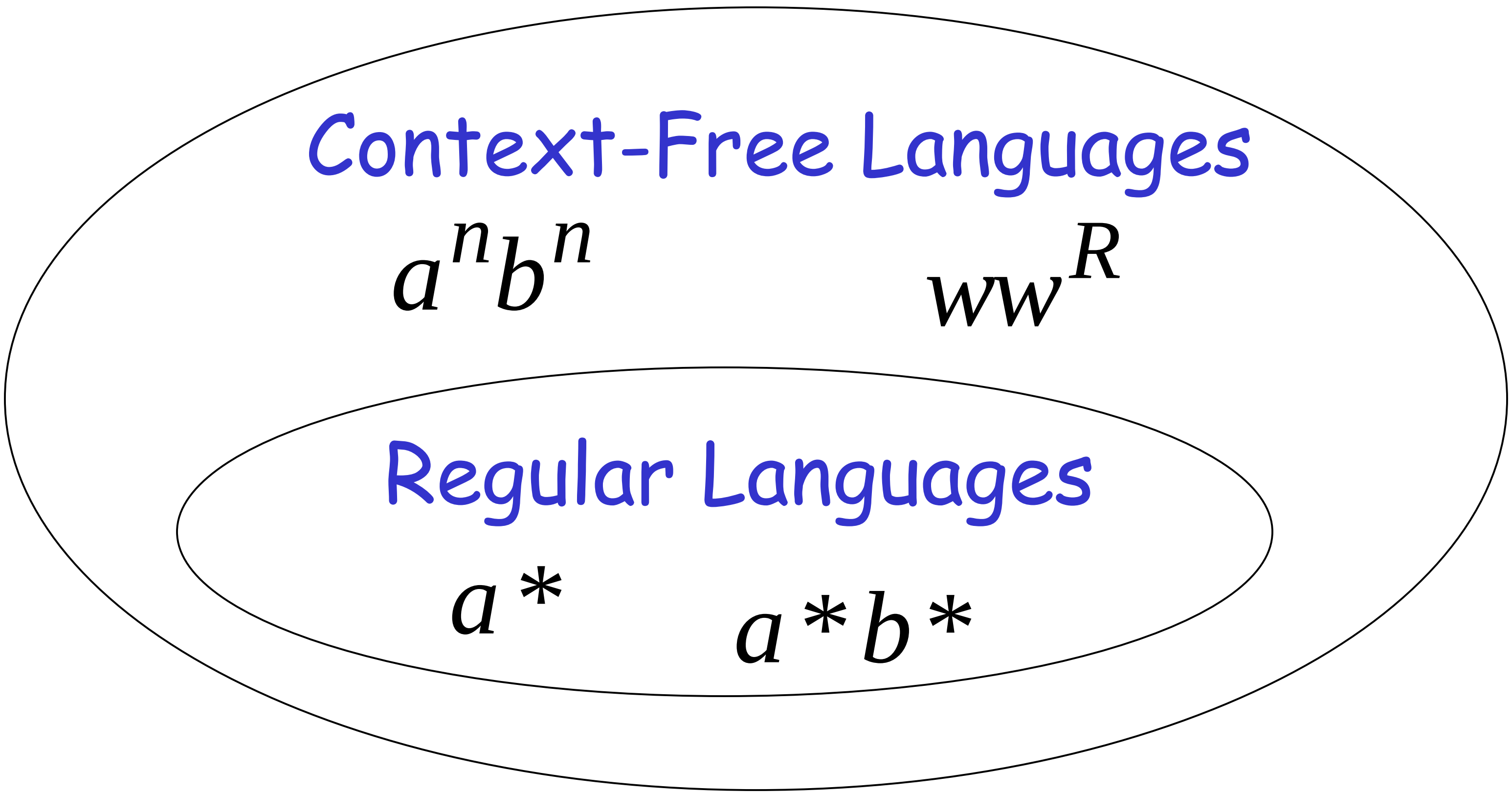
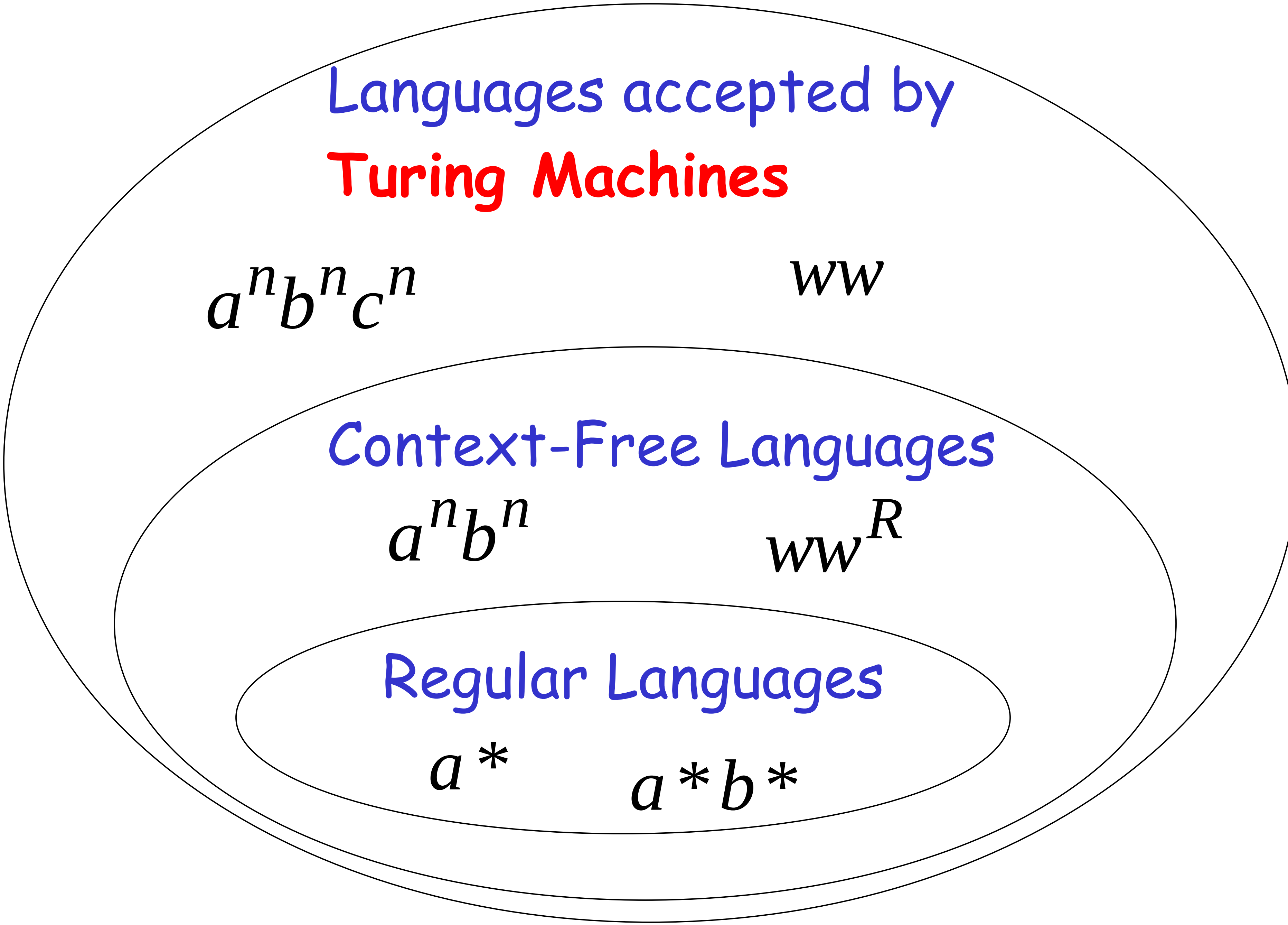
Raju Halder

