**The Deadlock Problem:**

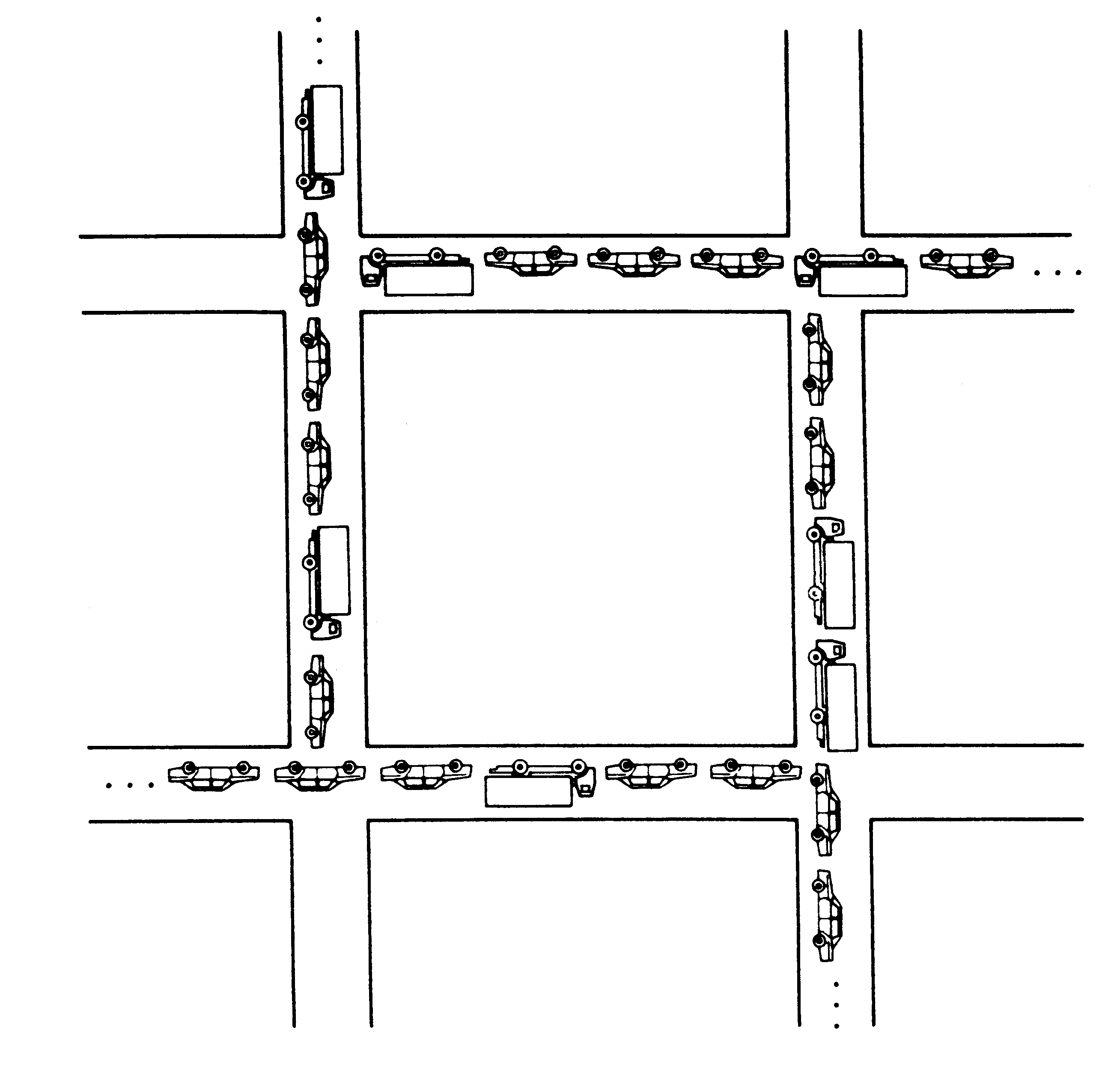
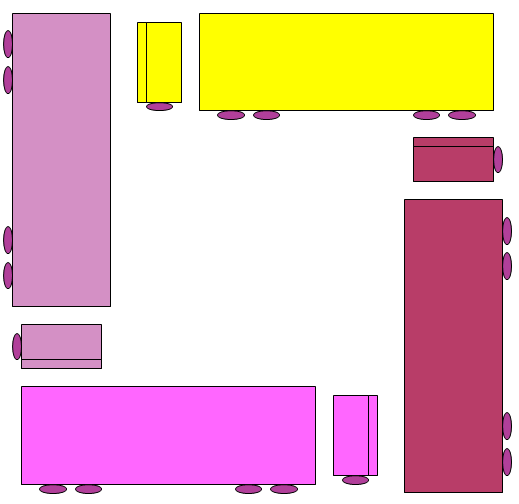
A set of blocked processes each holding a resource and waiting to acquire a resource that held by another process in the set. For example:

* System has 2 disk drives, P1 and P2 each one hold disk drive and each needs another one.
* semaphore A and B ,initialized to 1

P0 P1

wait(A) wait(B)

wait(B) wait(A)

**Real World Deadlocks**

**Bridge Crossing Problem:**

Traffic only in one direction, each section of a bridge can be viewed as a resource. If a deadlock occurs, it can be resolved if one car backs, so the Starvation state is possible.

