Deadlock

no progress as each P is waiting for R held by other (cyclic)

Deadlock → No progress

Starvation → Progress, But low priority process are starving

Deadlock Conditions

If all conditions are met (simultan) only then deadlock occurs

COFFMANN conditions

Avoiding

Circular wait

$P_1 \rightarrow P_2$
$\uparrow \qquad \downarrow$
$P_4 \leftarrow P_3$

Mutual Exclusion

Only 1 process can use a resource at a time

Hold & wait

Process can hold some resources and also request & wait for other resources held by other processes

No preemption

Can't forcibly take Resources from a process (only voluntarily removed).

If any of these conditions are not met then deadlock cannot occur !!

Deadlock Handling

- Prevention
- Avoidance
- Detection
- Recovery

Deadlock Detection:

Identifying and resolving situations where multiple resources are waiting indefinitely for resources held by each other, resulting in deadlock.

- To represent Resource allocation & Request in DS
  - ↳ Resource Allocation Graphs are used.

Resource Allocation Graph:

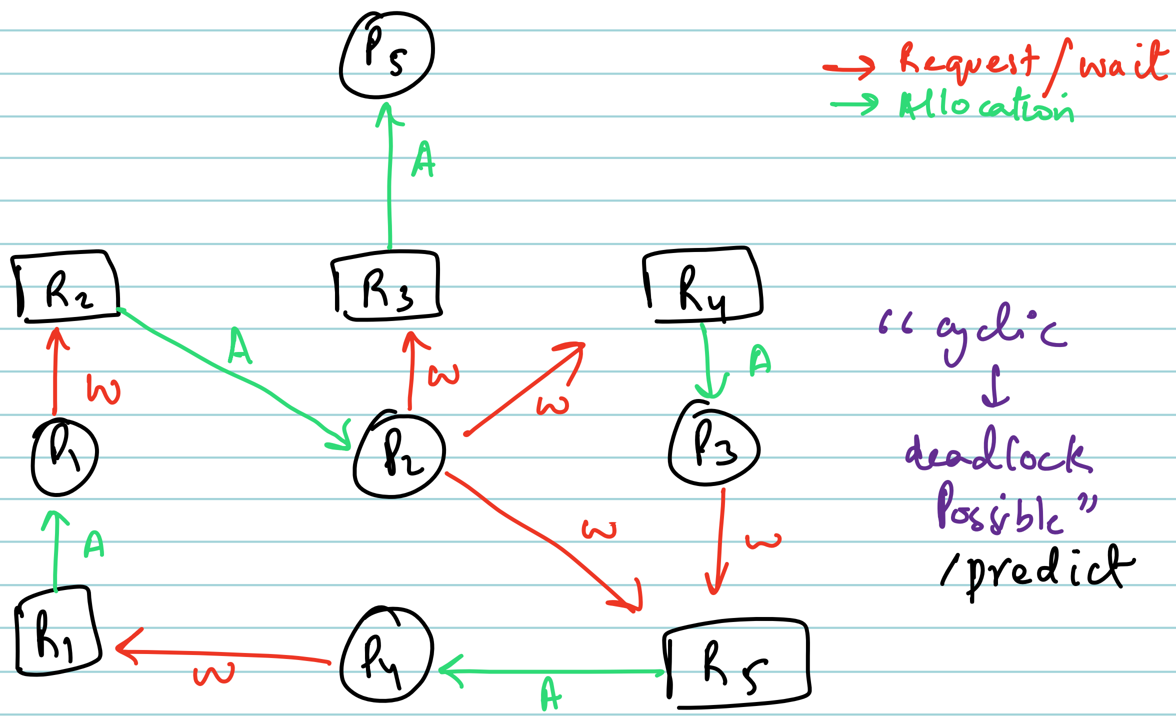
"Shows state of the System"

Process ◯

Resources ▭

P → R (Request)

R → P (Allocation)



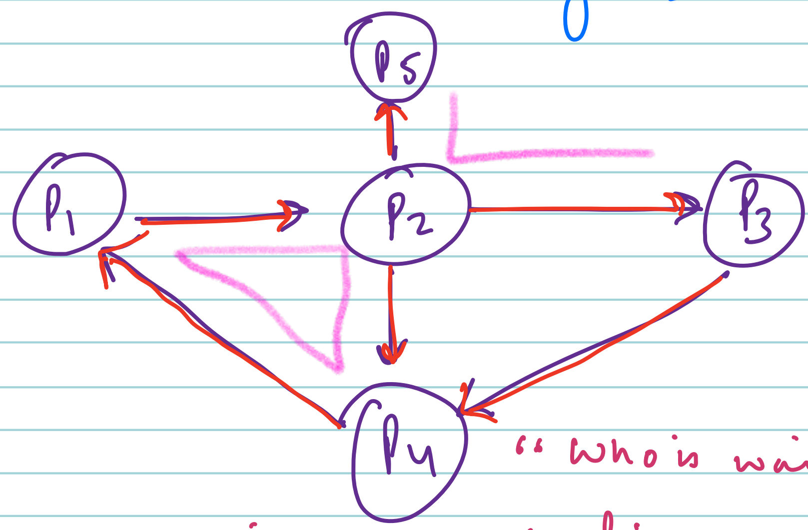
"If cycle exists then there could be a deadlock"

Resource Allocation Graph

Wait for Graph

↳ Only Processes not Resources with directed edges.

example from RAG, P_1 is waiting for P_2 to finish using R2



"ensures whether deadlock occurs or not"

"who is waiting for whom"

cyclic with knot

Wait for graph

$T_n \xrightarrow[\text{Req}]{} \boxed{\text{Resource}} \xrightarrow[\text{Hold.}]{} T_k$

$T_n \rightarrow$ Requests
$T_k \rightarrow$ Holds.

	Wait die	Wound wait
T_n is younger than t_k	T_n dies	T_n waits
T_n is older than t_k	T_n waits	T_k aborts

Wait die → Old waits, new dies

Wound wait → Old wounds new one, new waits