Assistant Professor
Department of Comp. Sc. & Engg.
Indian Institute of technology Patna
halder@iitp.ac.in

The Language Hierarchy

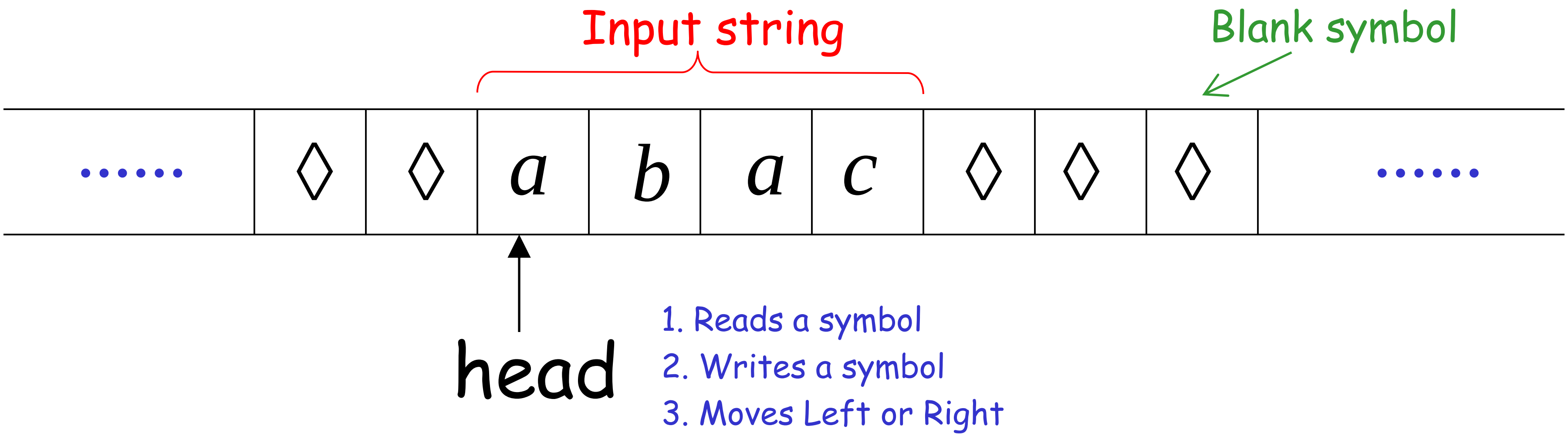
$a^n b^n c^n$ **?** ww **?**

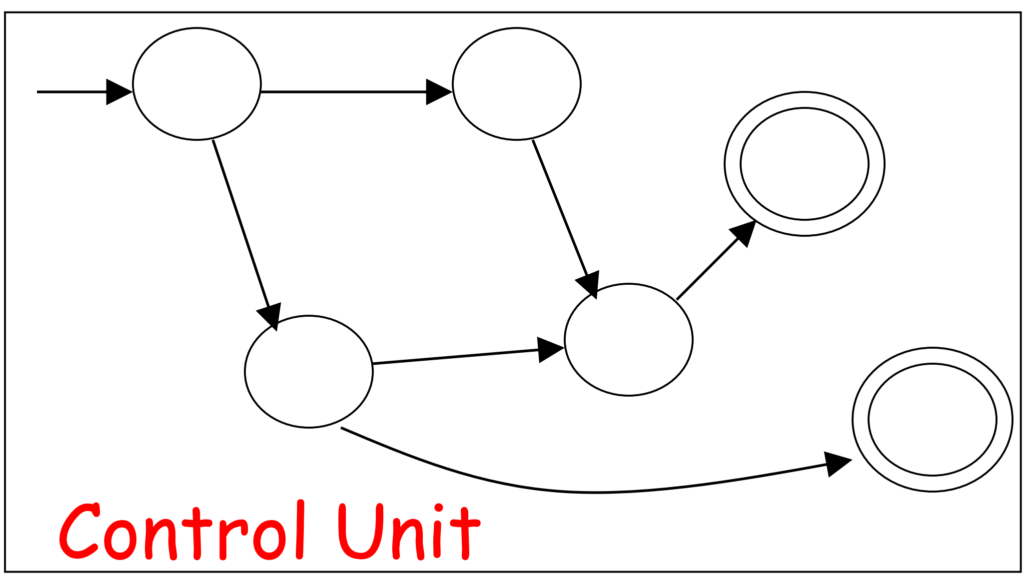


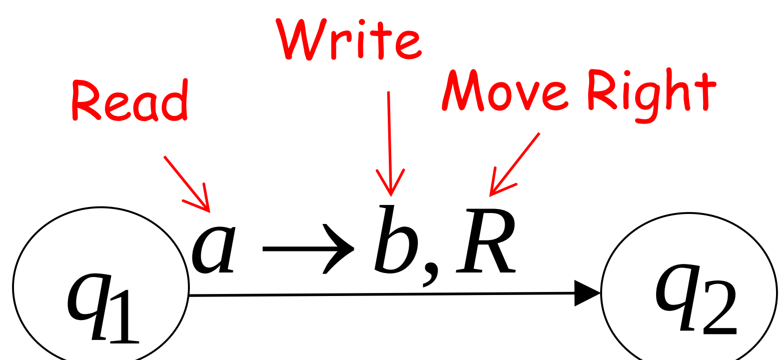

Languages accepted by
Turing Machines
$a^n b^n c^n$
ww
Context-Free Languages
$a^n b^n$
ww^R
Regular Languages
a^*
$a^* b^*$

A Turing Machine

infinite length Tape







Acceptance

Accept Input string → If machine halts in an accept state

Reject Input string → If machine halts in a non-accept state **or** If machine enters an *infinite loop*

Decidable Language:

A language L is **decidable** if there is a Turing machine M (decider) which accepts L and halts on every input string.

