SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

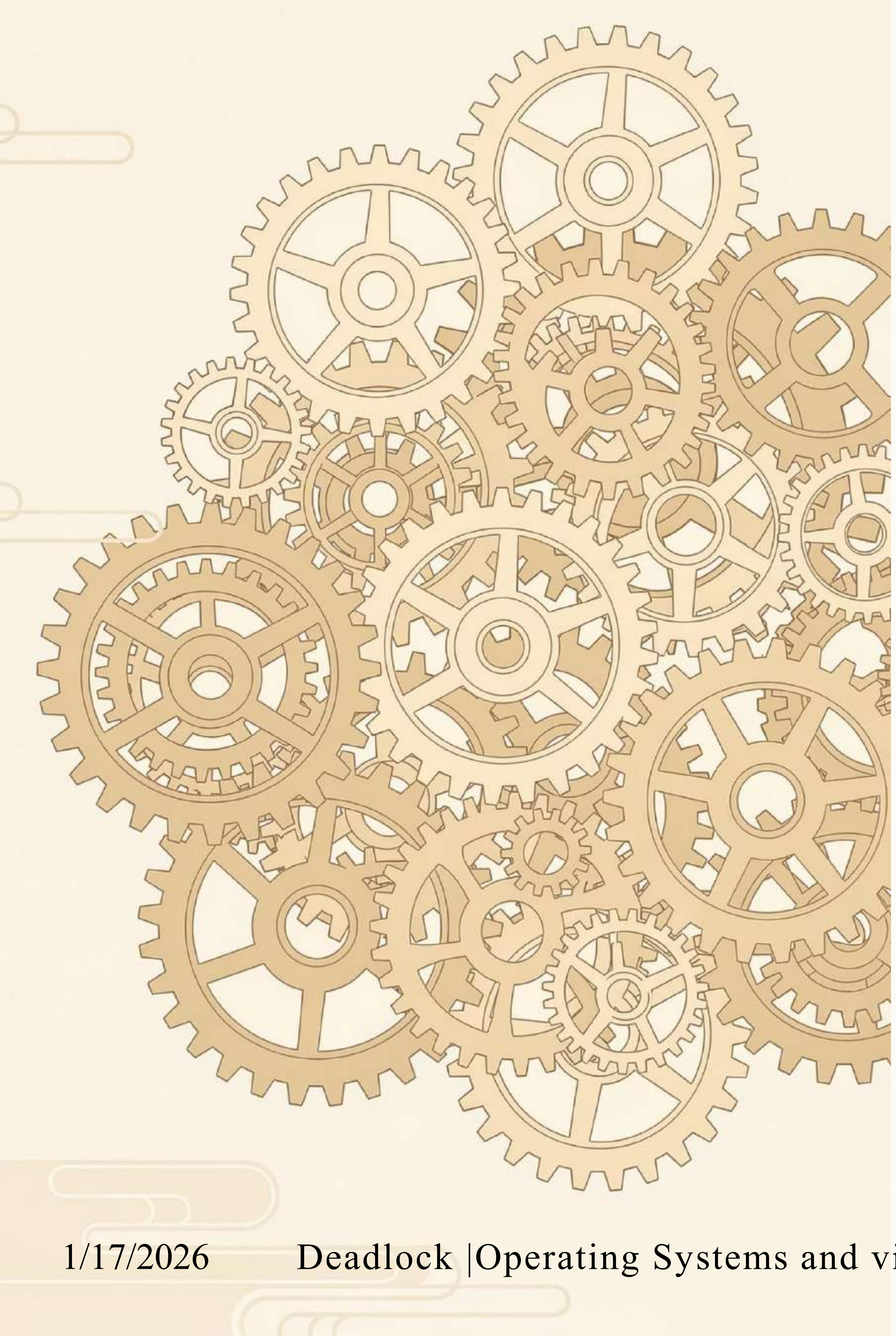
Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

UNIT – II PROCESS MANAGEMENT

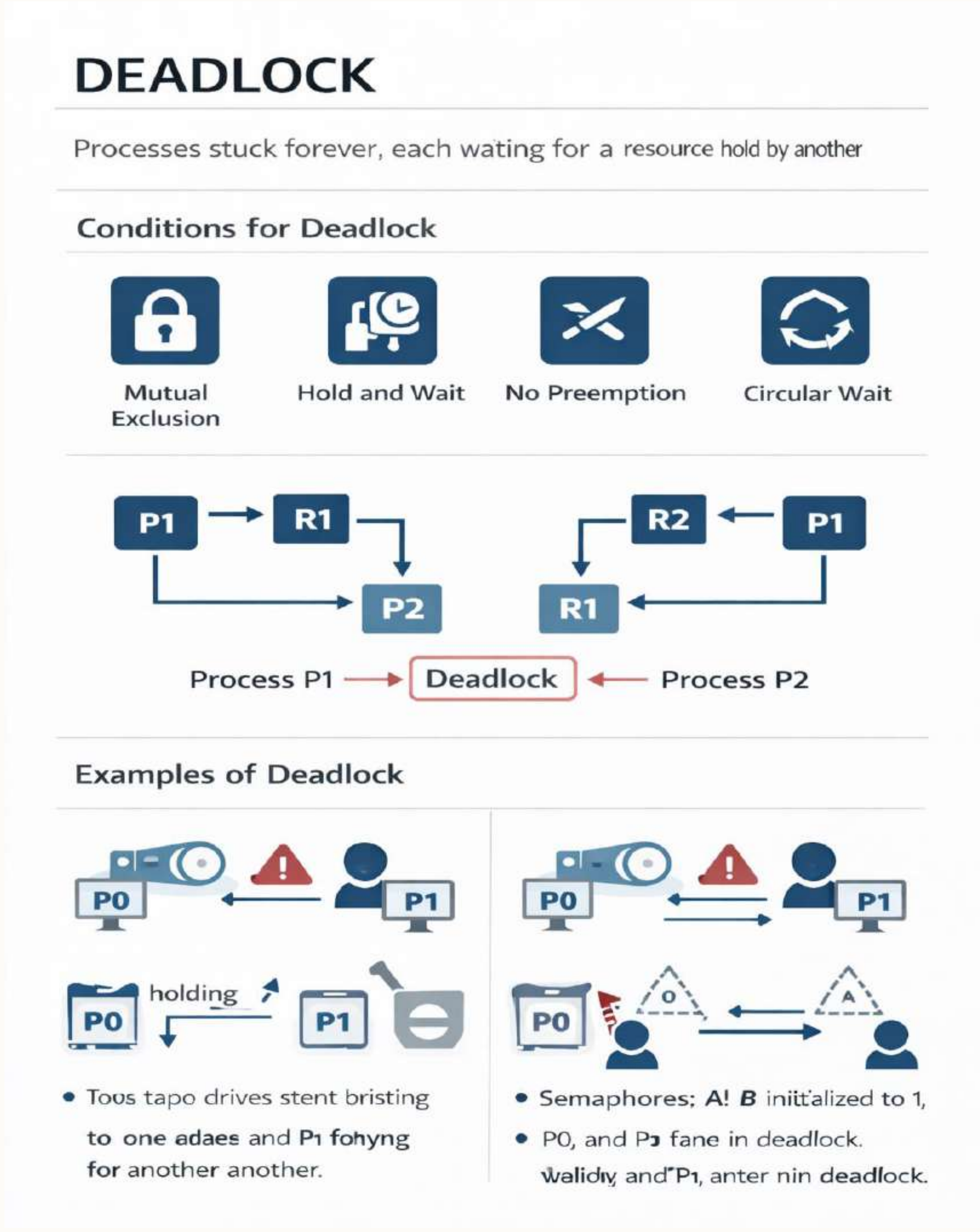
Topic 8: Deadlock



OPERATING SYSTEMS

Deadlock in Operating Systems

Understanding how to identify, prevent, and resolve system deadlocks in multi-process environments.



