

Causes of deadlocks

- Four necessary conditions for deadlock to occur are:
 - **Exclusive access:** processes require exclusive access to a resource
 - **Wait while hold:** processes hold on previously acquired resources while waiting for additional resources
 - **No preemption:** a resources cannot be preempted from a process without aborting the process
 - **Circular wait:** there is a set of blocked processes involved in a circular wait
- The first three properties are generally desirable
 - respectively to i) preserve resource integrity, ii) increase resource utilization, iii) reduce waste of CPU time

Deadlock handling policies

- Deadlock prevention
 - the system is designed so that granting requests never leads to a deadlock
- Deadlock detection
 - the system periodically (or when deadlock suspected) checks for deadlocks, and a recovery procedure is started if one is detected
- Deadlock avoidance
 - resources are granted only if the resulting system state is *safe* i.e. there is at least one sequence of execution in which all processes run to completion

Deadlock analysis

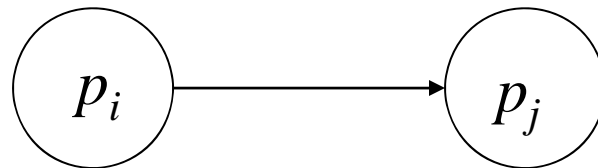
- Deadlock analysis is complicated by the number of variants of the process-resource relationship
 - how many and exactly which resources are needed?
 - How many times a resource can be used?
 - How many processes can simultaneously use a resource?
- We will study:
 - simple models used to represent requests and resource types
 - Increasingly sophisticated graph-based techniques for deadlock analysis

Models of deadlock

- Depending on the type of resource request, a deadlock can be classified according to one of four types
 - **Single-unit request model:** vanilla variety
 - **AND request model:** I need a CPU AND a disk AND a printer ...
 - **OR request model:** I need a disk OR a printer ...
 - **AND-OR request model:** I need a CPU AND (a disk OR a printer)
...
 - **P-out-of-Q request model:** I need at least three consensus votes out of five

Wait-for-graph (WFG)

- To study deadlocks the state of a system is often represented with a wait-for graph

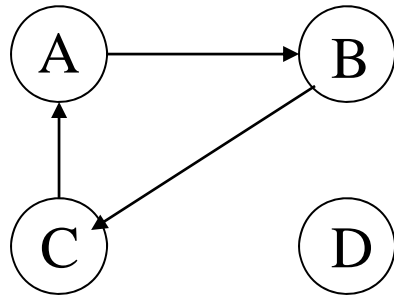


p_i needs a resource held by p_j

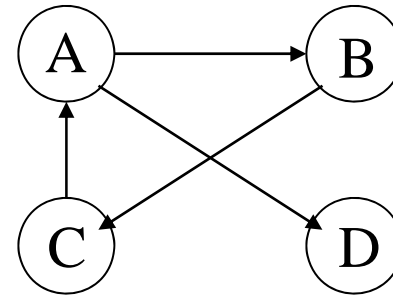
- A cycle in the graph represents a circular wait

Definition of *knot*

- A *knot* of a graph is a subset K of nodes such that the reachable set of each node in K is exactly K
- Examples:



A, B, C are in a cycle
and also in a knot



A, B, C are in a
cycle, not in a knot

Model differences

- **Single-unit request model:**
 - a deadlock corresponds to a *cycle* in the WFG
- **AND request model:**
 - same as above (if every resource is in single copy), but a process now can be involved in more than one deadlock
- **OR request model:**
 - a *knot* is a sufficient condition for a deadlock
- **AND-OR request model:**
 - a *knot* is a sufficient condition for a deadlock
- **P-out-of-Q request model:**
 - a *knot* is a sufficient condition for a deadlock

