

Deadlock and starvation: differences

- **Process status:**
 - deadlock: processes are permanently blocked because the resources never become available
 - starvation: it is not certain the process will ever acquire the requested resource
- **Resource status**
 - deadlock: contended resources are not in use
 - starvation: contended resources in continuous use (by others)

Deadlock handling strategies

- 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

Strategy #1: Deadlock prevention

- 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
- The fourth condition is the most commonly targeted for deadlock prevention
 - Typical approach: mandate some order in which resources must be acquired

Starategy #2: Deadlock detection

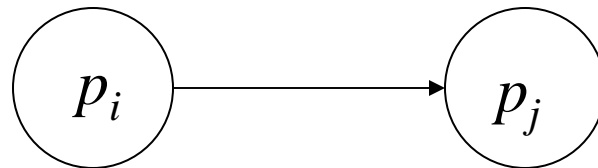
- Deadlock analysis is complicated by the number of variants of the process-resource relationship
 - how many and exactly which resources are needed? How many times a resource can be used?
- We will study basic principles of deadlock analysis:
 - simple graph-based models used to represents requests and resources
 - Increasingly powerful criteria that relate topological model features (i.e. presence of a cycle) to deadlock properties of the system

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

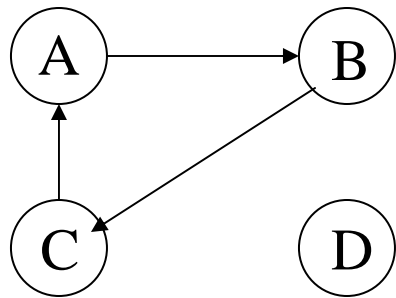
- A cycle in the graph represents a circular wait

Definition of “knot” in graph theory

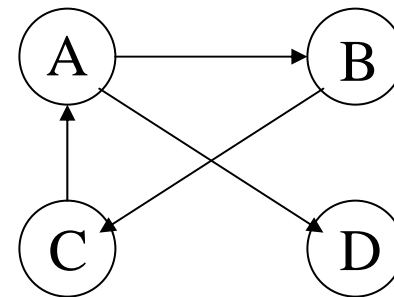
- The following definition seems more accurate to me than the one in the book:
*“A **knot** in a directed graph is a collection of vertices and edges with the property that every vertex in the knot has outgoing edges, and all outgoing edges from vertices in the knot have other vertices in the knot as destinations. Thus it is impossible to leave the knot while following the direction of the edges.”*

[From: “Glossary of Graph Theory” (2005, November 6). Wikipedia, The Free Encyclopedia. Retrieved November 18, 2005 from http://en.wikipedia.org/wiki/Glossary_of_graph_theory]

- Examples:



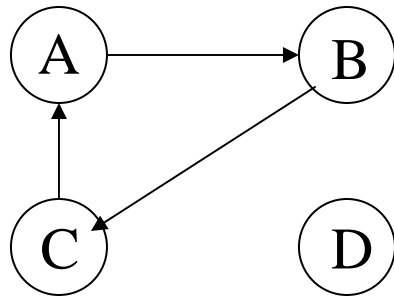
A, B, C are in a cycle
and also in a knot