**System Model**

Resource types R1,R2,…..,Rn, as a CPU cycles, memory space, I/O devices. Each resource type has Wi instances. Each process utilizes a resource as follows:

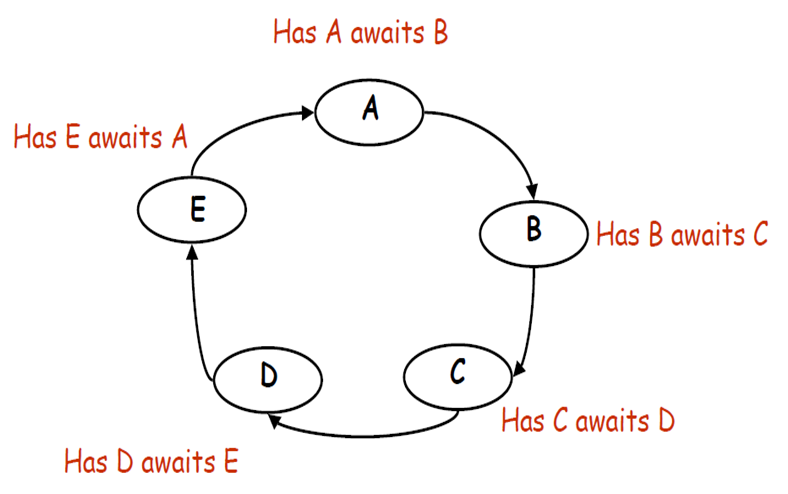
1. Requesting resource.
2. Using the resource.
3. Releasing the resource.

**Deadlock Characterization**

Deadlock can arise if four conditions hold simultaneously:

* **Mutual exclusion:** only one process at a time can use a resource.
* **Hold and wait:** a process is holding at least one resource and waiting to acquire additional resources held by another processes.
* **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task.
* **Circular wait :** there exists a set {P0,P1,….,P0} of waiting processes such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2,……, Pn-1 is waiting for a resource that is held by P0.

**Wait for cycle:**



**Resource Allocation Graph:**

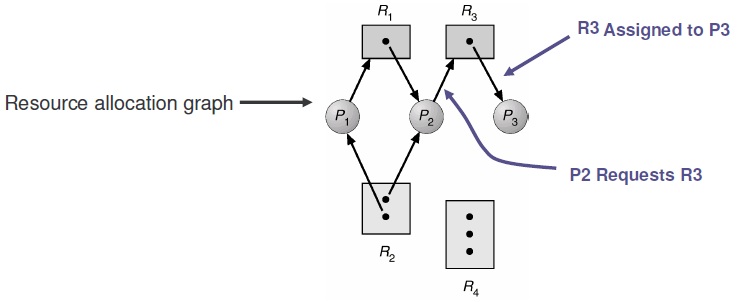
A set of vertices V and a set of edges E.

* V is partitioned into two types :
  + P = {P1, P2, …., Pn}, the set consisting of all the processes in the system.
  + R = {R1, R2, …, Rm}, the set consisting of all resource types in the system.
* E is partitioned into two types :
  + Request edge – directed edge Pi 🡪 Rj.
  + Assignment edge – directed edge Rj 🡪 Pi

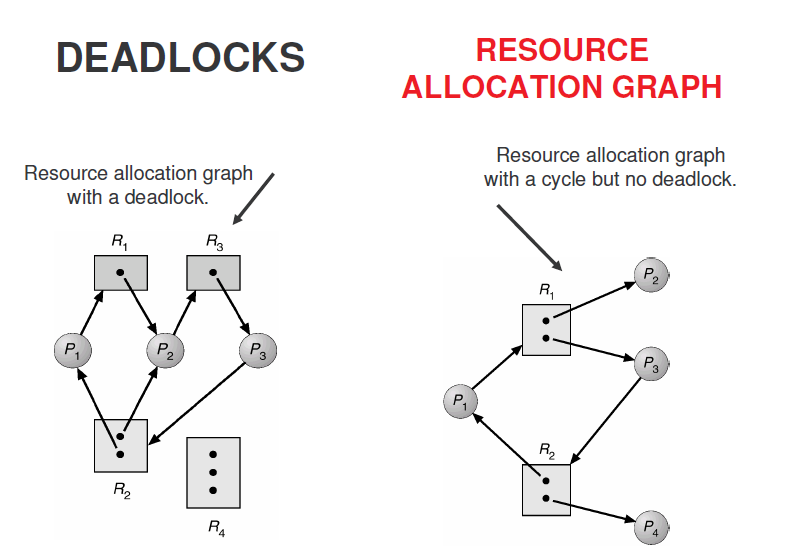
**Examples:**

* Process Pi

* Resource type with 4 instances Rj
* Pi requests instance of Rj
* An instance of Rj is assigned to Pi
* Resource Allocation Graph example.



* Another Resource Allocation Graph example.



**Strategies for Handling Deadlock**

We have three main strategies to ensure deadlock never occurs, these strategies as the following**:**

1. **Prevention:** Prevent any one of the 4 conditions from happening.
2. **Avoidance:** Allow all deadlock conditions, but calculate cycles about to happen and stop dangerous operations.
3. **Allow deadlock to happen:** This requires using bothDetection and Recovery method, where **Detection** is used **to** know a deadlock has occurred, and **Recovery** is used toregain the resources.