DEADLOCK

Processes stuck forever, each wating for a resource hold by another

Conditions for Deadlock

Examples of Deadlock

- Tous tapo drives stent bristing to one adaes and P1 fohyng for another another.

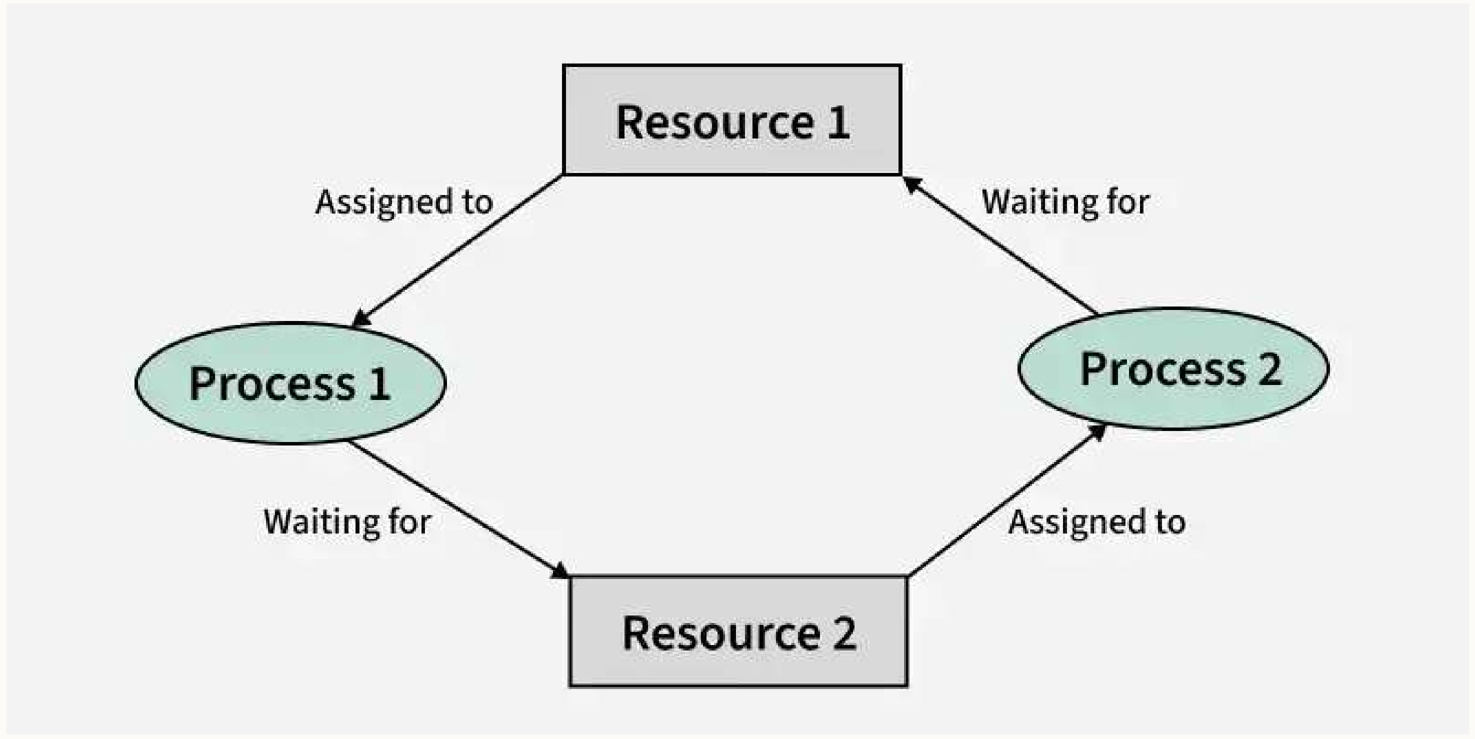
- Semaphores; A! B initalized to 1,
- P0, and P3 fane in deadlock. Walidiy and P1, anter nin deadlock.

What is Deadlock?

- Deadlock is a critical state in an operating system.
- Two or more processes become permanently stuck, each waiting for a resource held by another.
- No process in the affected set can proceed, freezing that portion of the system.

Example:

- Process P1 holds Resource R1 and requests R2.
- Process P2 holds Resource R2 and requests R1.
  Neither process can proceed causing a deadlock.



Real-World Deadlock Examples

1

Tape Drive Deadlock

The system has 2 tape drives. Process P0 and P1 each hold one tape drive and each needs the other one to complete their operation. Neither can proceed.

2

Semaphore Deadlock

Two semaphores A and B, both initialized to 1. P0 executes wait(A) then preempts. P1 executes wait(B). Now P0 waits for B while P1 waits for A — creating a circular dependency.

3

Memory Allocation Deadlock

System has 200KB available. P0 requests 80KB (granted), P1 requests 70KB (granted). Then P0 requests 60KB more and P1 requests 80KB more. Neither request can be satisfied deadlock occurs.

www.snsgroups.com

NECESSARY CONDITIONS FOR DEADLOCK IN OS

Deadlock requires these four conditions

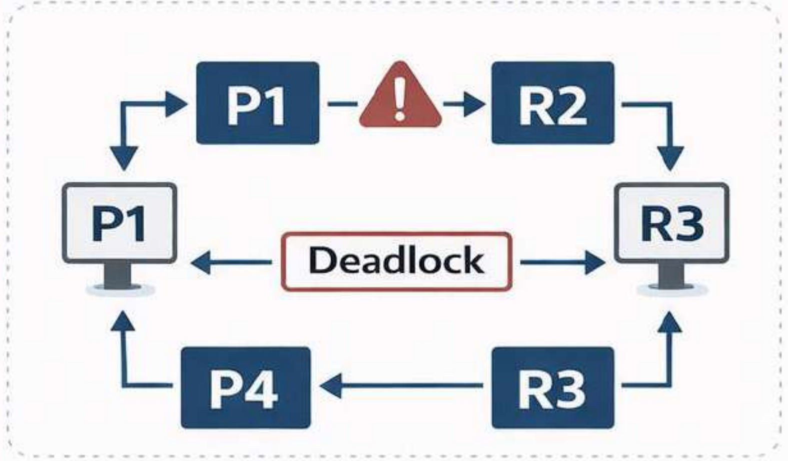
Mutual Exclusion
Only one process can use a resource at a time.

Hold and Wait
Holds at least one resource while requesting more.

No Preemption
Resources cannot be forcibly taken away.

Circular Wait
Processes form a circular chain, each waiting for the next.

Circular Wait Example



Circular Wait Example

- **P1** holds **R1**, and waits for **R2**,
- **P2** holds **R2**, and waits for **R3**,
- **P3** holds **R3**, and waits for **R4**.

Circular Wait Example

Circular Wait Illustrated

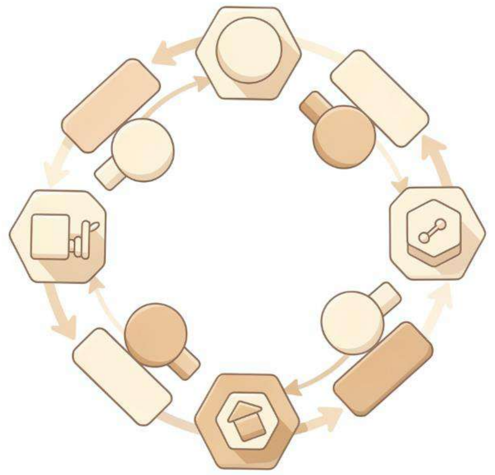
This creates an unbreakable cycle where no process can proceed.