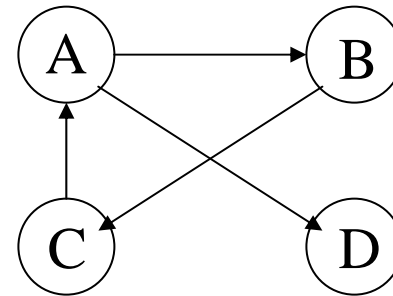
A, B, C are in a
cycle, not in a knot

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

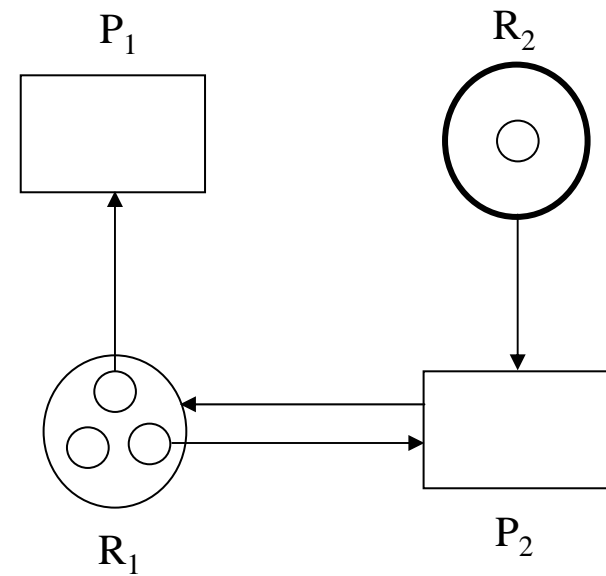
- Reusable resources
 - examples: CPU, memory, I/O devices
 - are not “consumed” by use
 - fixed number of units
 - when allocated to a process, they are hold until the process is done and then released
- Consumable resources
 - examples: messages, interrupt signals, V operation in semaphores
 - they 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

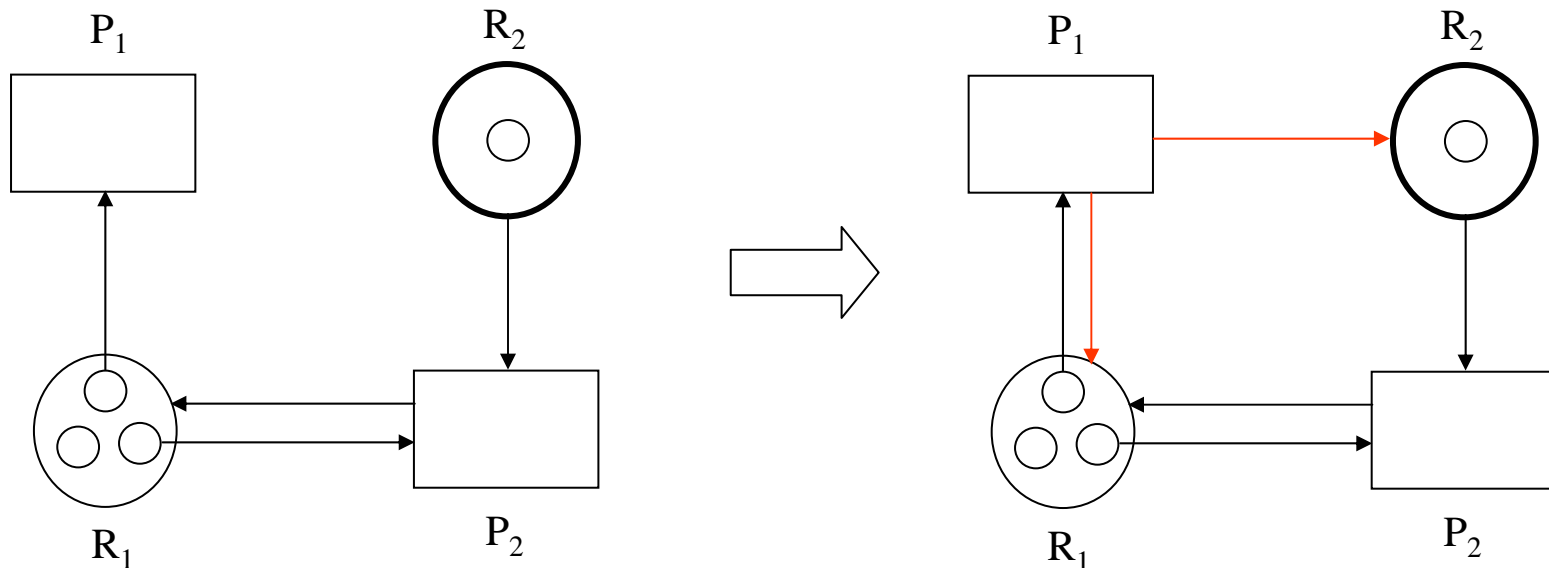
General Resource Graph (2)

