construct a particular type of model which accepts that language

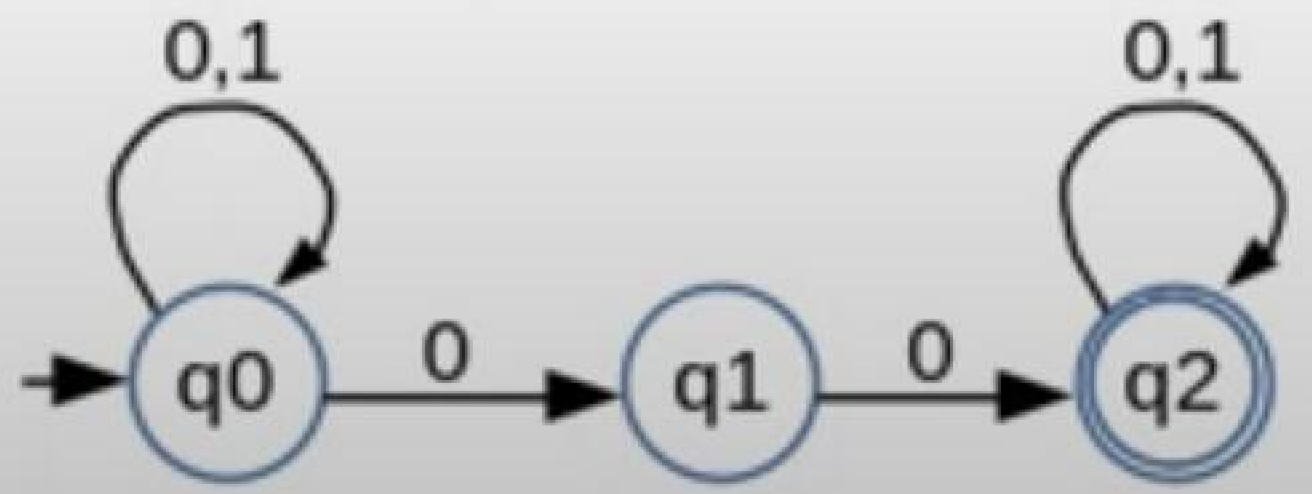
Example : Let $L = \{x \in \{0,1\}^* \mid x \text{ contains equal number of 0's and 1's}\}$

Construct a Context Free Grammar which accepts L

Construct a Pushdown Automata which accepts L

Given a model M construct another different type of model M' s.t. L(M')=L(M)

Example : Construct a DFA which is equivalent to the following NFA



Typical Questions in Theory of Computation - 1

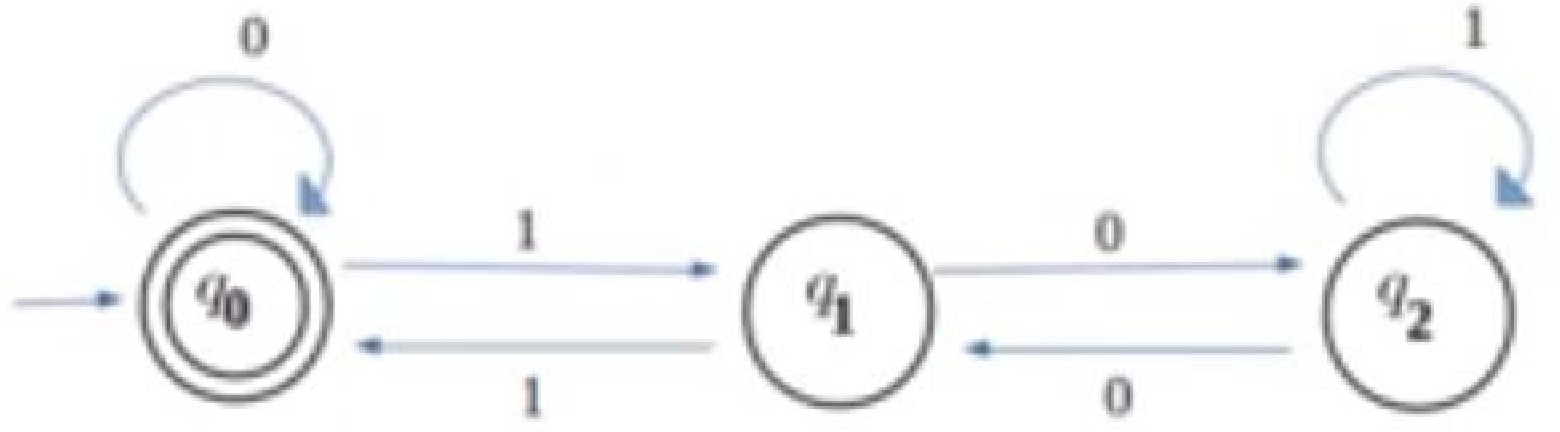
Prove that two languages (defined in different ways) are actually same.