P1 holds R1

Waiting for R2, part of the cycle

P2 holds R2

Waiting for R3, continuing cycle

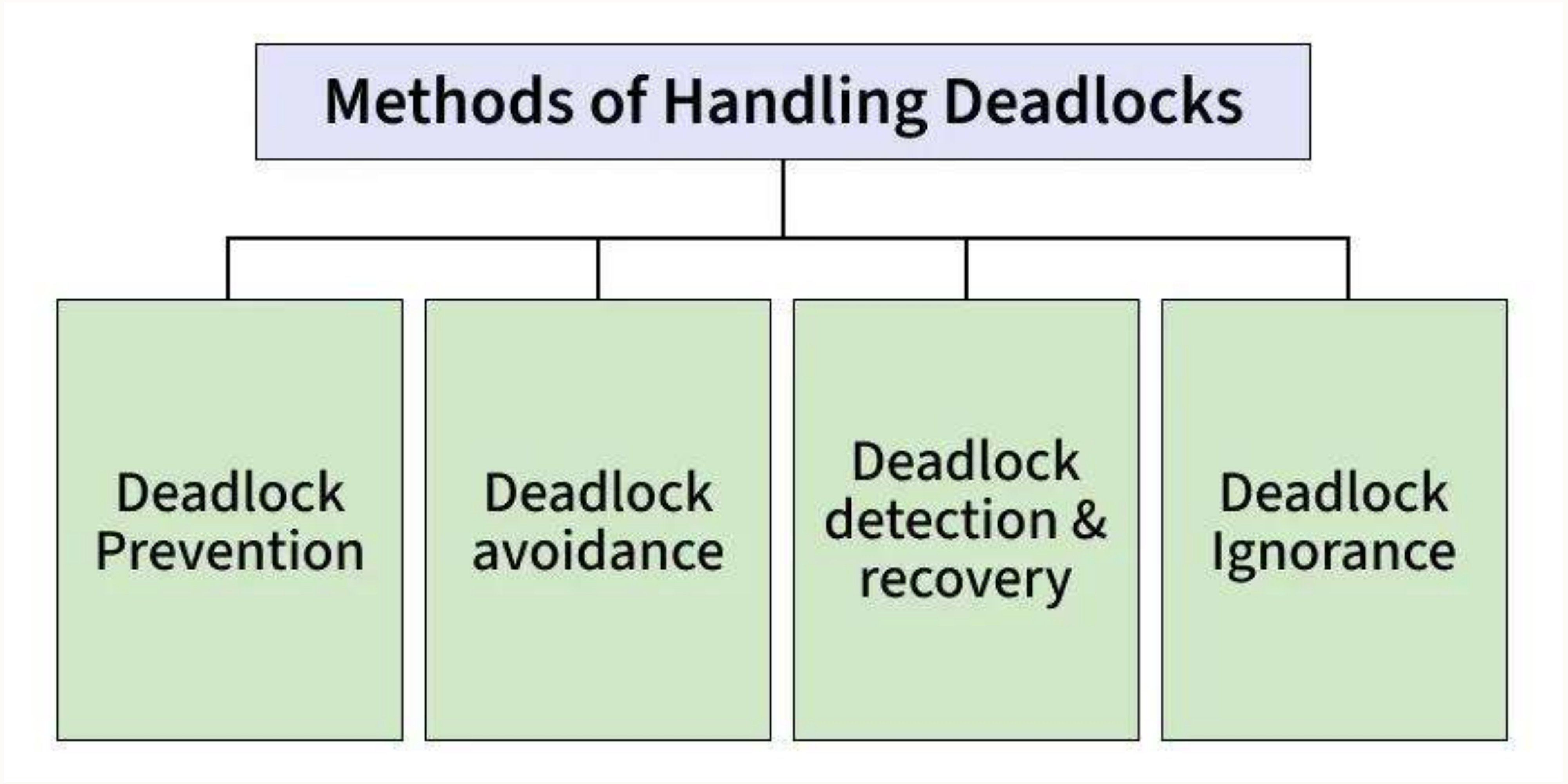


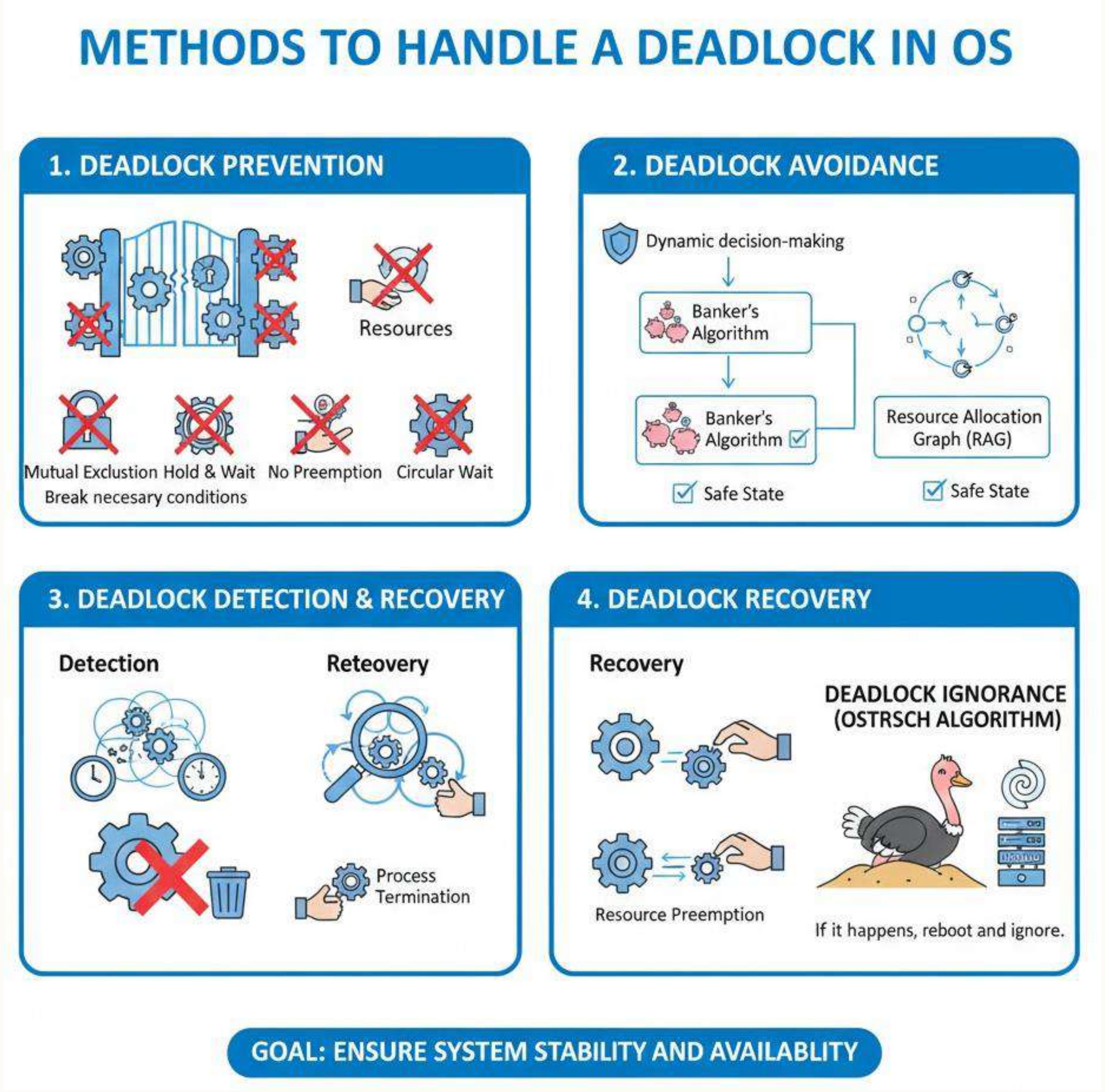
P3 holds R3

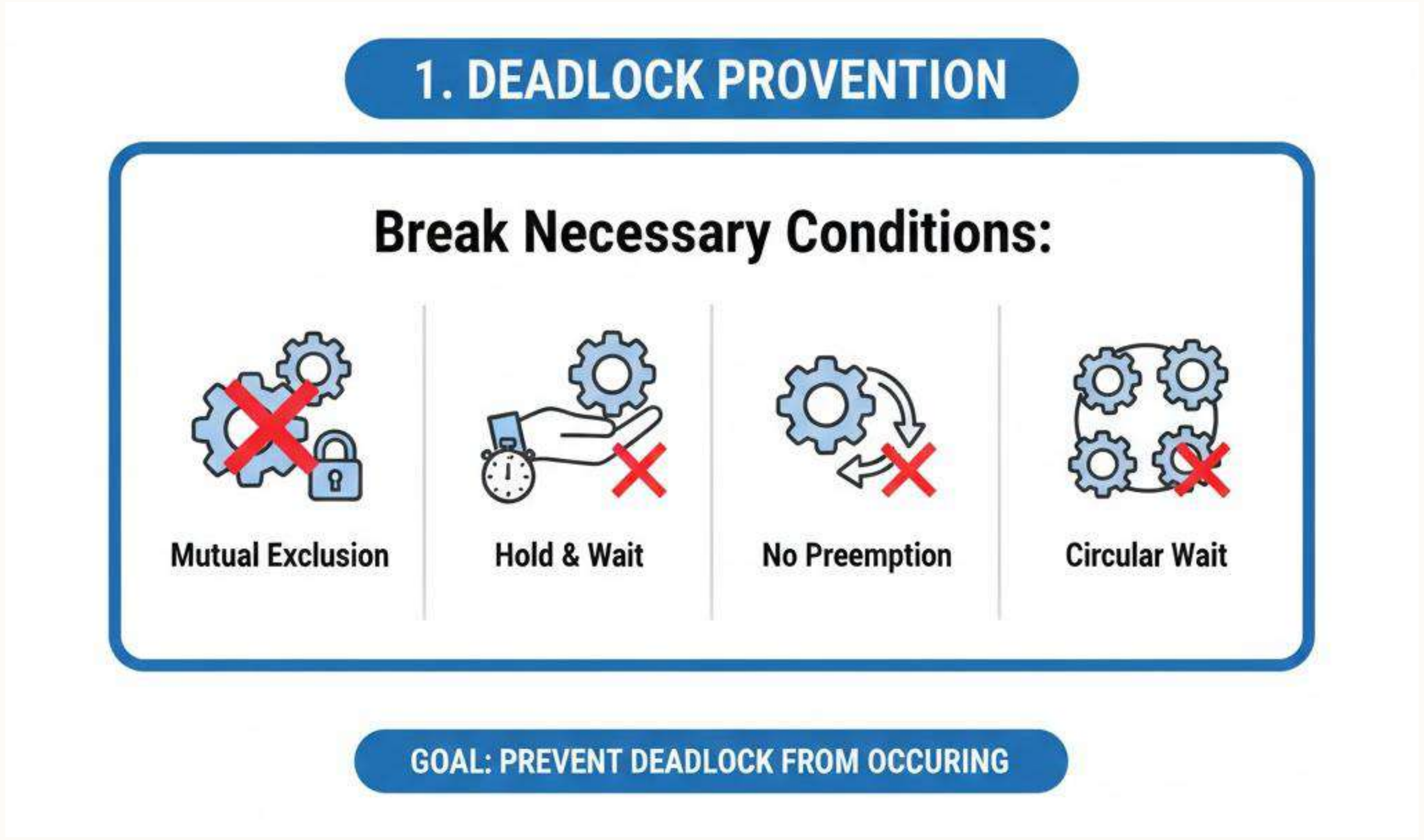
Waiting for R4, cycle persists

P4 holds R4

Waiting for R1, completes cycle


Methods of Handling Deadlocks
Deadlock Prevention
Deadlock avoidance
Deadlock detection & recovery
Deadlock Ignorance


METHODS TO HANDLE A DEADLOCK IN OS
1. DEADLOCK PREVENTION
Resources
Mutual Exclustion
Hold & Wait
No Preemption
Circular Wait
Break necesary conditions
2. DEADLOCK AVOIDANCE
Dynamic decision-making
Banker's Algorithm
Banker's Algorithm
Resource Allocation Graph (RAG)
Safe State
Safe State
3. DEADLOCK DETECTION & RECOVERY
Detection
Reteovery
Process Termination
4. DEADLOCK RECOVERY
Recovery
Resource Preemption
DEADLOCK IGNORANCE (OSTRSCH ALGORITHM)
If it happens, reboot and ignore.
GOAL: ENSURE SYSTEM STABILITY AND AVAILABLITY


1. DEADLOCK PROVENTION
Break Necessary Conditions:
Mutual Exclusion
Hold & Wait
No Preemption
Circular Wait
GOAL: PREVENT DEADLOCK FROM OCCURING

DEADLOCK AVOIDANCE

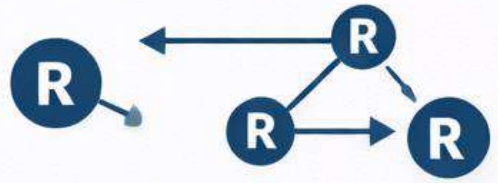
Dynamic decision-making ensures the system stays safe

Deadlock avoidance uses algorithms to check resource allocation for safety.