- 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



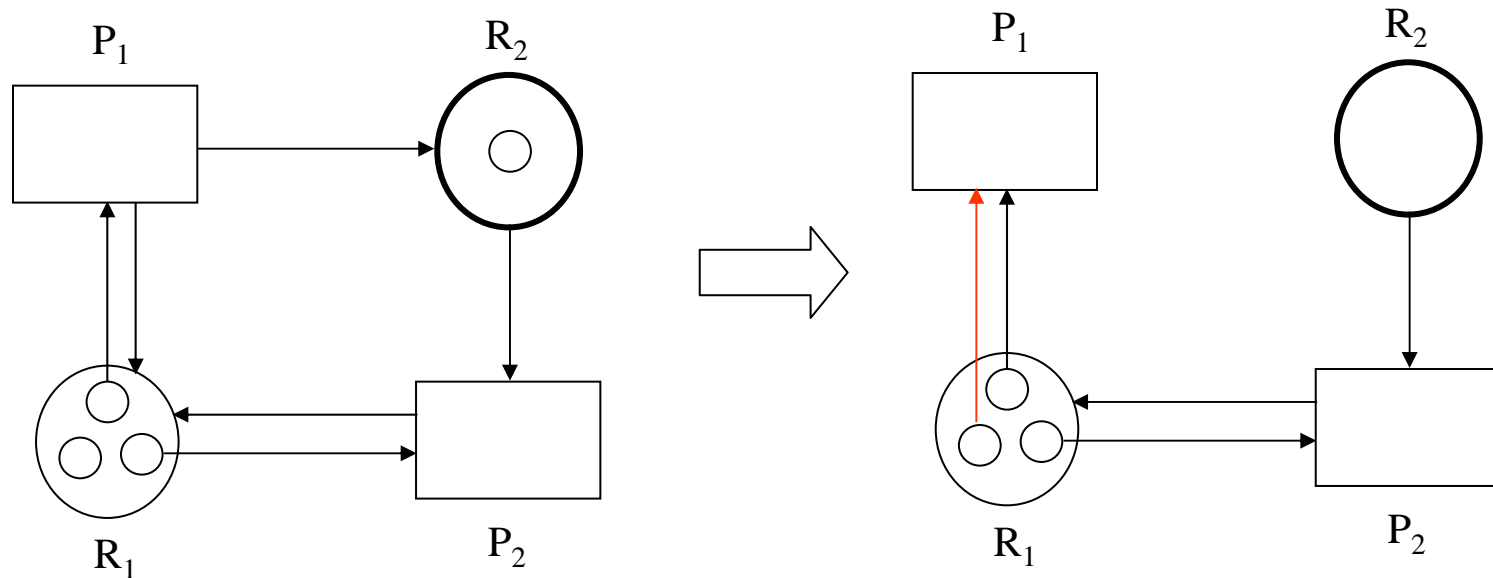
Operations on a GRG: Request

- 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



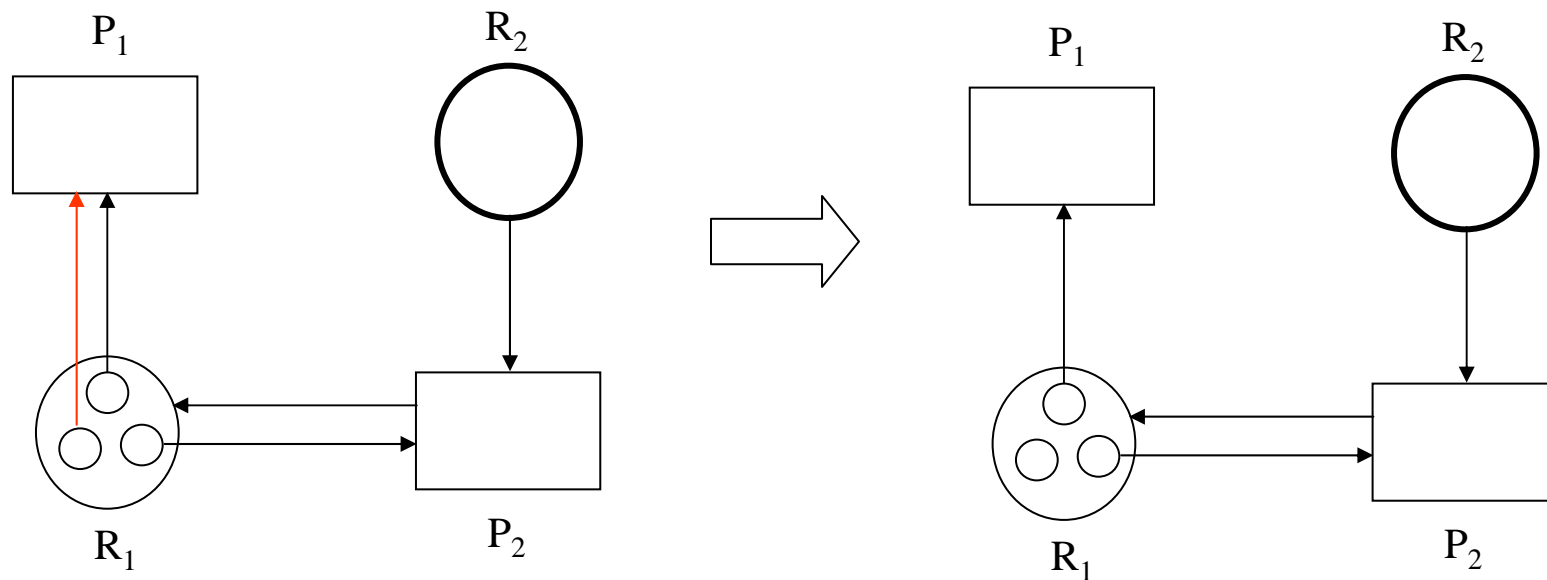
Operations on a GRG: Acquisition

- 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

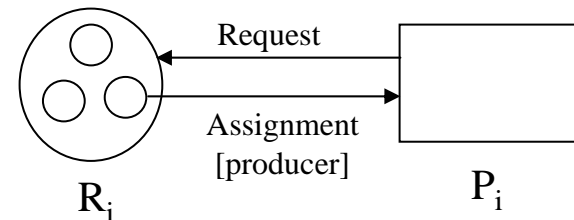


Deadlock analysis using the GRG

- The GRG is a powerful theoretical tool
 - simplified view of the state of the system
 - theorems are available that link 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

The Graph Reduction Method

- Reduction of a graph by a process P_i (unblocked):
 - for each reusable resource R_j
 - delete all request and assignment edges from/to P_i
 - increase r_j count by one for each assignment edge deleted
 - for each consumable resource R_j
 - delete all request and production edges from/to P_i
 - decrease r_j count by one for each request edge deleted
 - if P_i is a producer of R_j then set r_j to ∞



Theorems on GRGs

Good ...

- “A system state is deadlock free if its general resource graph is completely reducible”
 - Note: a general resource graph is *completely reducible* if a sequence of reductions deletes all edges in the graph

Best ...

- “If there is only a single unit of every resource, then a *cycle* in an expedient resource graph is a **necessary and sufficient** condition for a deadlock”

Better ...

- “In a general resource graph:
 - a cycle is a necessary condition for a deadlock
 - if the graph is expedient, then a knot is sufficient condition for a deadlock”
 - Note: A state is an **expedient state** if all processes having outstanding requests are blocked

Table of deadlock theorems

	Deadlock freedom conditions	Deadlock presence conditions
General case	$CR \Rightarrow DLF$	$DL \Rightarrow Cy$ $Kn, Exp \Rightarrow DL$
Reusable resources	$CR \Leftrightarrow DLF$	
Reusable + single unit resources		$Cy, Exp \Leftrightarrow DL$
Single unit requests		$Kn, Exp \Leftrightarrow DL$
Consumable resources	Claim-limited Graph $CR \Rightarrow DLF$	

Note: CR = "GRG is Completely Reducible" -- DLF = "State is Deadlock Free" -- DL = "State is deadlocked"
 Cy = "GRG contains a Cycle" -- Kn = "GRG contains a Knot" -- Exp = "GRG is Expedient"