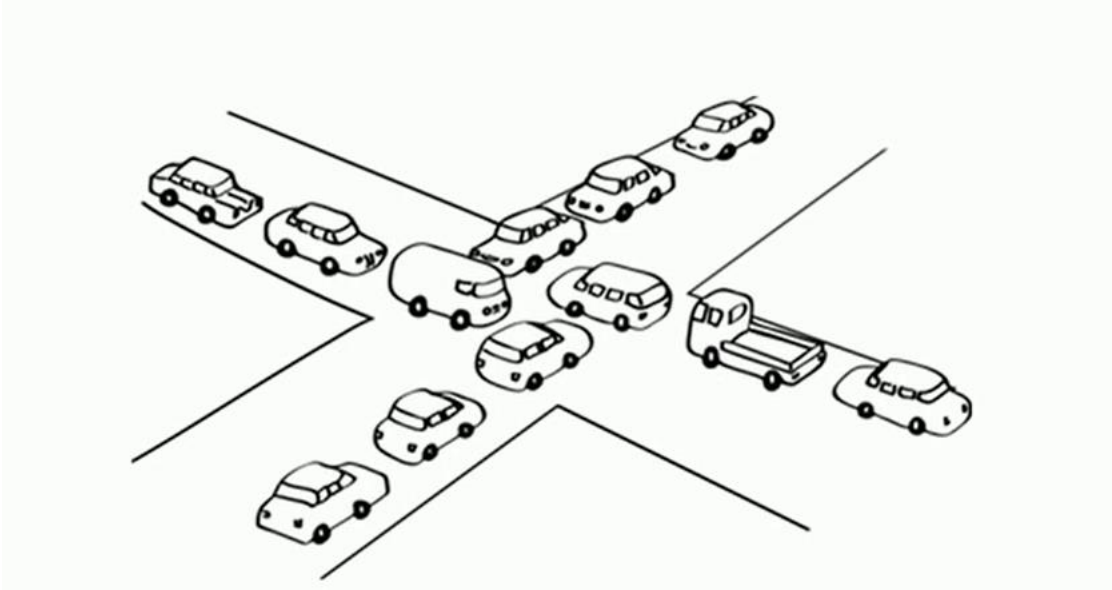
OPERATING SYSTEM: CSET209

OUTLINE

- Deadlock
- Resource Allocation graphs

DEADLOCK

A situation where a set of processes **continue to run indefinitely** without making any **progress**. In this situation, none of the process gets executed since the **resource it needs**, is **held by some other process** which is also **waiting for some other resource to be released.**



If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback)

DEADLOCK EXAMPLE

- A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set

- Example
  - semaphores *A* and *B*, initialized to 1

P_0	P_1
wait (A);	wait(B)
wait (B);	wait(A)

DEADLOCK EXAMPLE

```
void transaction(Account from, Account to, double amount)
{
   mutex lock1, lock2;
   lock1 = get_lock(from);
   lock2 = get_lock(to);
   acquire(lock1);
      acquire(lock2);
         withdraw(from, amount);
         deposit(to, amount);
      release(lock2);
   release(lock1);
}
```

- Transactions 1 and 2 execute concurrently. Transaction 1 transfers $25 from account A to account B, and Transaction 2 transfers $50 from account B to account A

NECESSARY CONDITIONS FOR DEADLOCKS

1. Mutual Exclusion

A resource can only be shared in mutually exclusive manner. It implies, if two process cannot use the same resource at the same time.

2. Hold and Wait

A process waits for some resources while holding another resource at the same time.

3. No preemption

A resource can be released only voluntarily by the process holding it, after that process has completed its task

4. Circular Wait

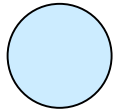
All the processes must be waiting for the resources in a cyclic manner so that the last process is waiting for the resource which is being held by the first process.

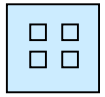
RESOURCE-ALLOCATION GRAPH

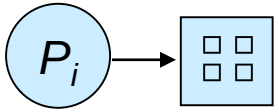
A directed graph used to model resource requests and allocations.

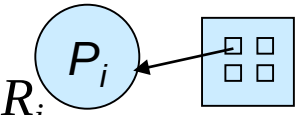
request edge – directed edge $P_i \rightarrow R_j$