Example : Let $A, B \subset \Sigma^*$. Prove that $(A \cup B)^* = (A^* B^*)^*$

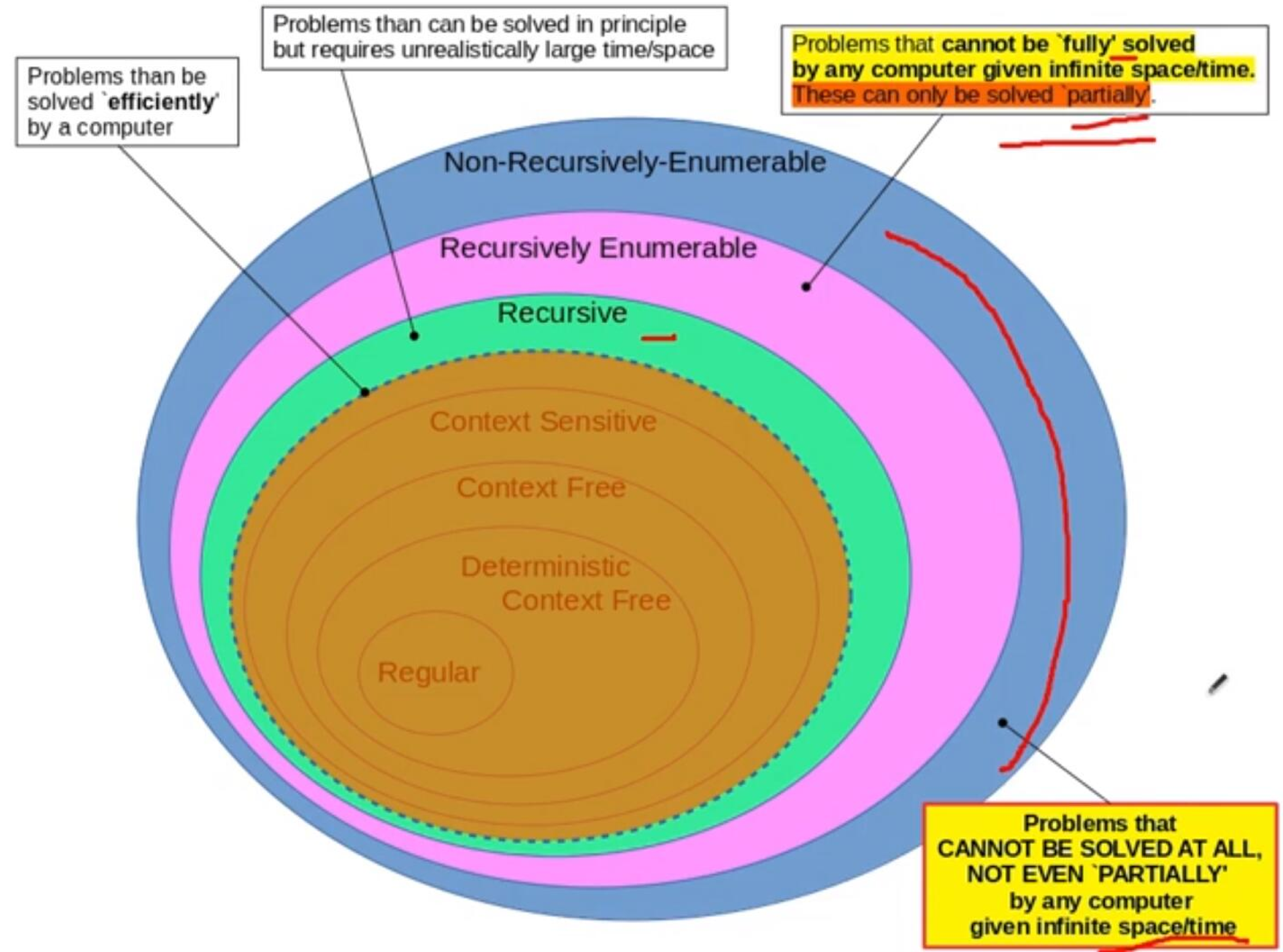
Given a language and a model prove that the model accepts the given language.