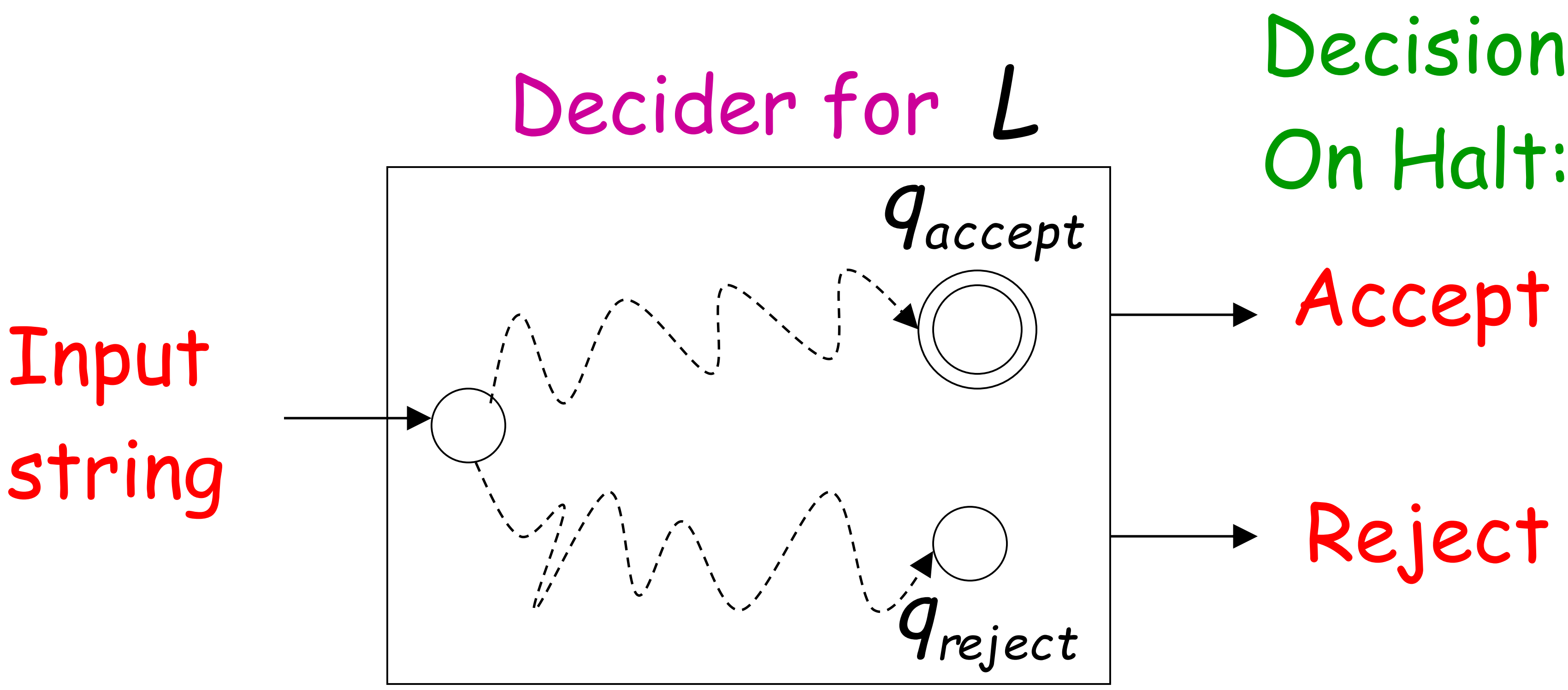
For any string w :

$w \in L \Longrightarrow M$ halts in an accept state

$w \notin L \Longrightarrow M$ halts in a non-accept state

L is Also known as *recursive languages.*

For a decidable language L :



For each input string, the computation halts in the accept or reject state

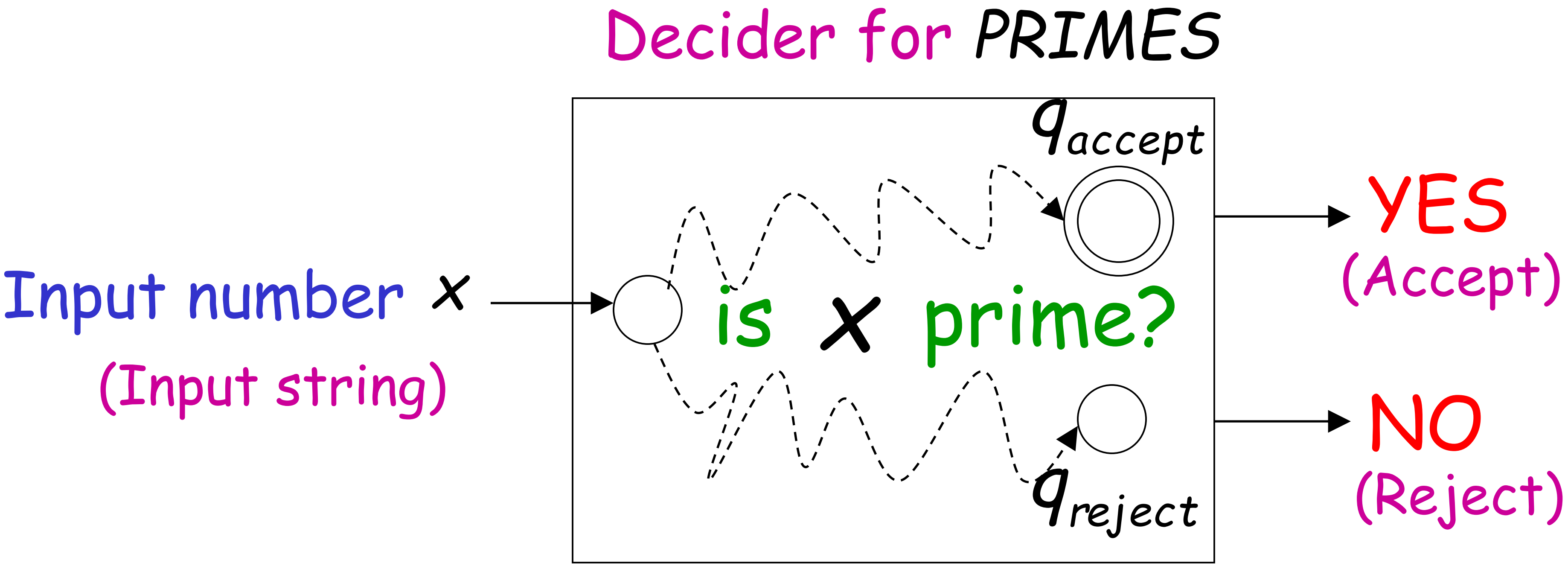
Problem: Is number x prime?

Corresponding language:

$$PRIMES = \{1, 2, 3, 5, 7, \ldots\}$$

We will show it is decidable

Can we have such a Decider?



Decider for *PRIMES* :

On input number x :

Divide x with all possible numbers between 2 and $\sqrt{x}$

If any of them divides x

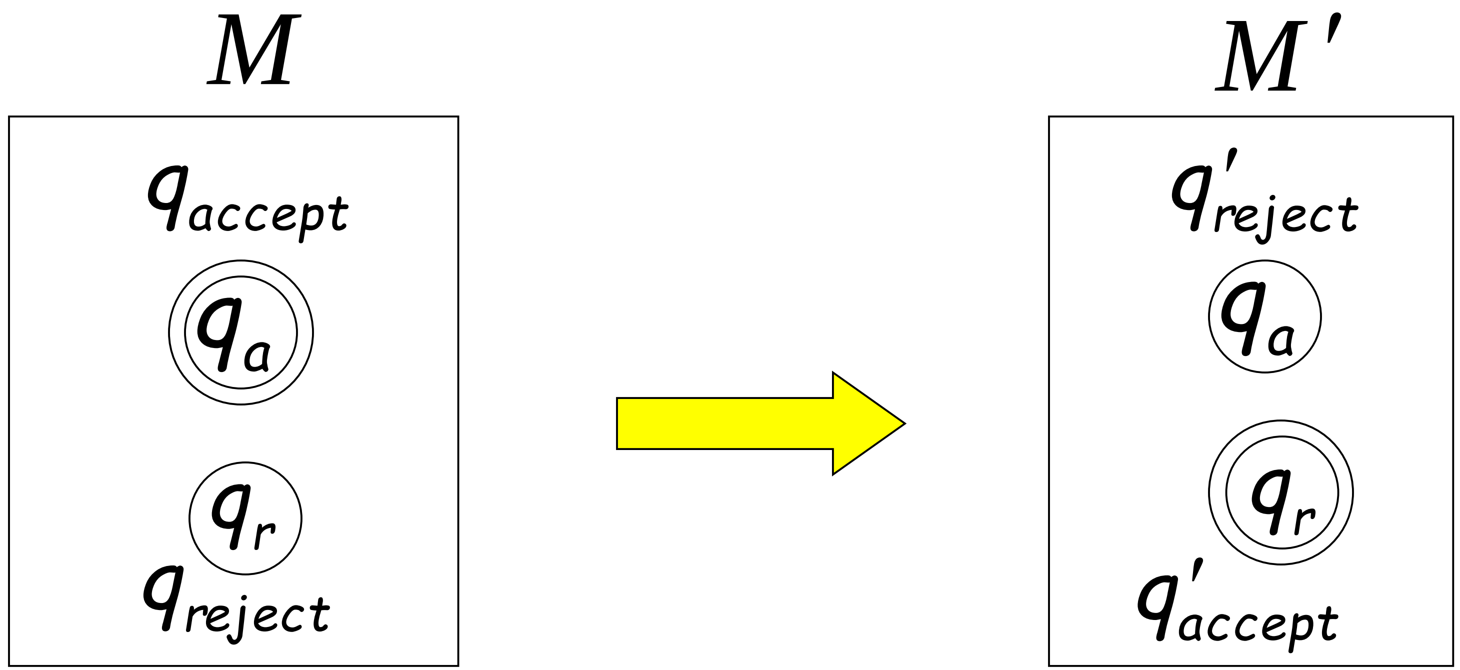
<u>Then</u> reject

<u>Else</u> accept

Theorem:

If a language L is decidable, then its complement $\overline{L}$ is decidable too

Proof:



Build a TM M' that accepts $\overline{L}$ and halts on every input string.

