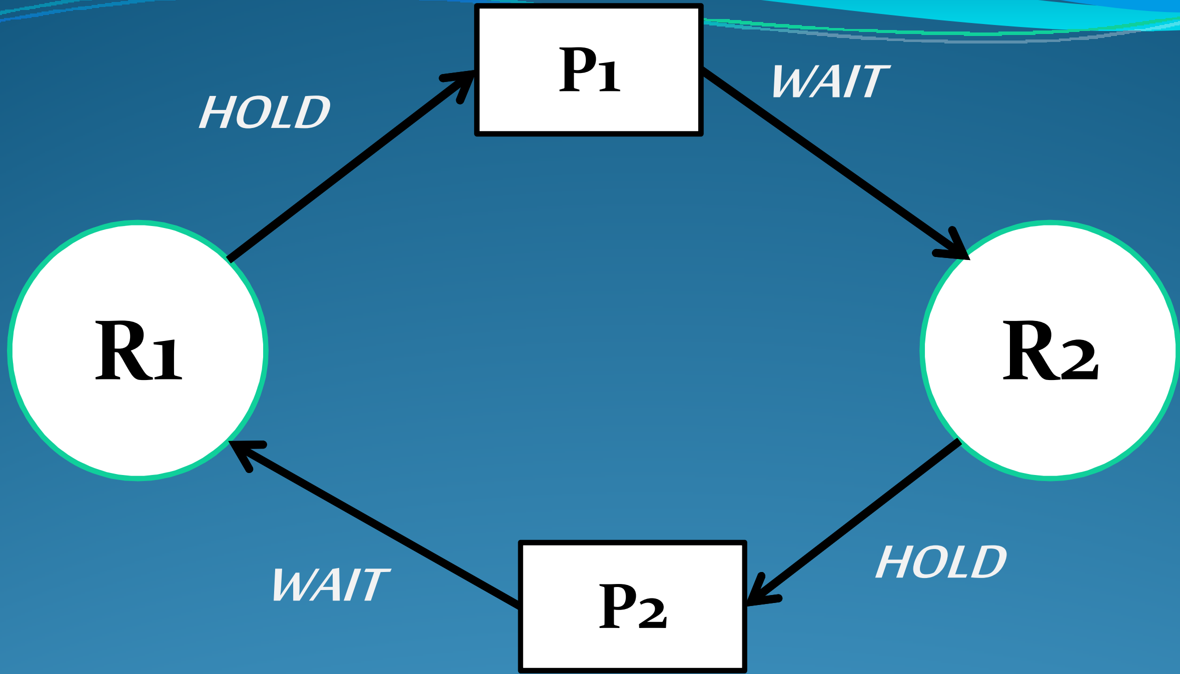
DEADLOCK

- *In an operating system, a deadlock occurs when a process enters a waiting state because a requested system resource is held by another waiting process, which in turn is waiting for another resource held by another waiting process.*
- *The process goes to the infinitely wait state.*
- ***Conditions for the cause of Deadlock:***
  - ✓ ***Mutual Exclusion***
  - ✓ ***Hold and Wait***
  - ✓ ***Non preemptive***
  - ✓ ***Circular wait***

- *In the above example, Dhawan is waiting for Cheyur to come to canteen1,while Cheyur is waiting for Dhawan to come to canteen2.*
- *Finally both goes into the infinitely wait state, which leads to deadlock.*



MUTUAL EXCLUSION:

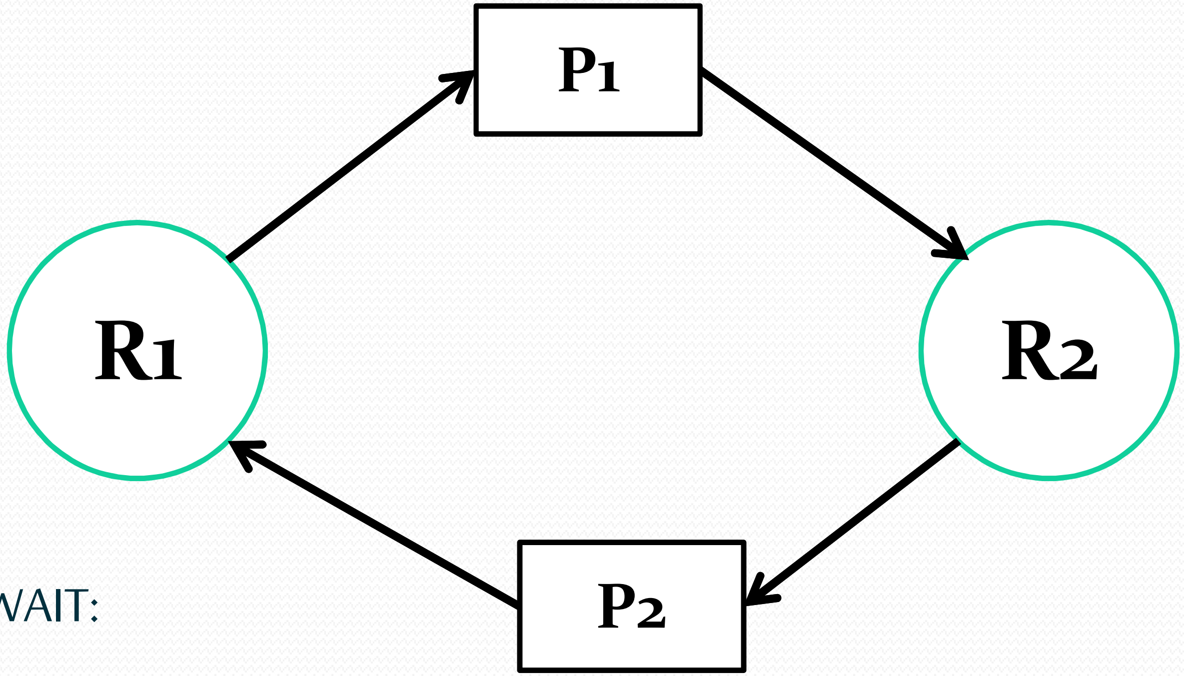
One process at a time.