Example : Let $L = \{x \in \{0,1\}^* \mid x$ is binary representation of an integer divisible by $3\}$

And let M be the DFA

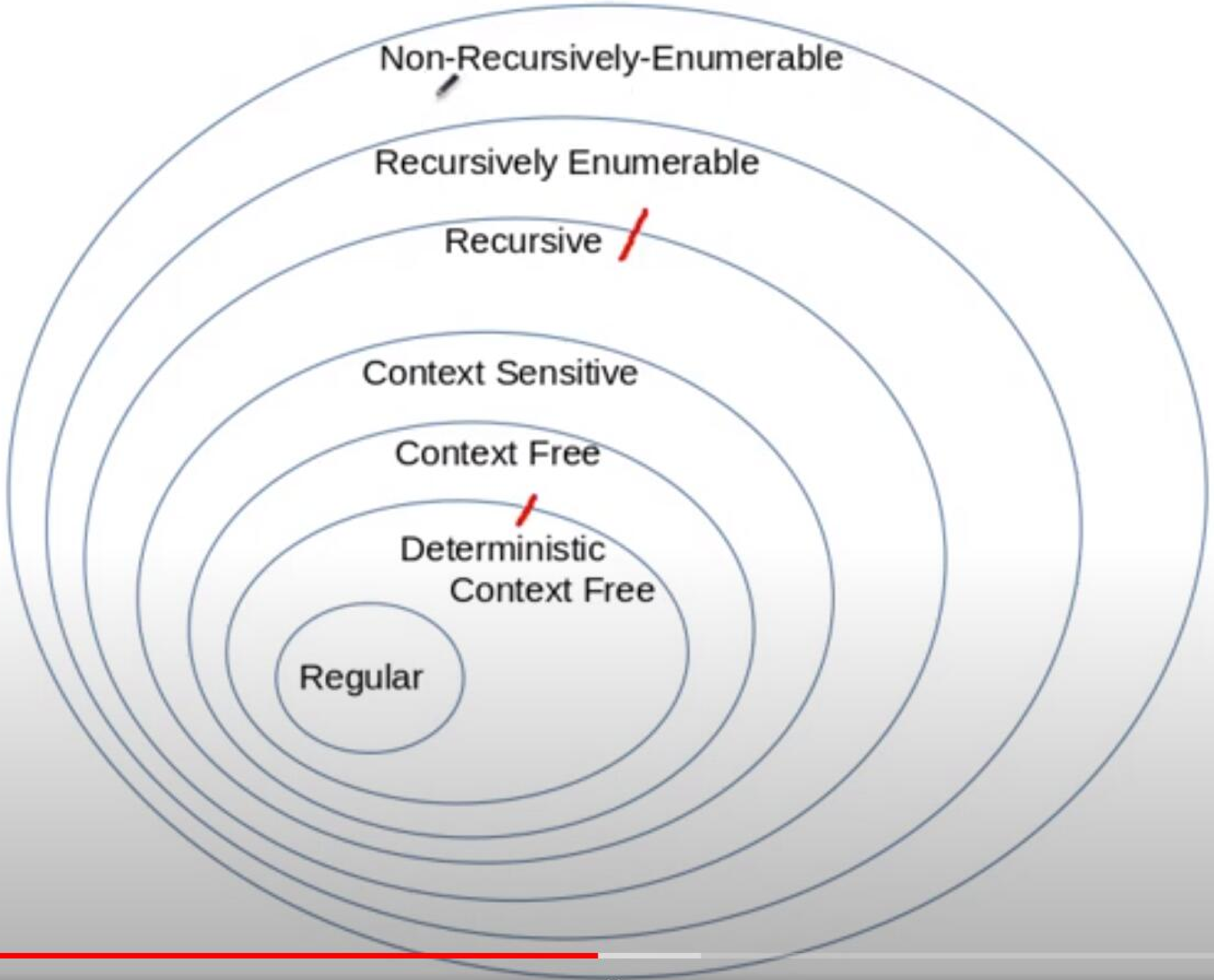


$$\text{Prove that } L(M) = L$$



Problems than be solved `**efficiently**' by a computer
Problems than can be solved in principle but requires unrealistically large time/space
Problems that **cannot be `fully' solved by any computer given infinite space/time.** These can only be solved `partially'.
Non-Recursively-Enumerable
Recursively Enumerable
Recursive
Context Sensitive
Context Free
Deterministic Context Free
Regular
Problems that CANNOT BE SOLVED AT ALL, NOT EVEN `PARTIALLY' by any computer given infinite space/time