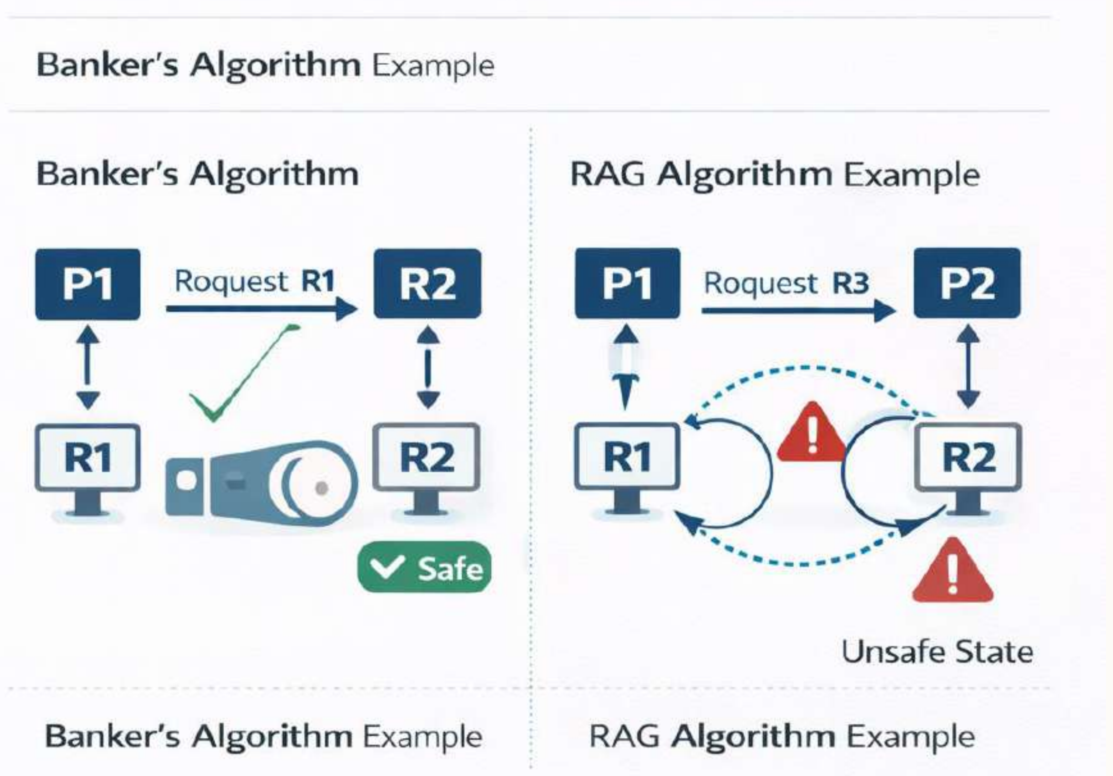
Banker's Algorithm

- Multiple resource instances
- Simulates safe allocations

Resource Allocation Graph (RAG) Algorithm



- Single resource instance
- Detects unsafe cycles



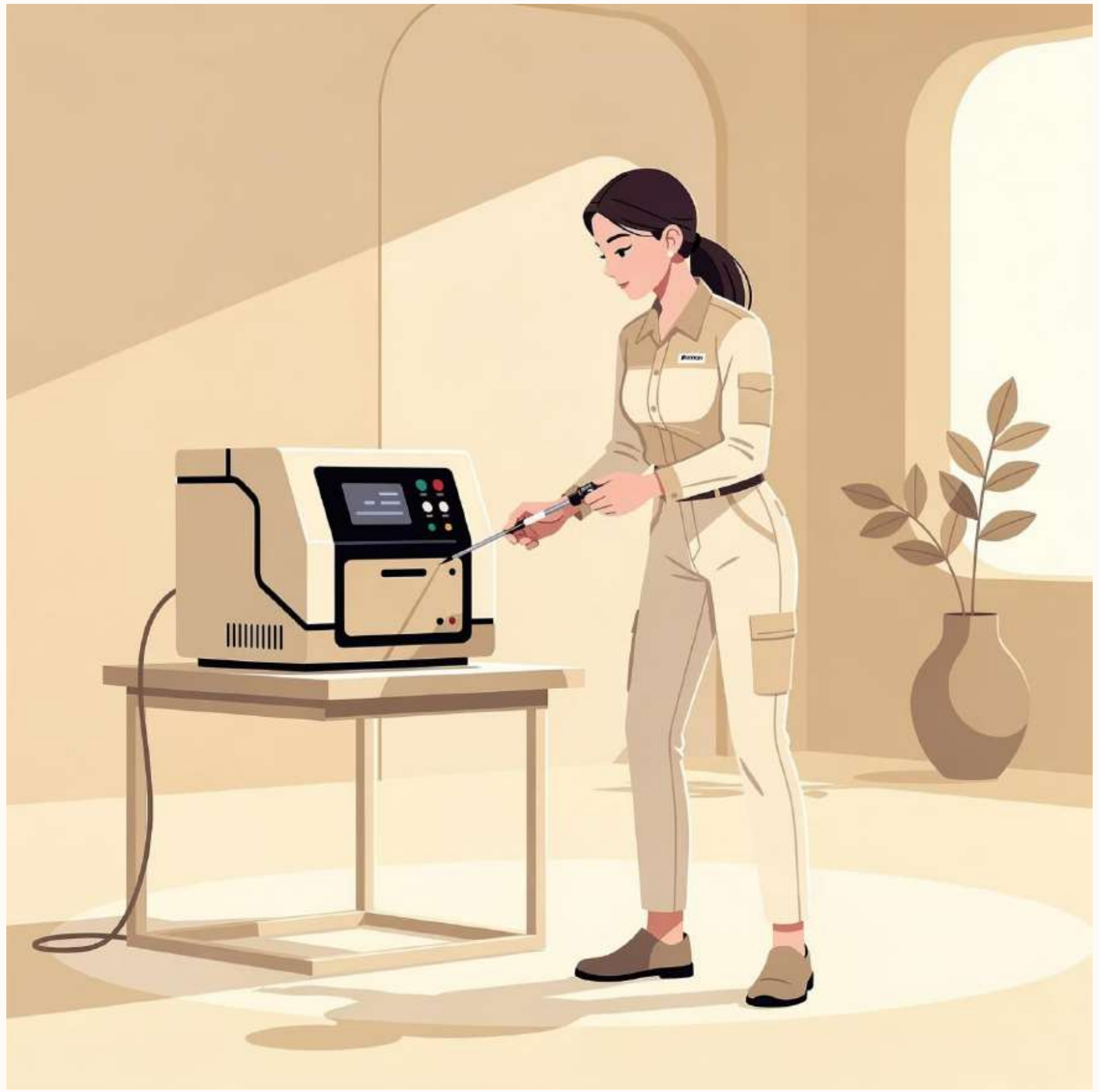
What Is Deadlock Recovery?

Deadlock recovery refers to the corrective techniques used by an operating system to eliminate an existing deadlock after it has been detected.

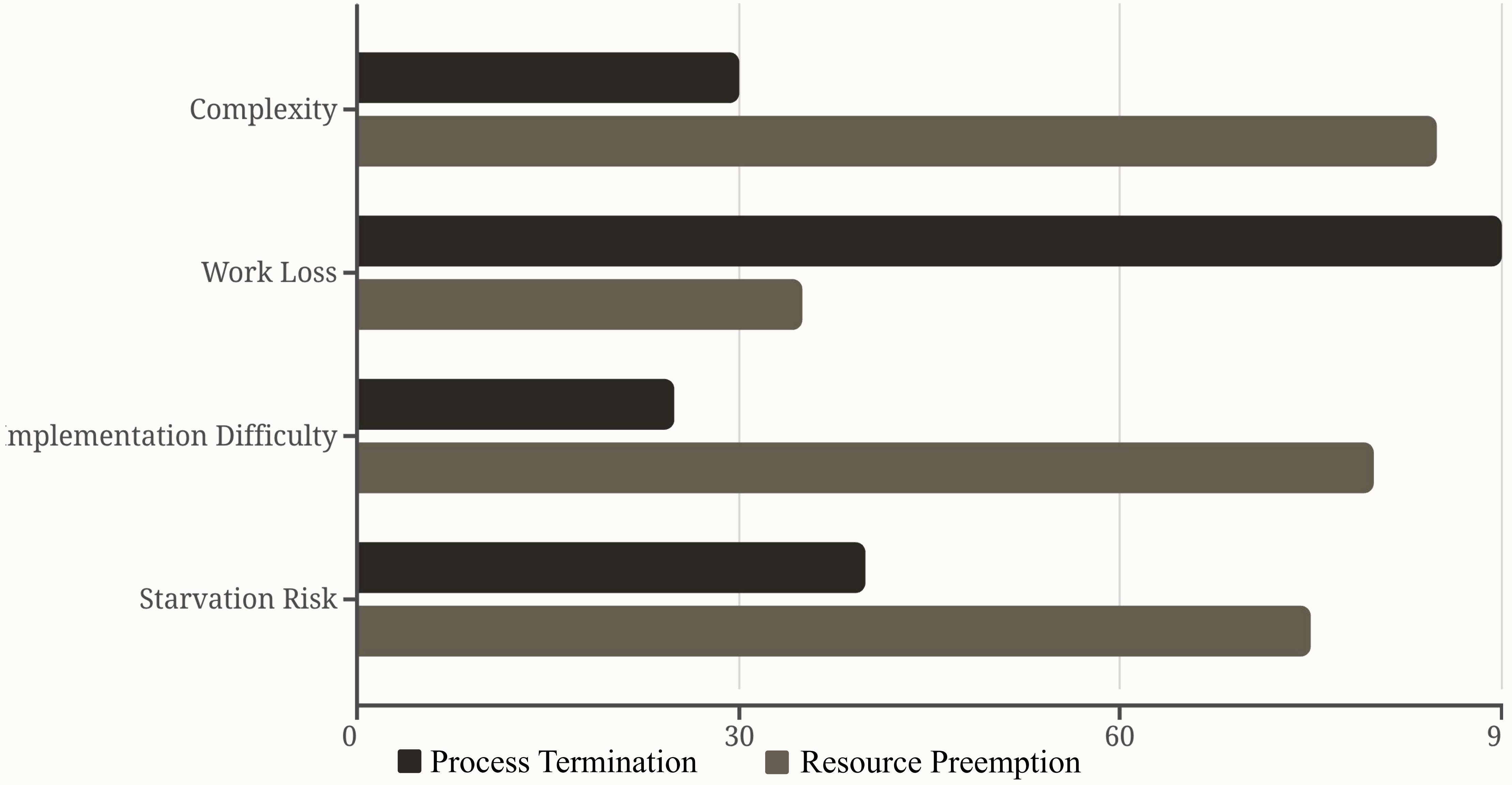
The goal is to break the circular wait condition and free up resources so that blocked processes can resume normal execution.