assignment edge – directed edge $R_j \rightarrow P_i$

Process 

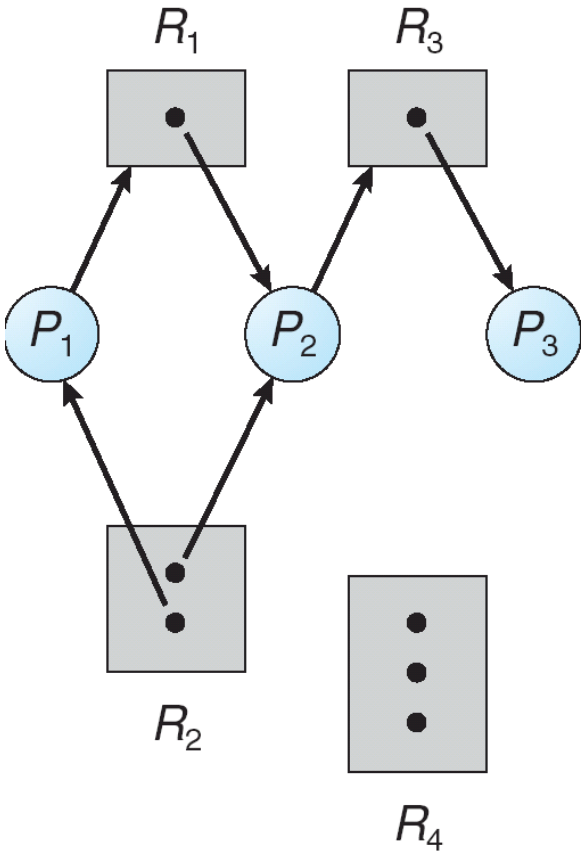
Resource Type with 4 instances 

P_i requests instance of R_j 

P_i is holding an instance of R_j 

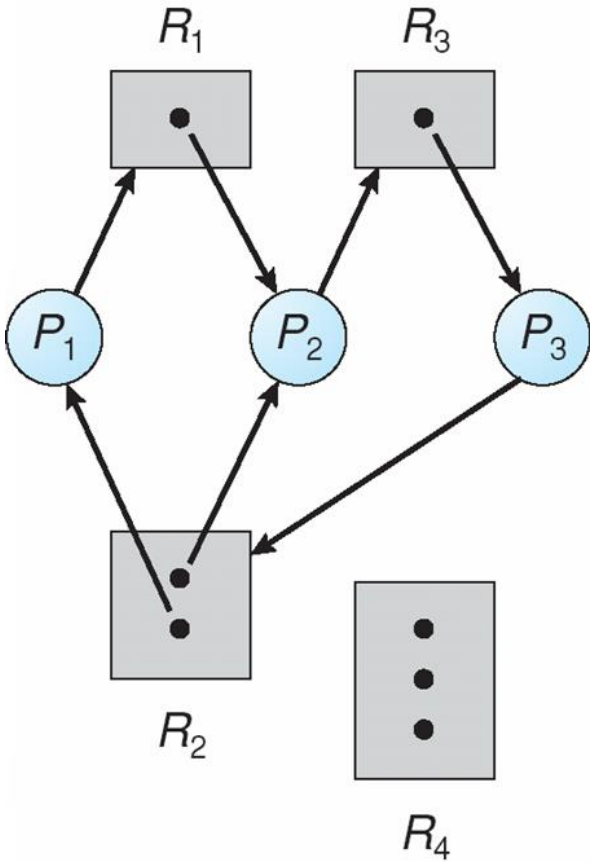
R_j

RESOURCE-ALLOCATION GRAPH EXAMPLE



- **IS THERE A DEADLOCK HERE?**
- **NO**

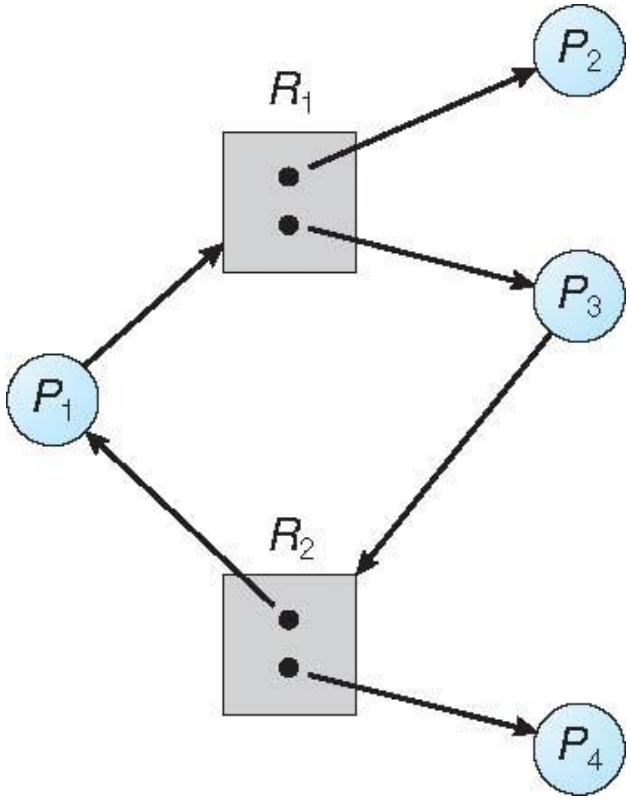
RESOURCE-ALLOCATION GRAPH EXAMPLE



IS THERE A DEADLOCK HERE?

YES

RESOURCE-ALLOCATION GRAPH EXAMPLE



IS THERE A DEADLOCK HERE?