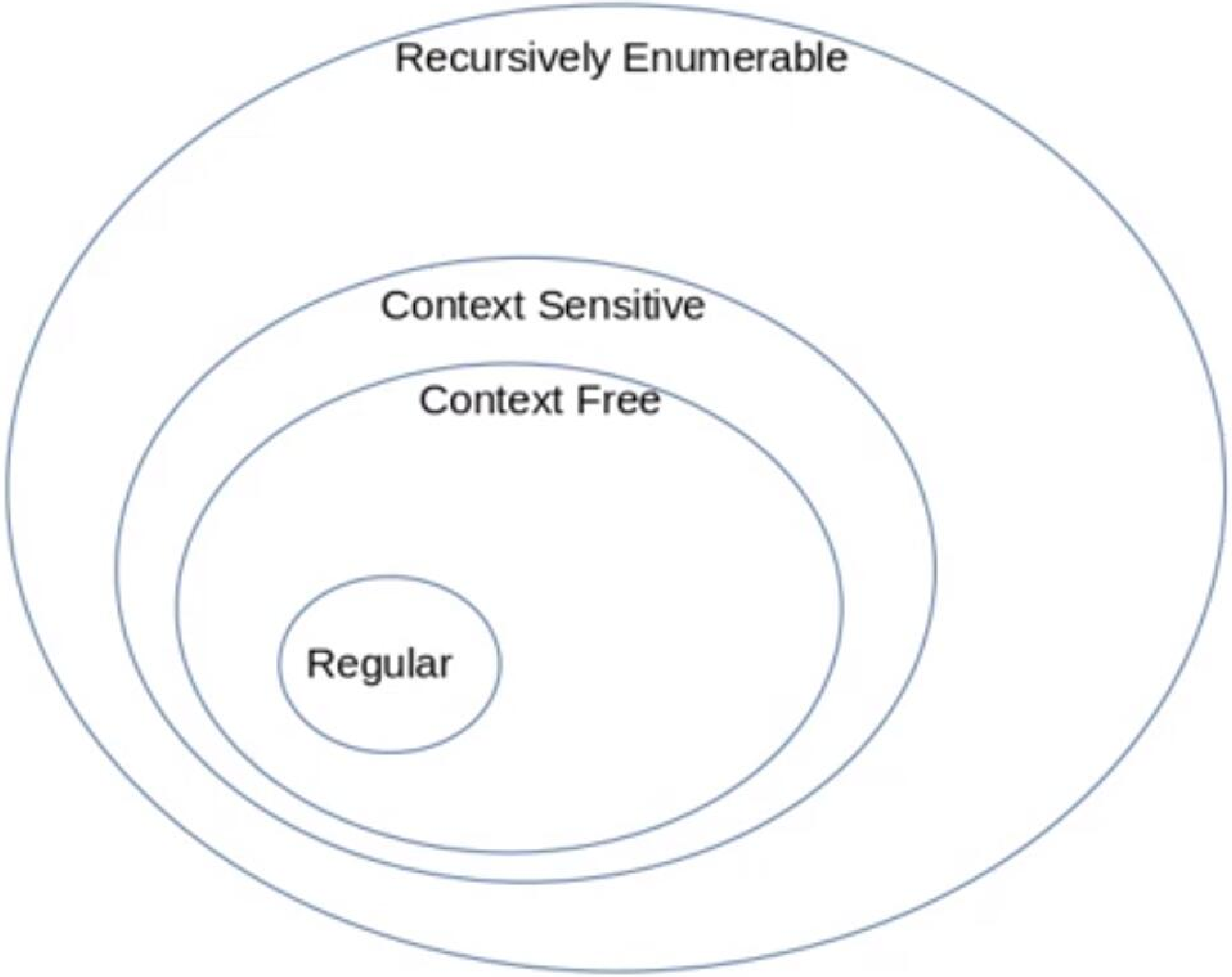
Hierarchy of Language Classes



Chomsky Hierarchy

Regular $\subset$ Context Free $\subset$ Context Sensitive $\subset$ Recursively Enumerable