Two Main Approaches

- **Process Termination:** Abort one or more deadlocked processes
- **Resource Preemption:** Forcibly take resources from processes



Comparison of Recovery Techniques



www.snsgroups.com

Deadlock Problem in Operating Systems

Problem Statement

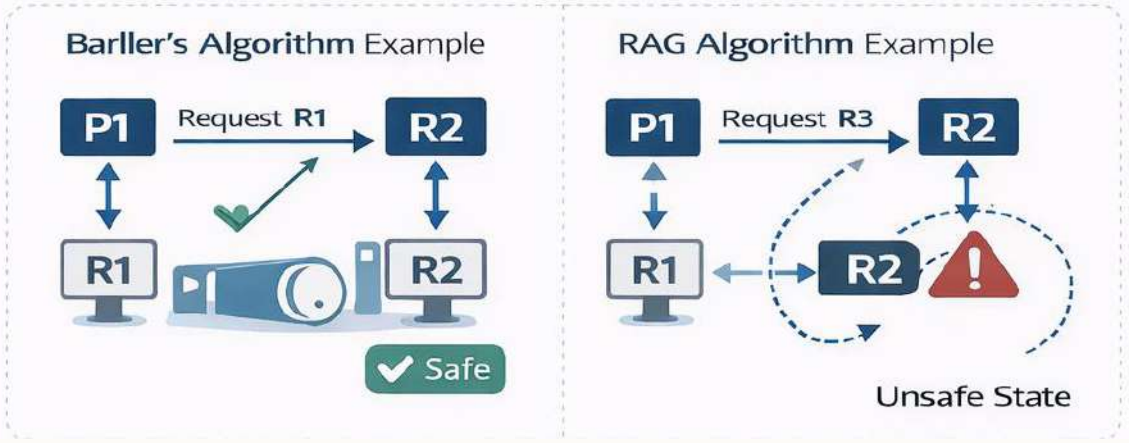
In a multi-programming operating system, multiple processes compete for limited system resources such as CPU, memory, I/O devices, and files.

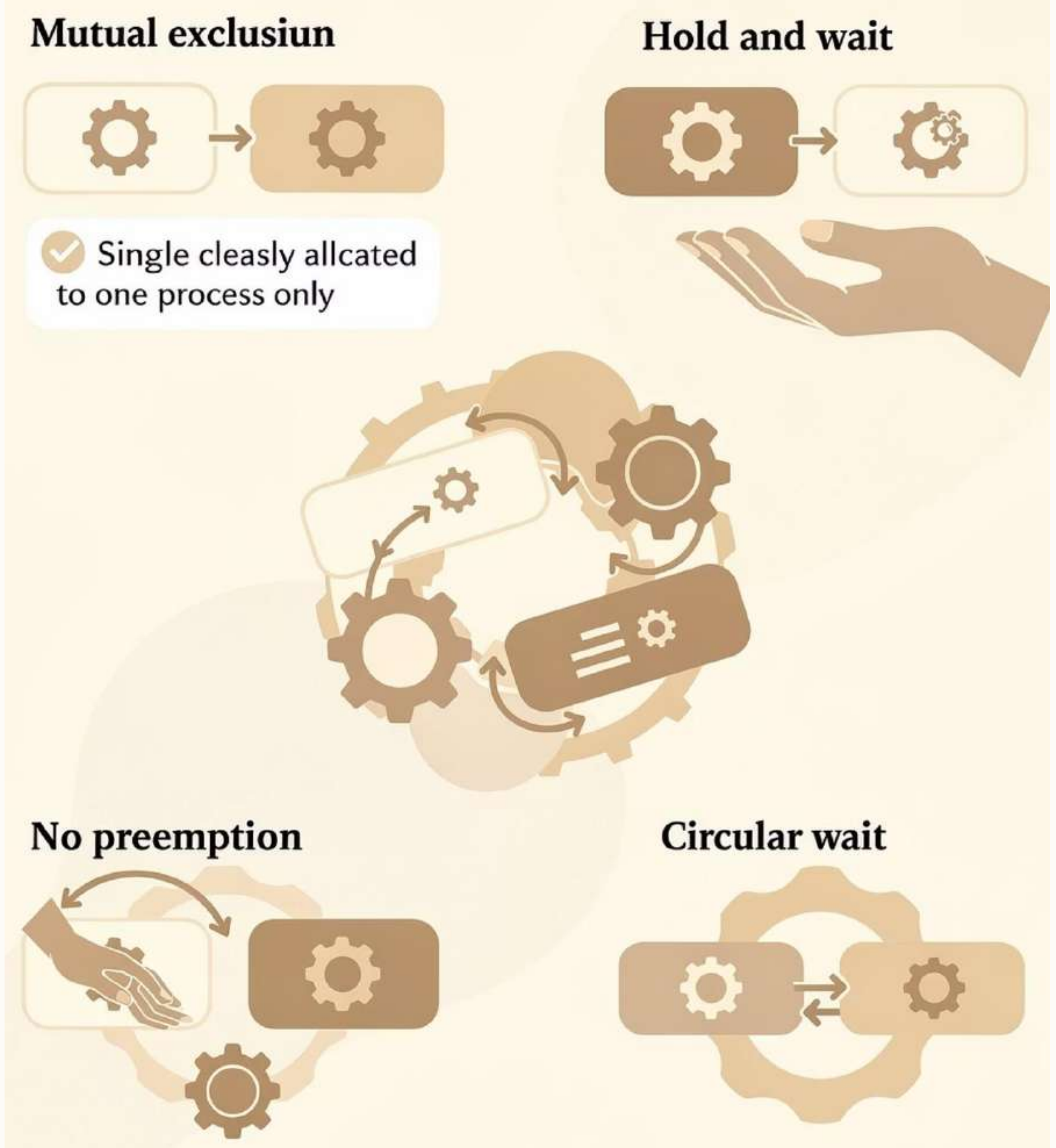
If resource allocation is not handled properly, the system may enter a **deadlock state**, where a group of **processes** are permanently blocked, each waiting for resources held by others.

The challenge is to design strategies that either **prevent**, **avoid**, detect, or **recover** from **deadlocks** while maintaining system **efficiency** and fairness.