A computational problem is decidable
if the corresponding language is decidable

We also say that the problem is solvable

Undecidable Languages

A language L is **Turing-Acceptable** if there is a Turing machine M that accepts L

Also known as: *Turing-Recognizable*

or

Recursively-enumerable

languages

For any string w :

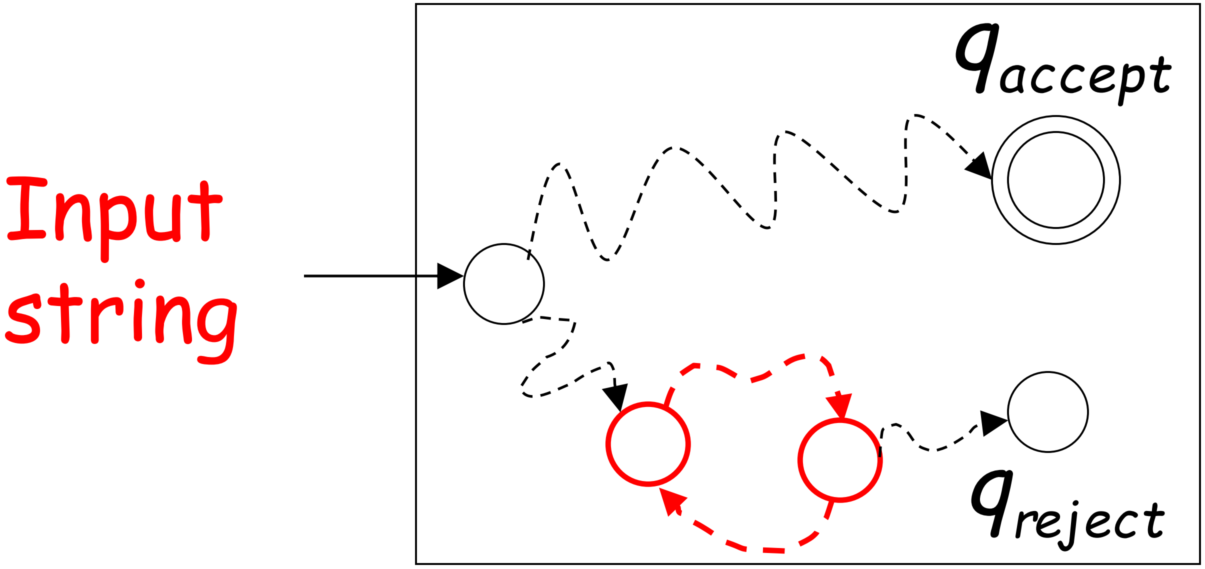
$w \in L \Rightarrow M$ halts in an accept state

$w \notin L \Rightarrow M$ halts in a non-accept state

or loops forever

For a Turing-Acceptable language L :

Turing Machine for L



It is possible that for some input string the machine enters an infinite loop

Theorem 1:
If M Decides L then M Recognizes L.

But not necessarily vice versa.

Theorem 2:
If L is Decidable then L is Turing-Recognizable.

But not necessarily vice versa.

Theorem 3:

If L is Decidable then $\overline{L}$ is Decidable.

Theorem 4:

L is Decidable if and only if L and $\overline{L}$ both Turing-Recognizable.

Four possibilities:

- L and $\overline{L}$ both Turing-recognizable.
  - Equivalently, L is Turing-decidable.
- L is Turing-recognizable, $\overline{L}$ is not.
- $\overline{L}$ is Turing-recognizable, L is not.
- Neither L nor $\overline{L}$ is Turing-recognizable.

Cardinality Argument/Cantor's diagonal argument
(countable TM/uncountable L)

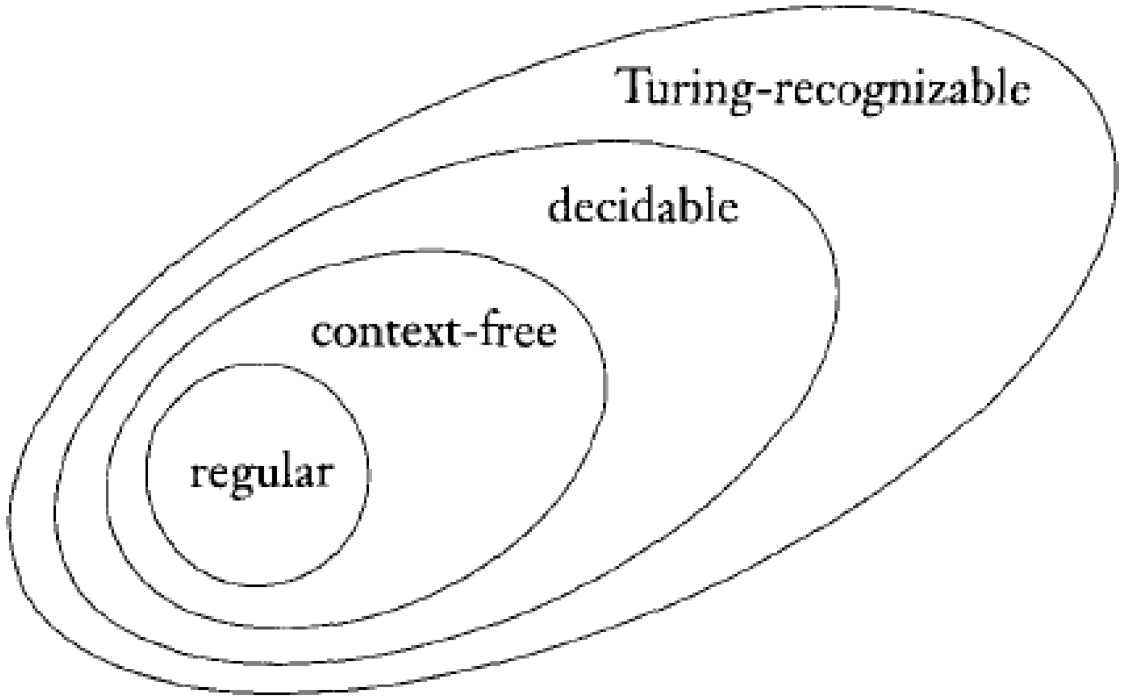


FIGURE 4.10
The relationship among classes of languages

Undecidable Languages

undecidable language = not decidable language

There is no decider:

there is no Turing Machine
which accepts the language
and makes a decision (halts)
for every input string

(machine may make decision for some input strings)

For an undecidable language, the corresponding problem is undecidable (unsolvable):

there is no Turing Machine (Algorithm) that gives an answer (yes or no) for every input instance

(answer may be given for some input instances)

We will prove that two particular problems are unsolvable:

Halting problem

Membership problem

Halting Problem

Input:
- Turing Machine M
- String w

Question: Does M halt while processing input string w ?