Non-Recursively-Enumerable



Strictly more powerful

Different Models of Computation		Corresponding Language Class
Equivalent	Deterministic Finite Automata Nondeterministic Finite Automata Regular Expression, Regular Grammar	Regular language
Equivalent	Deterministic Pushdown Automata Deterministic Context Free Grammar	Deterministic Context Free Language
Equivalent	Nondeterministic Pushdown Automata Nondeterministic Context Free Grammar	Context Free Language
Equivalent	Linear Bounded automata Context Sensitive Grammer	Context Sensitive Language
	Turing Machine that halt on all input	Recursive Language
Equivalent	Deterministic Turing Machine Nondeterministic Turing Machine Unrestricted Grammer	Recursively Enumerable Language

Problems that cannot be solved by any computer given infinite space and time

Such languages exist !!!

Hierarchy of Different Models / Language Classes

Q. What do we mean by - "PDAs are (strictly) more powerful than DFAs (or NFAs)" ?

A. If a language is accepted by some DFA then it is also accepted by some PDA but there exists a language which is accepted by a PDA and there does not exist any DFA that can accept it.

$$\forall \text{ DFA } D, \exists \text{ some PDA } P \text{ s.t. } L(D) = L(P)$$

$$\text{but... } \exists \text{ a PDA } P \text{ s.t. } \forall \text{ DFA } D, L(D) \neq L(P)$$

$\Downarrow$

The class of CFLs is a **strict superset** of the class of Regular languages

$\Downarrow$

Set of problems that PDA/CFG can solve
is a strict superset of the
set of problems that Finite Automata can solve

Equivalence of Different Models Contd.

Similarly PDAs and CFGs are equivalent in power.

Set of all languages accepted by PDAs
= Set of all languages accepted by CFGs

This set of languages is called the **class of Context Free Languages** or **CFL**

represents **class of problems that PDA/CFGs can solve**

Similarly LBAs and CSGs are equivalent in power.

Set of all languages accepted by LBAs
= Set of all languages accepted by CSGs

This set of languages is called the **class of Context Sensitive Languages** or **CSL**

represents **class of problems that LBA/CSGs can solve**

Similarly DTMs , NTMs and Unrestricted Grammar are all equivalent in power.

Set of all languages accepted by DTMs
= Set of all languages accepted by NTMs

This set of languages is called the **class of Recursively Enumerable Languages** **or RE languages**

represents **class of problems that TMs can solve**

Equivalence of Different Models

Q. What do we mean by - "DFAs and NFAs are equivalent" ?

A. A language is accepted by some DFA if and only if it is accepted by some NFA

$$\forall \text{ DFA } D, \exists \text{ some NFA } N \text{ s.t. } L(D) = L(N)$$
$$\text{and... } \forall \text{ NFA } N, \exists \text{ some DFA } D \text{ s.t. } L(D) = L(N)$$

In this sense DFA, NFA, Regular Expression, Regular Grammar are all equivalent in power.

Set of all languages accepted by DFAs
= Set of all languages accepted by NFAs
= Set of all languages expressible as Regular Expression
= Set of all languages accepted by Regular Grammar

This set of languages is called the **class of Regular Languages**

Since each language corresponds to a computational problem,
the class of regular languages represents
the **class of problems that can be solved using finite automata**.

Different Models of Computation

Strictly more powerful

Equivalent

Deterministic Finite Automata, Nondeterministic Finite Automata , Regular Expression, Regular Grammar
or DFA or NFA

Equivalent

Deterministic Pushdown Automata , Deterministic Context Free Grammar
or DPDA

Equivalent

(Nondeterministic) Pushdown Automata, (Nondeterministic) Context Free Grammar
or PDA or CFG

Equivalent

Linear Bounded Automata, Context Sensitive Grammar
or LBA or CSG

Equivalent

Deterministic Turing Machine , Nondeterministic Turing Machine, Unrestricted Grammar
or DTM or NTM