Deadlock Handling Strategies

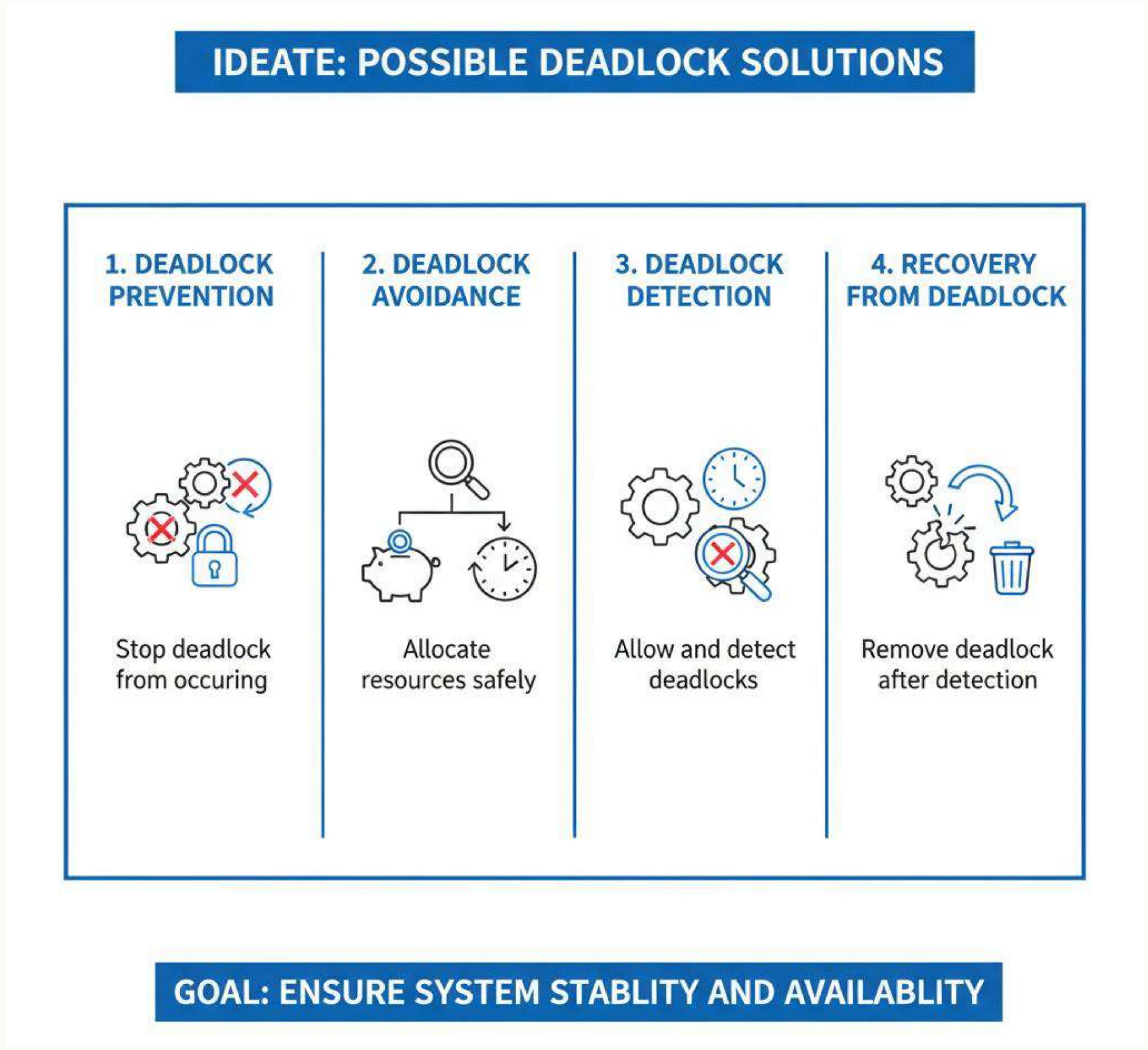
- Prevention
- Avoidance
- Detection
- Recovery





DEFINE: Understanding Deadlock

Identify the four necessary conditions and analyze the impact of deadlock on system responsiveness and resource utilization.



IDEATE: POSSIBLE DEADLOCK SOLUTIONS

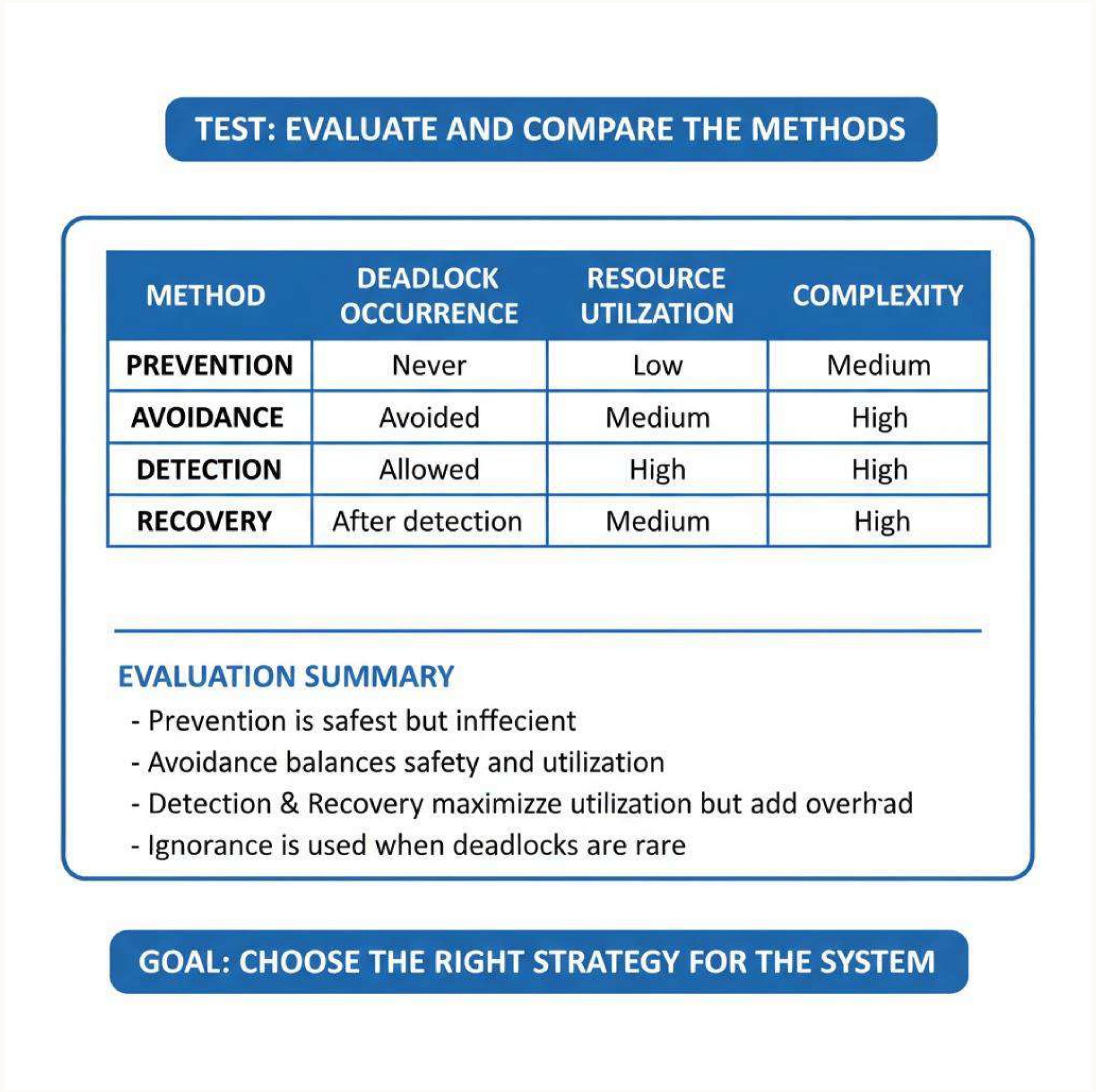
1. DEADLOCK PREVENTION	2. DEADLOCK AVOIDANCE	3. DEADLOCK DETECTION	4. RECOVERY FROM DEADLOCK
Stop deadlock from occuring	Allocate resources safely	Allow and detect deadlocks	Remove deadlock after detection

GOAL: ENSURE SYSTEM STABLITY AND AVAILABLITY

PROTOTYPE

Designing Deadlock Management Methods

- **Resource Ordering:** Enforce a strict order for resource requests.
- **Banker's Algorithm:** Assess resource allocation safety before granting requests.
- **Cycle Detection:** Detect deadlocks using resource allocation graphs.
- **Recovery Procedures:** Terminate processes or preempt resources to break deadlocks.