Types of resources

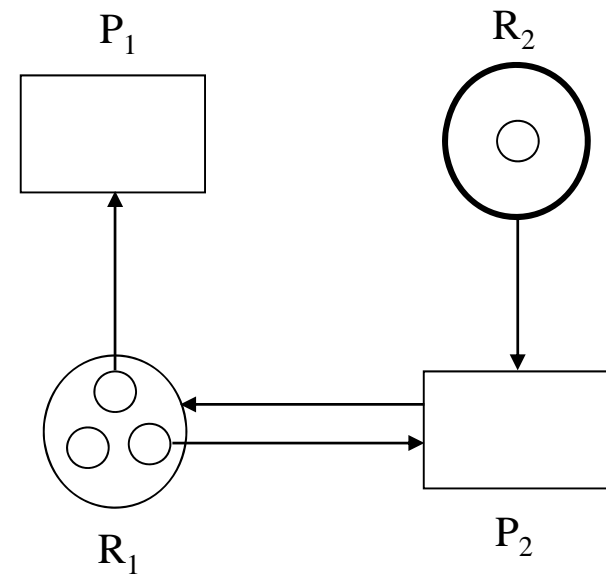
- Reusable resources
 - examples: CPU, memory, I/O devices
 - are not “consumed” by use
 - fixed number of units
 - when allocated to a process, they are hold until the process is done and then released
- Consumable resources
 - examples: messages, interrupt signals, V operation in semaphores
 - they vanish as a result of their use
 - when allocated to a process, they are consumed and cease to exist
- Exclusive vs. shared access
 - We will only consider exclusive access

General Resource Graph

- A General Resource Graph is a directed graph where
 - nodes represent resources and processes
 - resources are denoted with circles (thick circles for consumable resources)
 - processes are designated with rectangles
 - edges represent interaction between processes and resources
- The GRG is used to represent snapshots of a system state
 - changes in the state are represented by changes in the graph

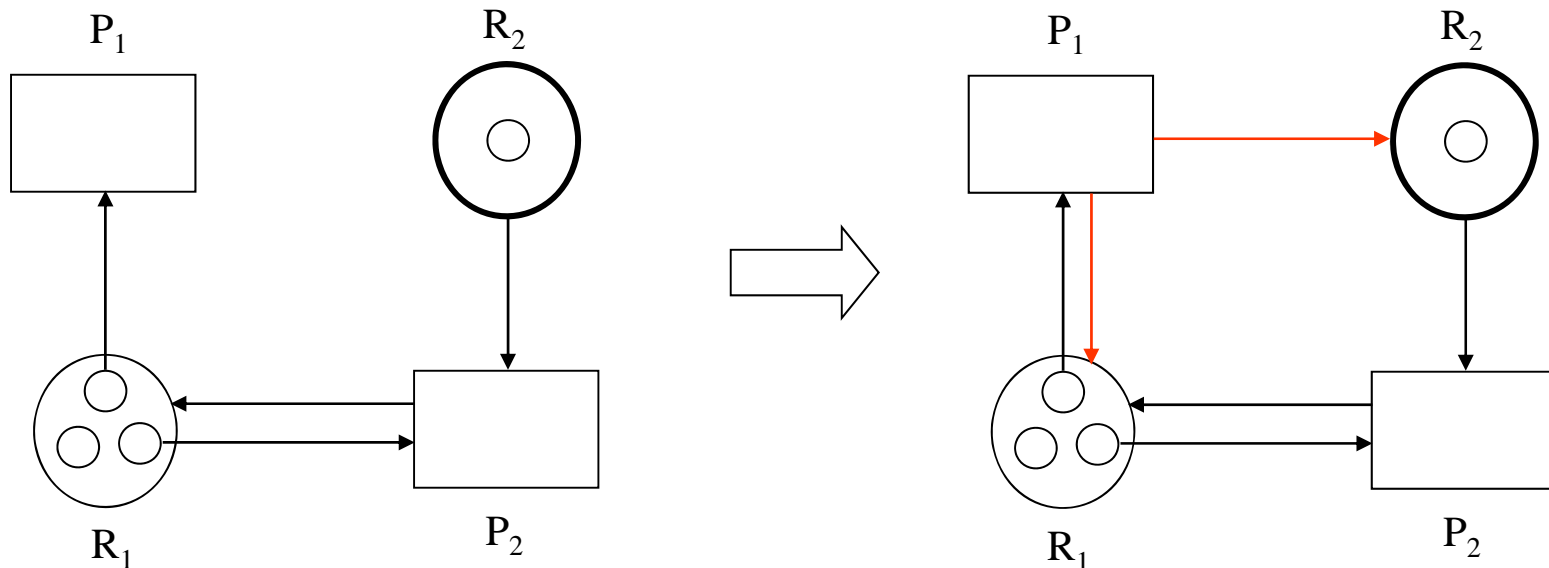
General Resource Graph (2)