NO

BASIC FACTS

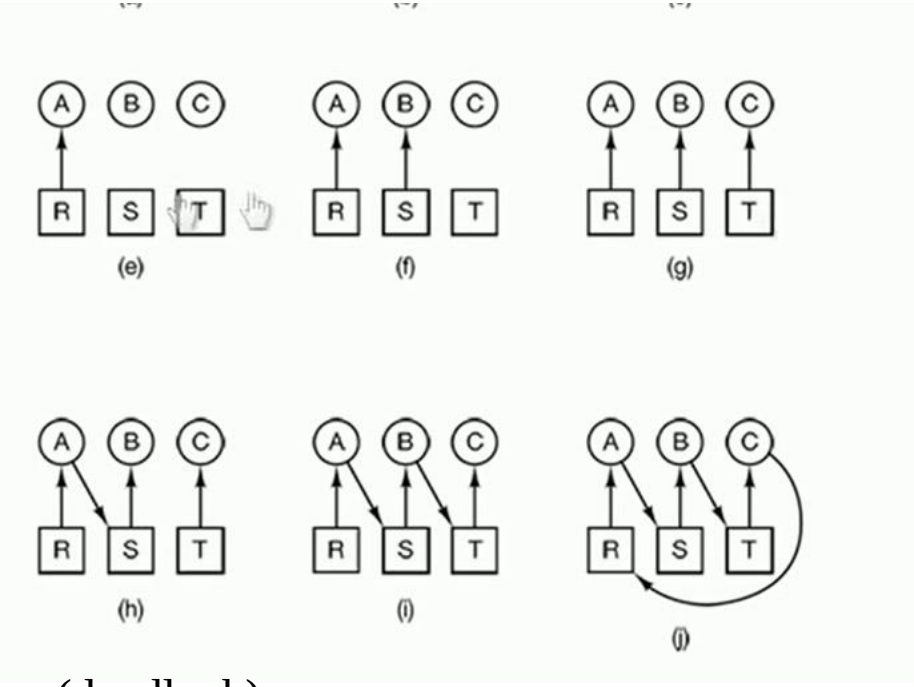
- If graph contains no cycles $\Rightarrow$ no deadlock
- If graph contains a cycle $\Rightarrow$
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

DEADLOCK IS A PROBABILISTIC EVENT

- Suppose there are 3 processes A, B, and C requesting resources R, S, and T as follows.

A	B	C
Request R	Request S	Request T
Request S	Request T	Request R
Release R	Release S	Release T
Release S	Release T	Release R

- Let the order of execution be
  A request R
  B request S
  C request T
  A request S
  B request T
  C request R



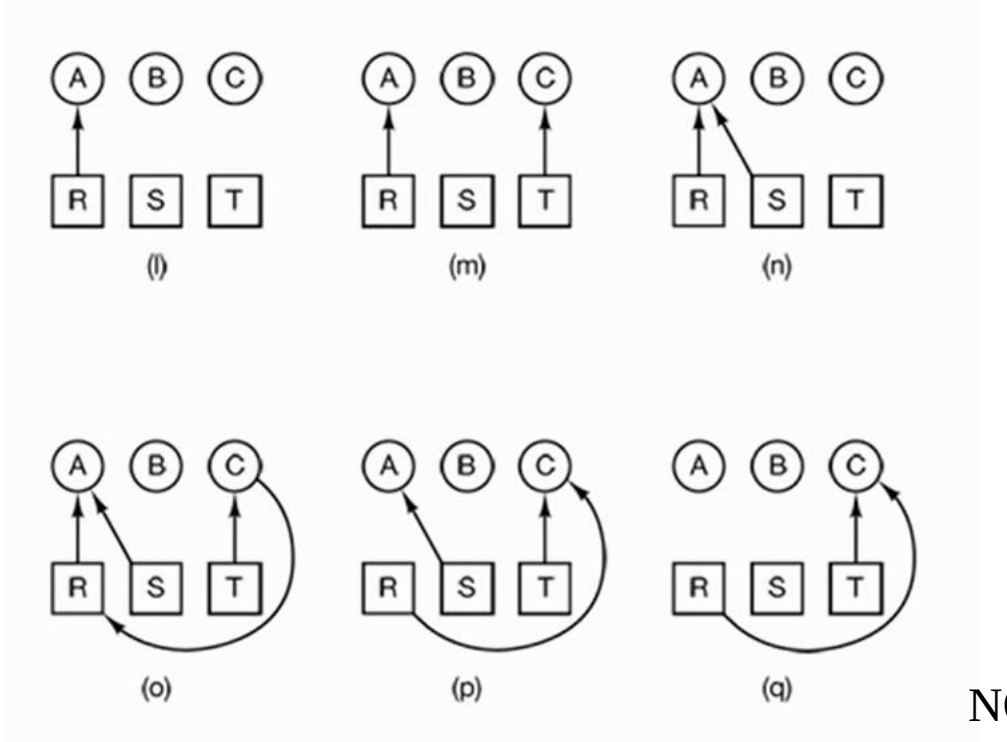
(deadlock)

DEADLOCK IS A PROBABILISTIC EVENT

- Suppose there are 3 processes A, B, and C requesting resources R, S, and T as follows.

A	B	C
Request R	Request S	Request T
Request S	Request T	Request R
Release R	Release S	Release T
Release S	Release T	Release R

- Let the order of execution be
  A request R
  C request T
  A request S
  C request R
  A release R
  A release S



NO deadlock)

Thank you !