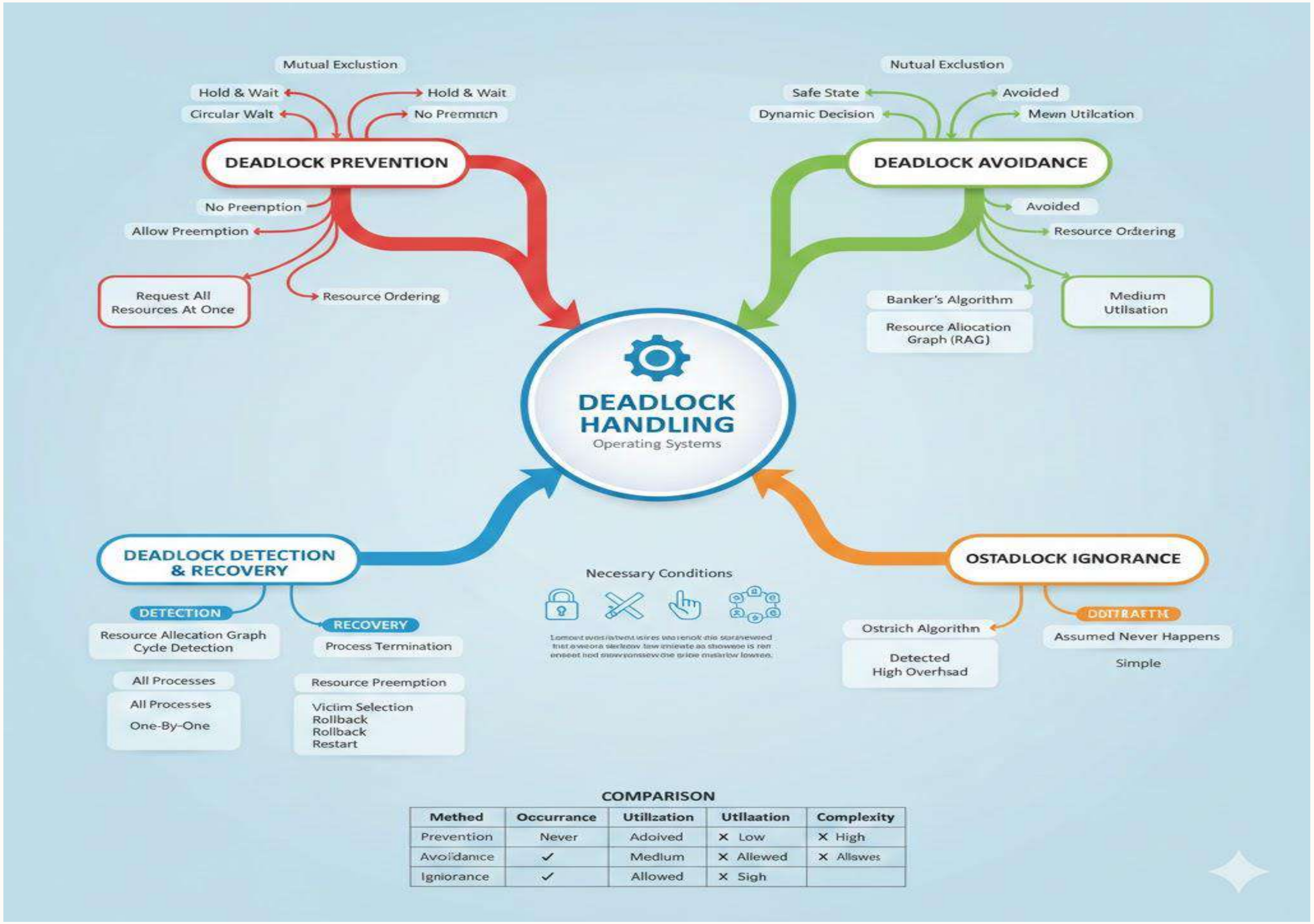
TEST: EVALUATE AND COMPARE THE METHODS

METHOD	DEADLOCK OCCURRENCE	RESOURCE UTILZATION	COMPLEXITY
PREVENTION	Never	Low	Medium
AVOIDANCE	Avoided	Medium	High
DETECTION	Allowed	High	High
RECOVERY	After detection	Medium	High

EVALUATION SUMMARY

- Prevention is safest but inffecient
- Avoidance balances safety and utilization
- Detection & Recovery maximizze utilization but add overh·ad
- Ignorance is used when deadlocks are rare

GOAL: CHOOSE THE RIGHT STRATEGY FOR THE SYSTEM



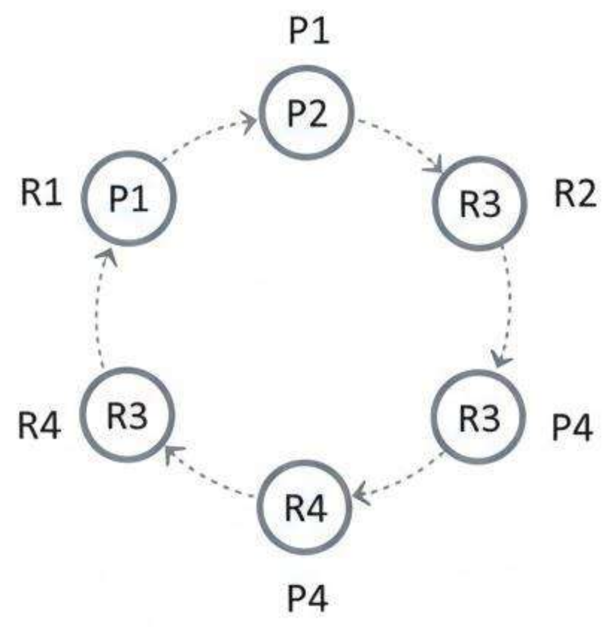
COMPARISON

Methed	Occurrance	Utillzation	Utllaation	Complexity
Prevention	Never	Adoived	✕ Low	✕ High
Avoiidanice	✓	Medlum	✕ Allewed	✕ Allswes
Igniorance	✓	Allowed	✕ Sigh	

DEADLOCK PUZZLE (REASONING BASED)

The Four Resource Lab

Current Situation



Tasks (Answer in Words / Diagrams)

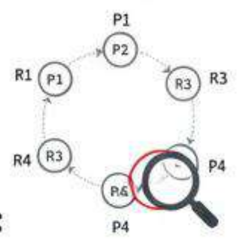
1. DEADLOCK STATE?

✓ YES, a circular wait deperdency exists.

2. NECERSARY CONDITIONS:

- **Mutual Exclusion:** Resources are non-sharable.
- Hold & Wait: Processes hold resources while waiting ion others.
- No Preemption: Resources cannot to taken.
- Circular Wait: P1 >R2, P2-R3, P3 >R>R4 P4 >R1

3. RESOURCE ALLOCATION GRAPH:



4. PREVENTION TECHNIQUE:

(e.gl., impose resource ordering)

5. RECOVERY METHOD

Process Termination (e.g. Abort P1 to break cycle. Victim choice depends on cost/priority).

Thank You