Corresponding language:

$$HALT_{TM} = \{\langle M, w \rangle : M \text{ is a Turing machine that halts on input string } w\}$$

Theorem: $HALT_{TM}$ is undecidable

(The halting problem is unsolvable)

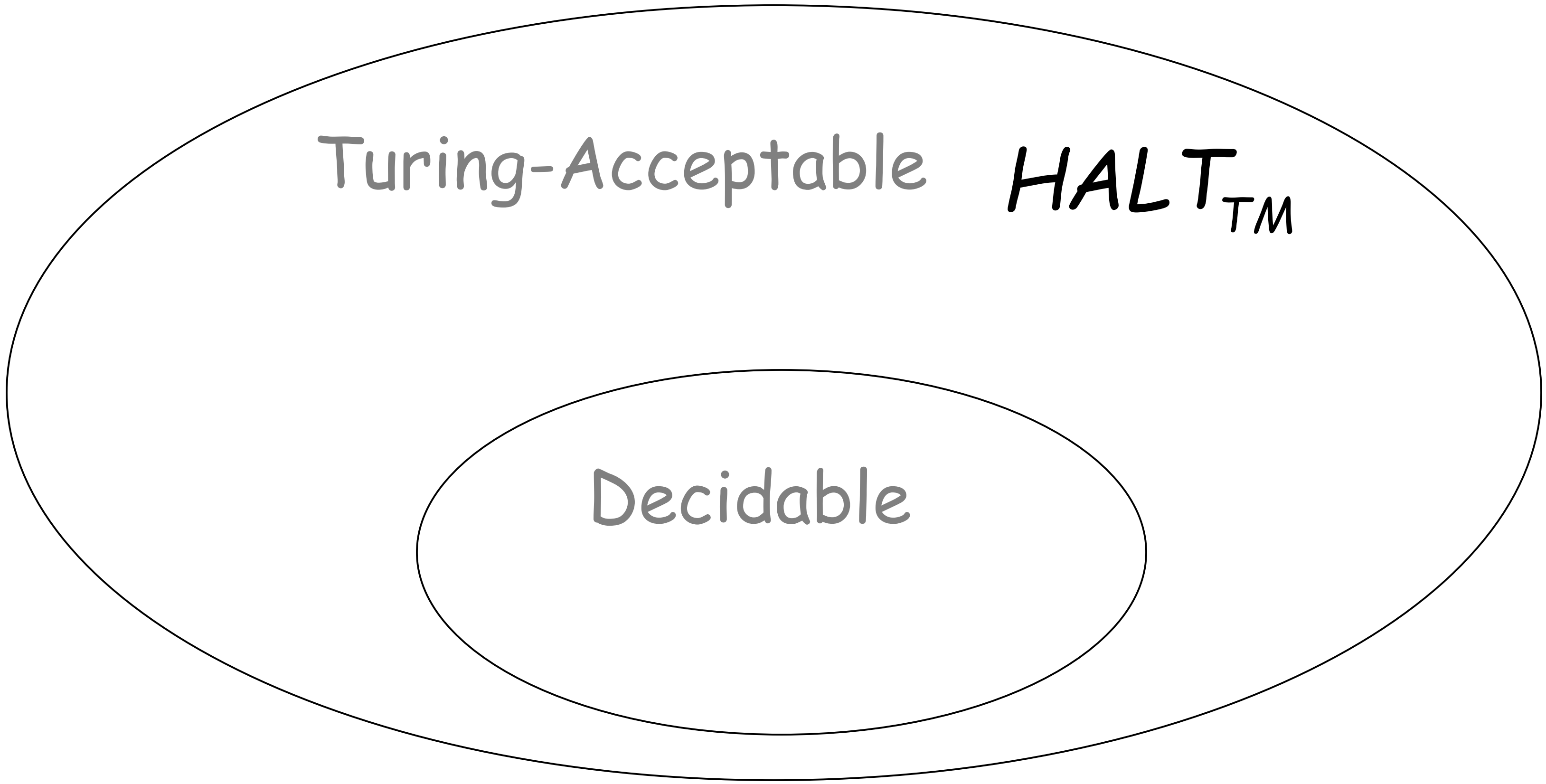
Proof:

Basic idea:

Suppose that $HALT_{TM}$ is decidable;
we will prove that
every decidable language
is also Turing-Acceptable

A contradiction!

We can actually show:



Membership Problem

Input: • Turing Machine M
• String w

Question: Does M accept w ?

$$w \in L(M)?$$

Corresponding language:

$$A_{TM} = \{\langle M, w\rangle : M \text{ is a Turing machine that accepts string } w\}$$

Theorem: A_{TM} is undecidable

(The membership problem is unsolvable)

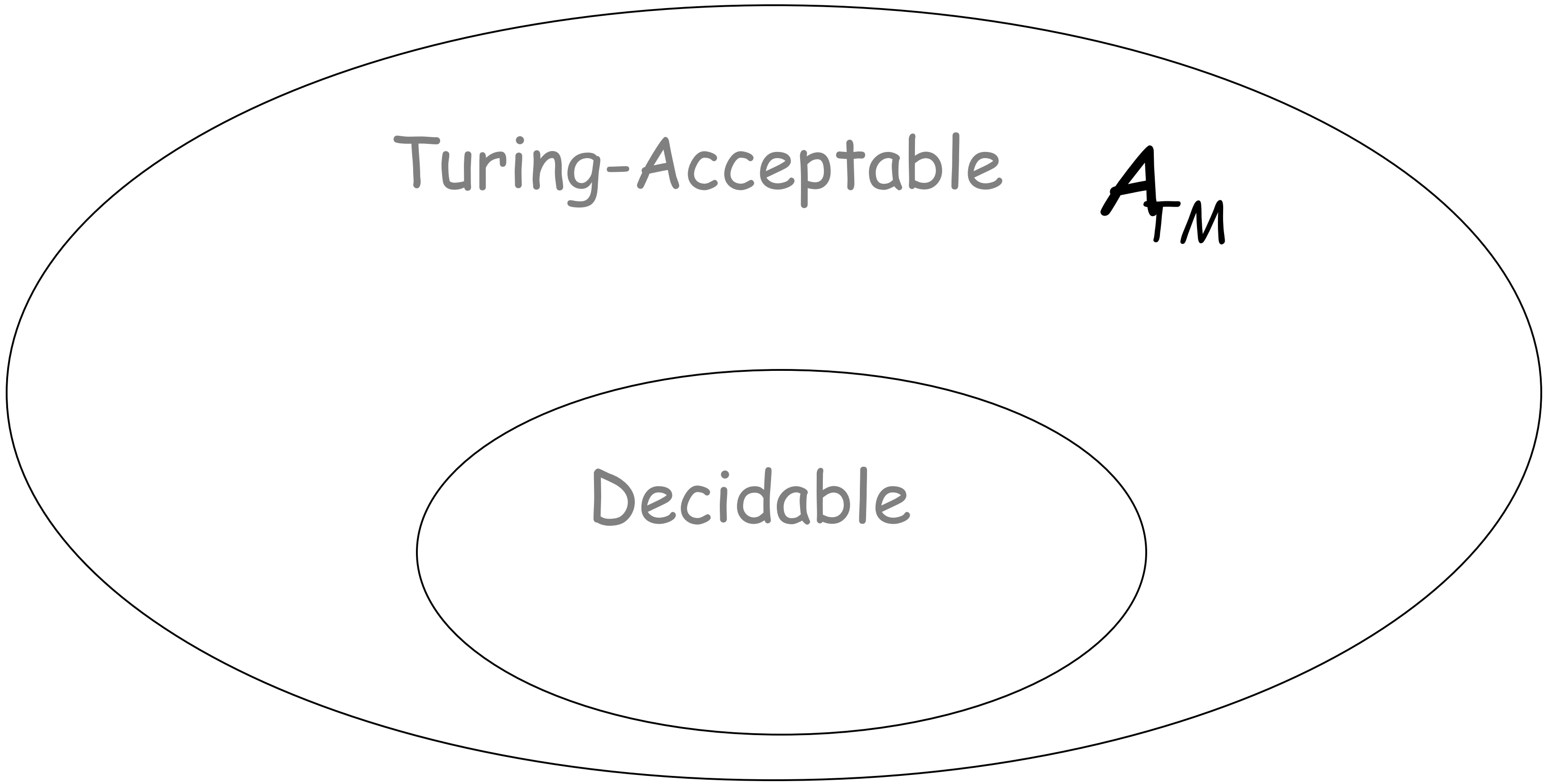
Proof:

Basic idea:

We will assume that A_{TM} is decidable;
We will then prove that
every decidable language
is Turing-Acceptable

A contradiction!

We can actually show:

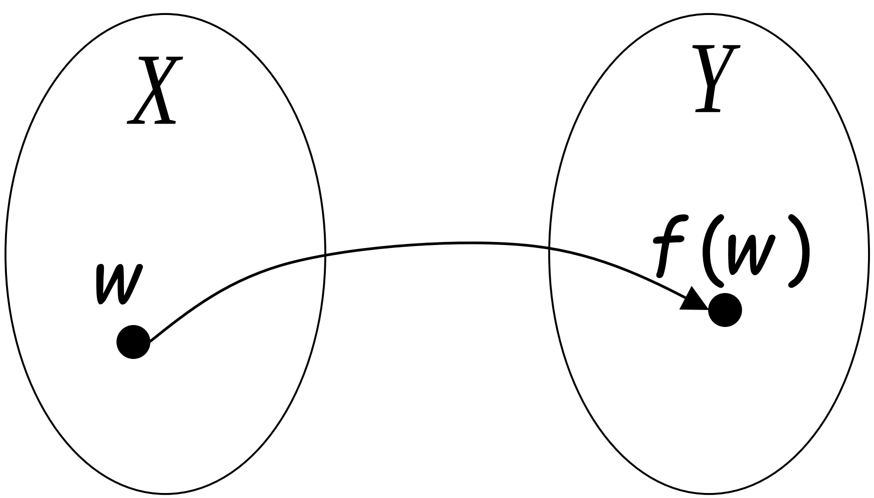


Reductions

Problem X is reduced to problem Y

(a) If Y is decidable then X is also decidable

(b) If X is undecidable, then Y is also undecidable



Rice Theoem

Non-trivial property:

A property P possessed by some Turing-acceptable languages but not all

Example: P_1 : L is empty?

YES $L = \emptyset$

NO $L = \{troy\}$

NO $L = \{troy, albany\}$

Trivial property:

A property P possessed by ALL Turing-acceptable languages

Examples: P_4 : L has size at least 0?

True for all languages

P_5 : L is accepted by some Turing machine?

True for all Turing-acceptable languages

Let L be a Turing-acceptable language

- L has size 2?

$$SIZE2_{TM} = \{\langle M \rangle : M \text{ is a Turing machine that accepts exactly two strings}\}$$

This can be generalized to all non-trivial properties of Turing-acceptable languages

$$PROPERTY_{TM} = \{\langle M \rangle : M \text{ is a Turing machine such that } L(M) \text{ has the non-trivial property } P, \text{ that is, } L(M) \in P\}$$

Rice's Theorem: $PROPERTY_{TM}$ is undecidable

(the non-trivial property problem is unsolvable)

Proof: Reduce A_{TM} (membership problem) to $PROPERTY_{TM}$ or $\overline{PROPERTY_{TM}}$

Time Complexity

Common Terminology for the Complexity of Algorithms

TABLE 1 Commonly Used Terminology for the Complexity of Algorithms.

Complexity	*Terminology*
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n \log n)$	$n \log n$ complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta(b^n)$, where $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity

Computer Time Examples

- Assume that time = 1 ns (10^{-9} second) per operation, problem size = n bits, and #ops is a function of n.

#ops(n)	$n = 10$ (1.25 bytes)	$n = 10^6$ (125 kB)
$\log_2 n$	3.3 ns	19.9 ns
n	10 ns	1 ms
$n \log_2 n$	33 ns	19.9 ms
n^2	100 ns	16 m 40 s
2^n	1.024 μs	$10^{301,004.5}$
$n!$	3.63 ms	Ouch!

The Computer Time Used by Algorthms

Time=1 ns

Asterisk: 10^100 years

TABLE 2 The Computer Time Used by Algorithms.

Problem Size	*Bit Operations Used*					
n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-11} s	10^{-10} s	3×10^{-10} s	10^{-9} s	10^{-8} s	3×10^{-7} s
10^2	7×10^{-11} s	10^{-9} s	7×10^{-9} s	10^{-7} s	4×10^{11} yr	*
10^3	1.0×10^{-10} s	10^{-8} s	1×10^{-7} s	10^{-5} s	*	*
10^4	1.3×10^{-10} s	10^{-7} s	1×10^{-6} s	10^{-3} s	*	*
10^5	1.7×10^{-10} s	10^{-6} s	2×10^{-5} s	0.1 s	*	*
10^6	2×10^{-10} s	10^{-5} s	2×10^{-4} s	0.17 min	*	*

Tractable Vs. Non-Tractable

- A problem that is solvable using an algorithm with <u>at most polynomial time complexity</u> is called ***tractable*** (or *feasible*).
  P is the set of all tractable problems.

- A problem that cannot be solved using an algorithm with worst-case polynomial time complexity is called ***intractable*** (or *infeasible*).

Complexity Class P

$$\begin{aligned}
\text{CONNECTED} &:= \{\langle G \rangle \mid G \text{ is a connected undirected graph}\} \\
\text{BIPARTITE} &:= \{\langle G \rangle \mid G \text{ is an undirected bipartite graph}\} \\
\text{TRIANGLE-FREE} &:= \{\langle G \rangle \mid G \text{ is a triangle-free undirected graph}\}
\end{aligned}$$

$$\text{PATH} := \{\langle G, s, t \rangle \mid \text{ There is a path from vertex } s \text{ to vertex } t \text{ in a directed graph } G\}$$

$$\text{RELPRIME} := \{\langle x, y \rangle \mid \text{ The positive integers } x \text{ and } y \text{ are relatively prime}\}$$

Example: The Satisfiability Problem

Boolean expressions in
Conjunctive Normal Form:

$$t_1 \wedge t_2 \wedge t_3 \wedge \cdots \wedge t_k$$ clauses

$$t_i = x_1 \vee \bar{x}_2 \vee x_3 \vee \cdots \vee \bar{x}_p$$

Variables

Question: is the expression satisfiable?