- If R is a reusable resource node and P a process node:
 - edge (R, P) from R to P is an *assignment* edge
 - edge (P, R) from P to R is a *request* edge
- If R is a consumable resource
 - edge (R, P) is a *producer* edge



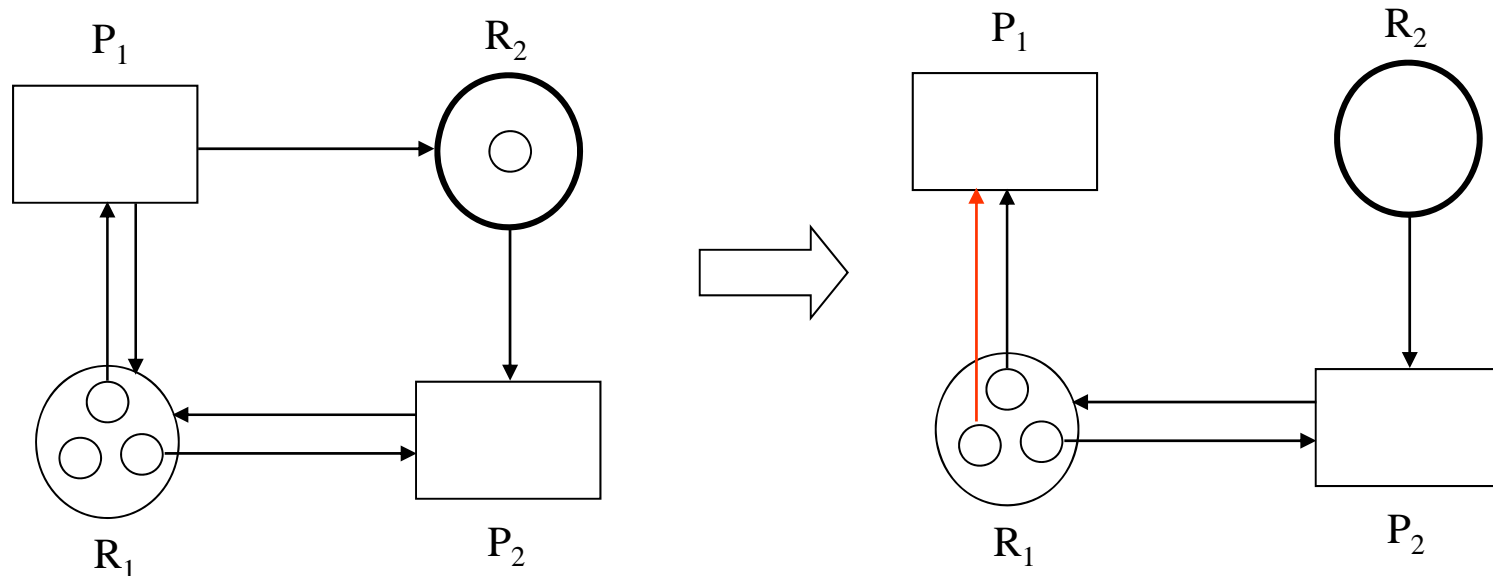
Operations on a GRG: Request

- A process P can request one or more units of resource $R \rightarrow$ addition of edges to the graph
- Example: P_1 requests one unit each of R_1 and R_2



