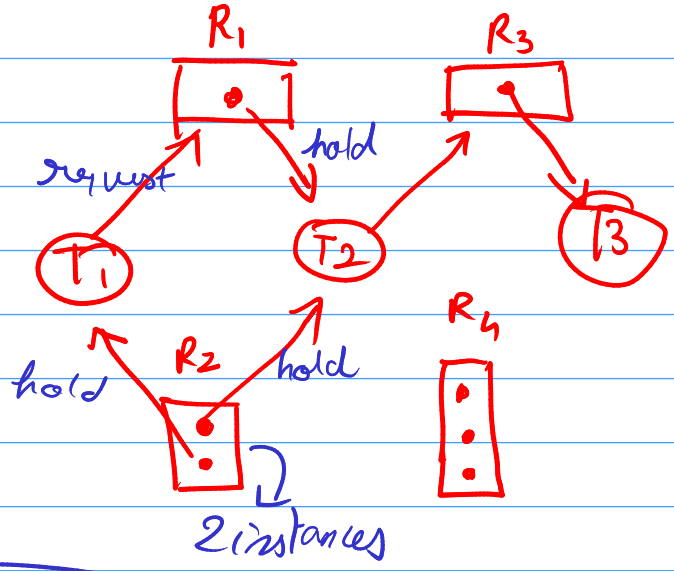
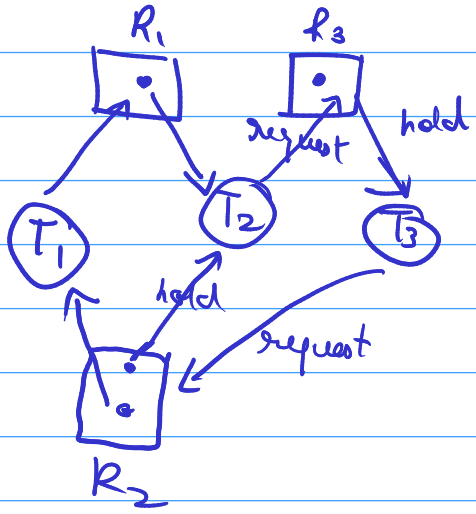
Resource allocation graph



Deadlocks



If cycle
one instance then deadlock
many instances ⇒ possibility of deadlock

Handle deadlocks

→ Deadlock prevention
→ Deadlock avoidance

→ Allow system to enter a deadlock and recover

→ Ignore it and pretend it never happens!

Prevention (invalidate 1 of cond. for deadlock)

→ Mutual Exclusion
  ↳ for non-shareable (writeable) files

→ Hold & wait
  ↳ cannot both hold & request resource
  (Starvation possible)

→ No preemption
(We need to prevent no-preemption ⇒ preempt the process holding the resources, and release resources)

→ Circular wait:
Make a TOset of all resource types

Circular Wait

Make a TOSET out of resources

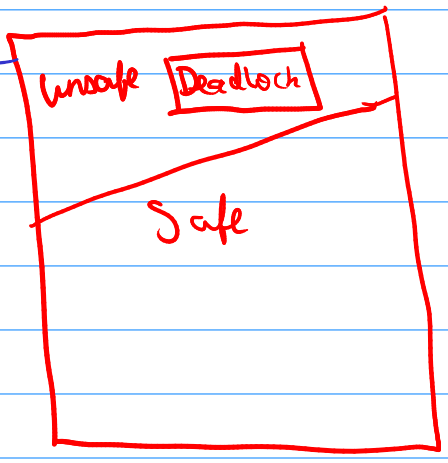
Deadlock Avoidance [Requires a priori info]

Safe state :- (S state)

System is in **safe state** if there exists a sequence $<P_1, P_2, \ldots, P_n>$ of ALL the processes in the systems such that for each P_i, the resources that P_i can still request can be satisfied by currently available resources + resources held by all the P_j, with $j < I$

Safe, Unsafe, Deadlock State

Ensure System never reaches here



Single instance of a resource type

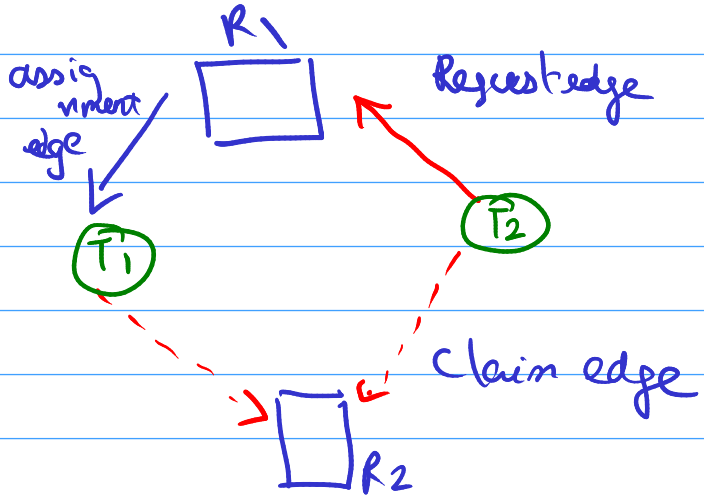
Resource allocation graph

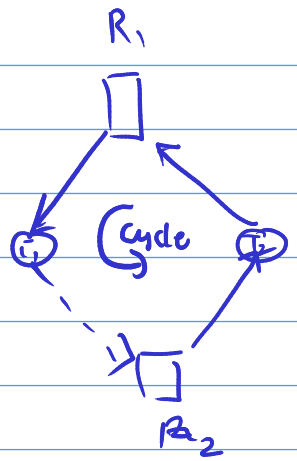
Multiple instances
⇒ Banker's algorithm

Resource Allocation Graph Scheme

Claim Edge: $P_i \rightarrow R_j$ (dashed) (may request)

to ↳ Request edge → assignment edge → resource released





if the claim edge becomes a request edge, deadlock happens.