Example: $(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3)$

Satisfiable: $x_1 = 0,\ \ x_2 = 1,\ \ x_3 = 1$

$$(\overline{x}_1 \vee x_2) \wedge (x_1 \vee x_3) = 1$$

Example: $(x_1 \vee x_2) \wedge \overline{x}_1 \wedge \overline{x}_2$

Not satisfiable

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

Exponential time complexity

Algorithm:

search exhaustively all the possible binary values of the variables

Example: The satisfiability problem

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

Non-Deterministic algorithm:

- Guess an assignment of the variables

- Check if this is a satisfying assignment

$$L = \{w : \text{expression } w \text{ is satisfiable}\}$$

Time for n variables:

- Guess an assignment of the variables $O(n)$
- Check if this is a satisfying assignment $O(n)$

Total time: $O(n)$

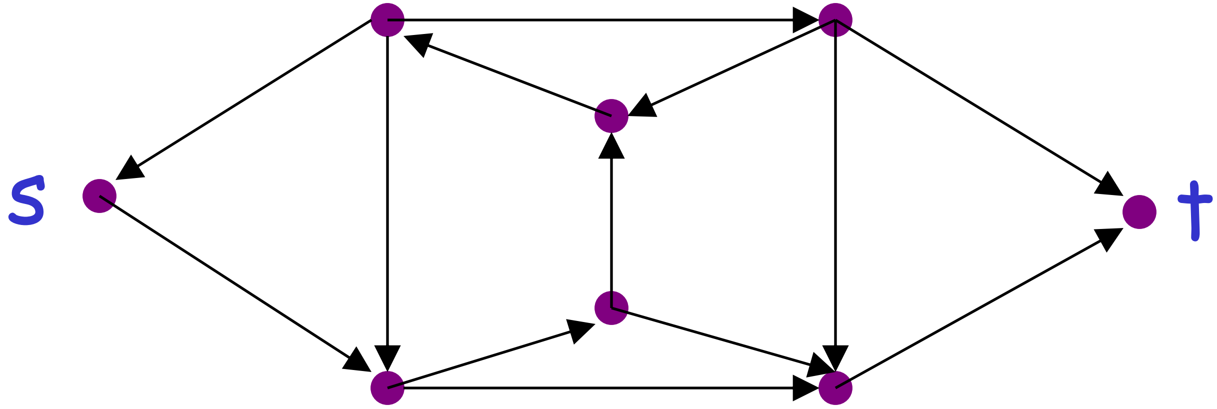
$$L = \{ w : \text{expression } w \text{ is satisfiable} \}$$

$$L \in NP$$

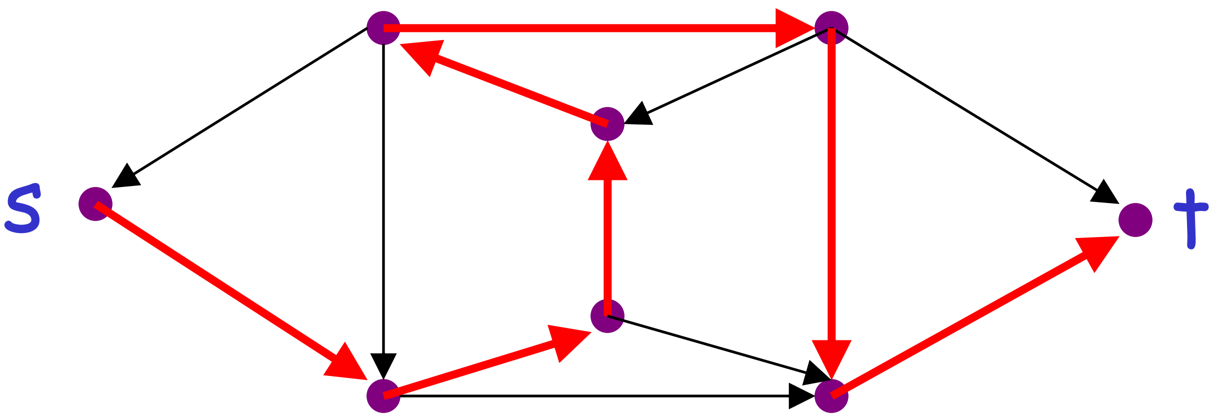
The satisfiability problem is an NP - Problem

- **NP** is the set of problems for which there exists a tractable algorithm for *checking a proposed solution* to tell if it is correct.

Example: the Hamiltonian Path Problem

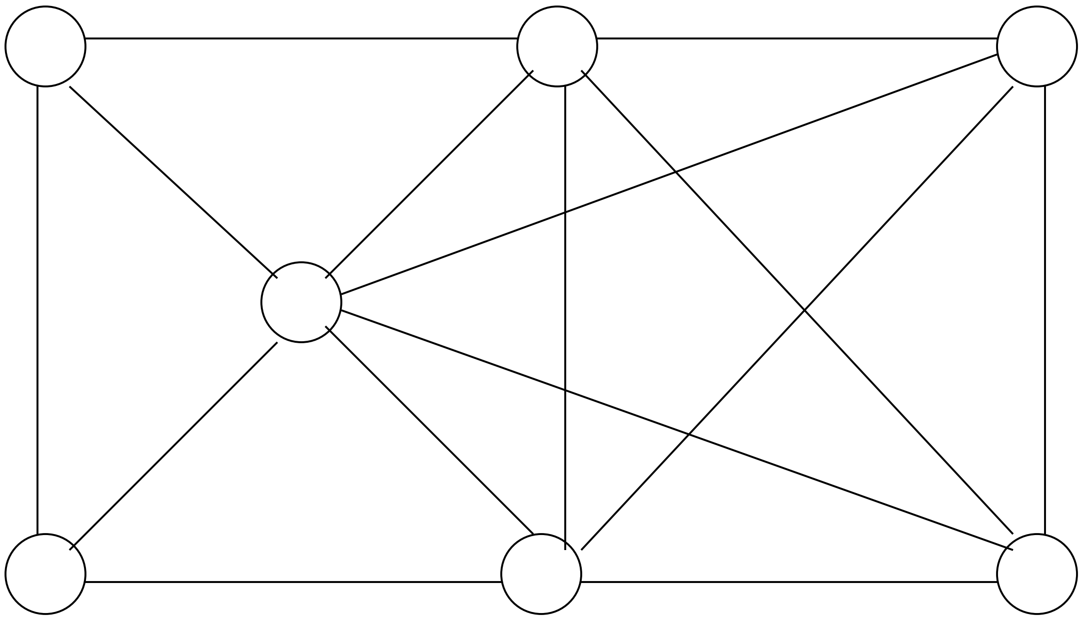


Question: is there a Hamiltonian path from s to t?



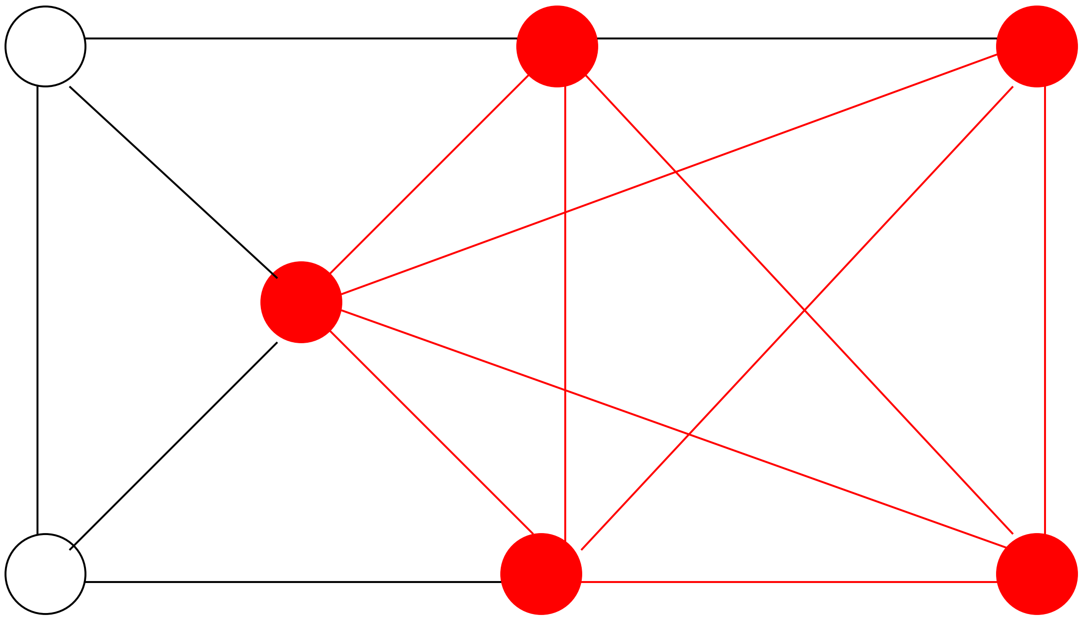
YES!