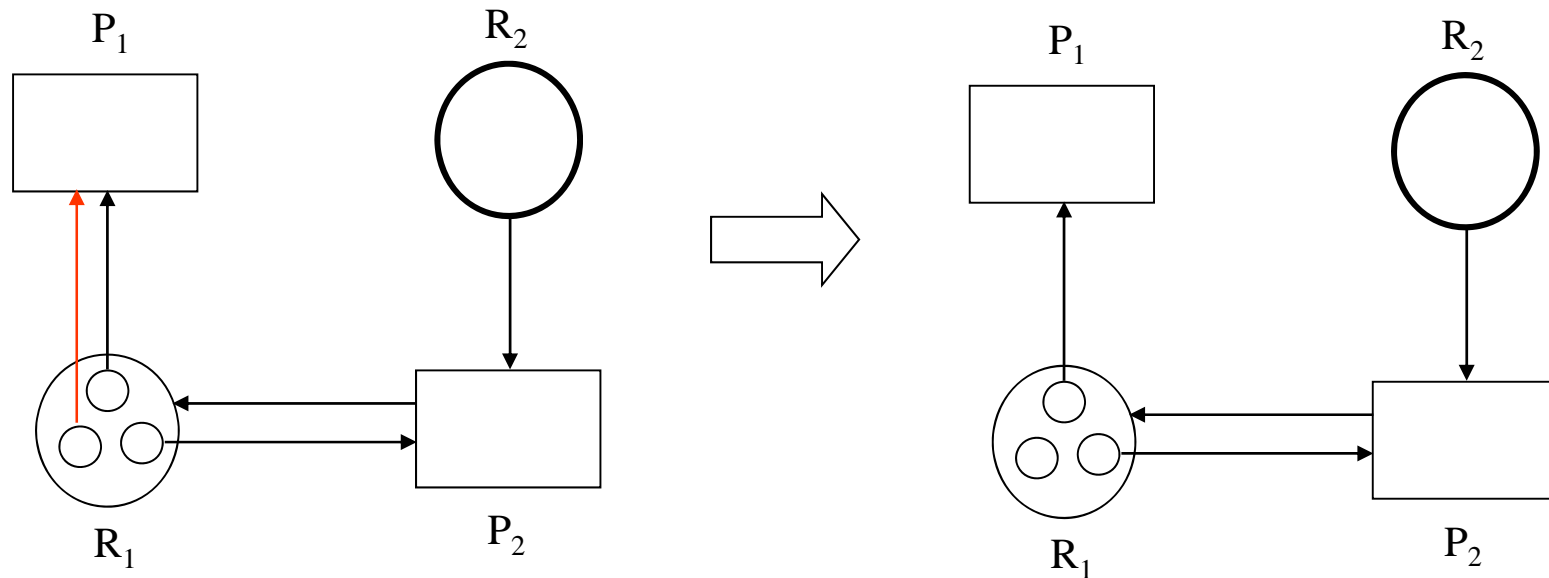
Operations on a GRG: Acquisition

- A process P can be granted some of its requests
 - \rightarrow request edges are replaced with assignment edges (reusable resources)
 - \rightarrow request edges are canceled (consumable resources)
- Example: P_1 gets one unit each of R_1 and R_2



Operations on a GRG: Release

- A process P can release some units of its currently held (or produced resources)
 - \rightarrow assignment edges are canceled
- Example: P_1 releases one unit of R_1



Deadlock analysis using the GRG

- The GRG is a powerful theoretical tool
 - simplified view of the state of the system
 - we are going to see theorems linking graph properties to presence/absence of deadlock
- A powerful analysis tool is the **graph reduction method**
 - the idea is to use a graph to test sets of operations on the system
 - a **reduction** is the most optimistic set of operations that unblocks one or more blocked processes