The clique problem



Does there exist a clique of size 5?

The clique problem



Does there exist a clique of size 5?

Observation: $P \subseteq NP$

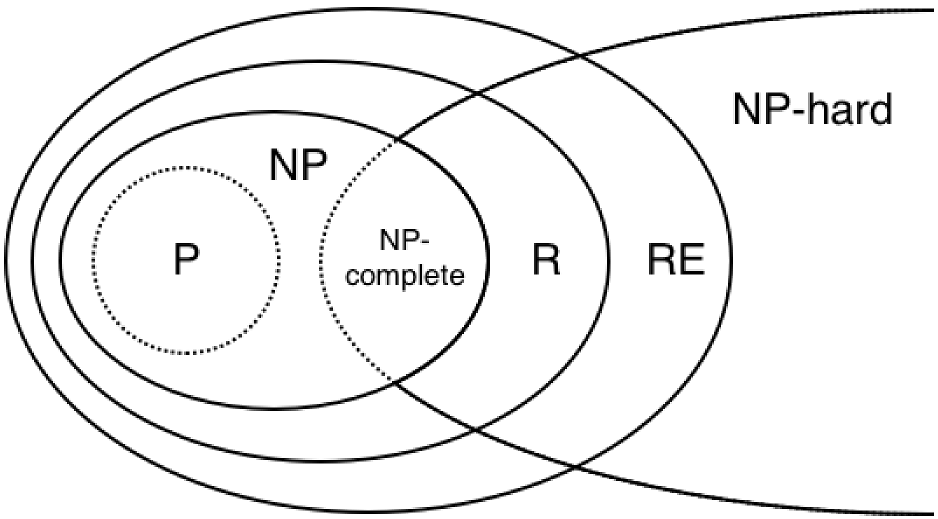
Deterministic Polynomial

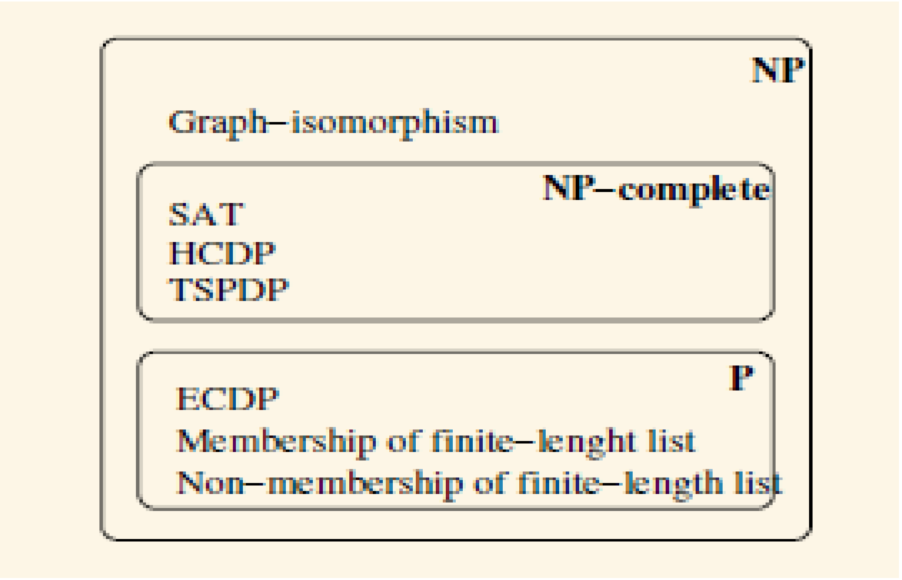
Non-Deterministic Polynomial

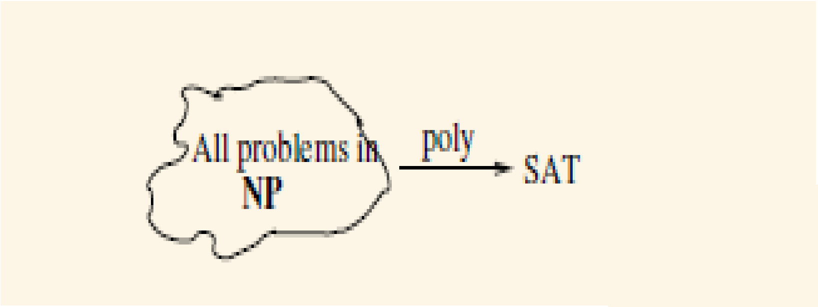
Open Problem: $P = NP$?

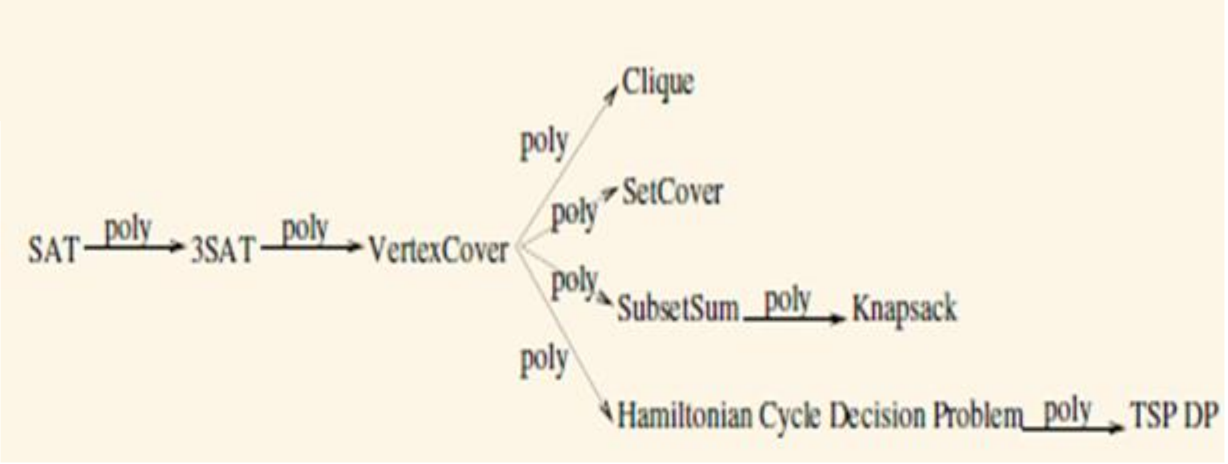
WE DO NOT KNOW THE ANSWER

P-NP-NPC-NPH


NP-hard
NP
P
NP-complete
R
RE


NP
Graph−isomorphism
NP−complete
SAT
HCDP
TSPDP
P
ECDP
Membership of finite−lenght list
Non−membership of finite−length list


All problems in NP
poly
SAT


SAT
poly
3SAT
poly
VertexCover
poly
Clique
poly
SetCover
poly
SubsetSum
poly
Knapsack
poly
Hamiltonian Cycle Decision Problem
poly
TSP DP

Thank You

halder@iitp.ac.in