HOLD AND WAIT:

A Process that holds and waits for a resource.

NON PREEMPTIVE:

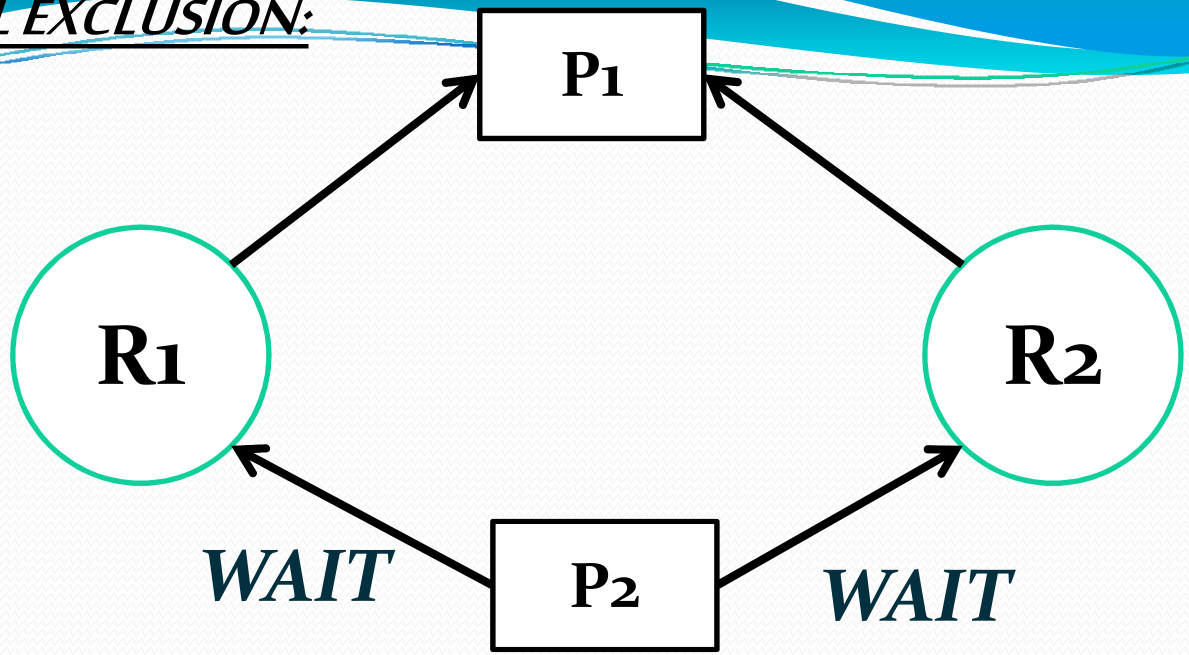
A Process never releases its resource until it complete its task.



CIRCULAR WAIT:

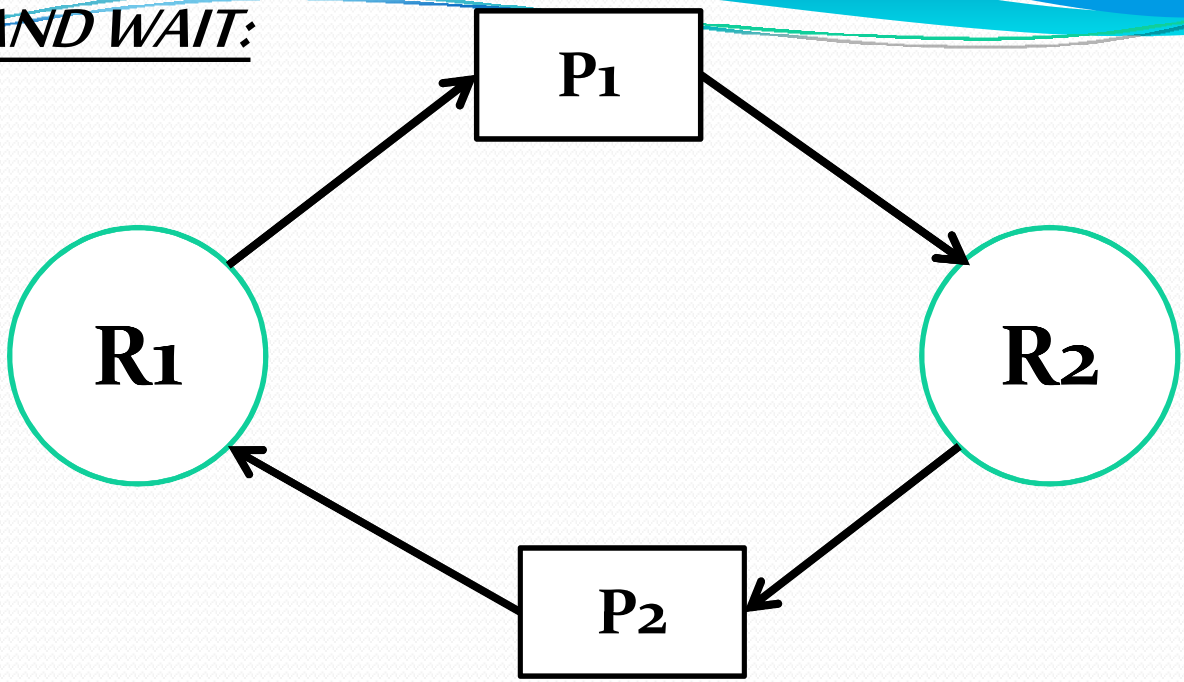
If every process of a system is waiting for another process and creates a cycle of waiting state.

MUTUAL EXCLUSION:



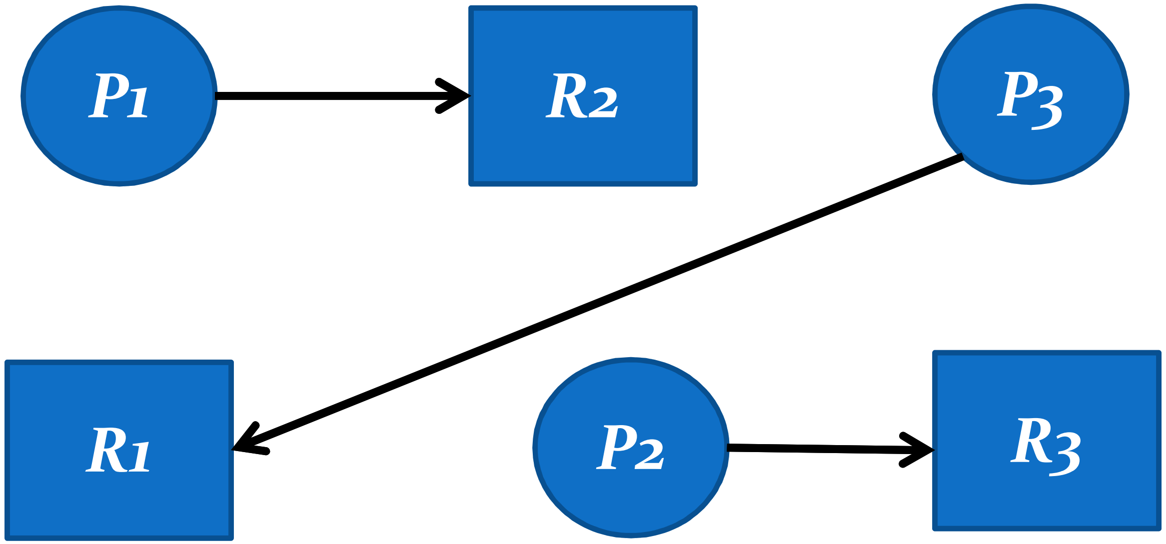
- Both P1 and P2 request for R1, and the permission is granted for P1.
- So P2 goes to wait state. The similar case happens for R2.
- After P1 complete its task, it releases R1 & R2 for the process P2.
- Since there is mutual exclusion, it doesn't mean that the system enters deadlock.

HOLD AND WAIT:



❖ *When P1 releases R1 for P2, but still the R2 holds P1 and waits for P2, then it doesn't mean that the system goes into deadlock.*

NON-PREEMPTIVE:



- ***The above condition is in Non-preemptive and each process holds different resources, but it doesn't enters the deadlock condition.***

- ***Before we go to the Circular wait condition, we need to understand two basic things:***

✓ ***To prevent the system that goes into deadlock &***

✓ ***To detect the system is in deadlock or not,***

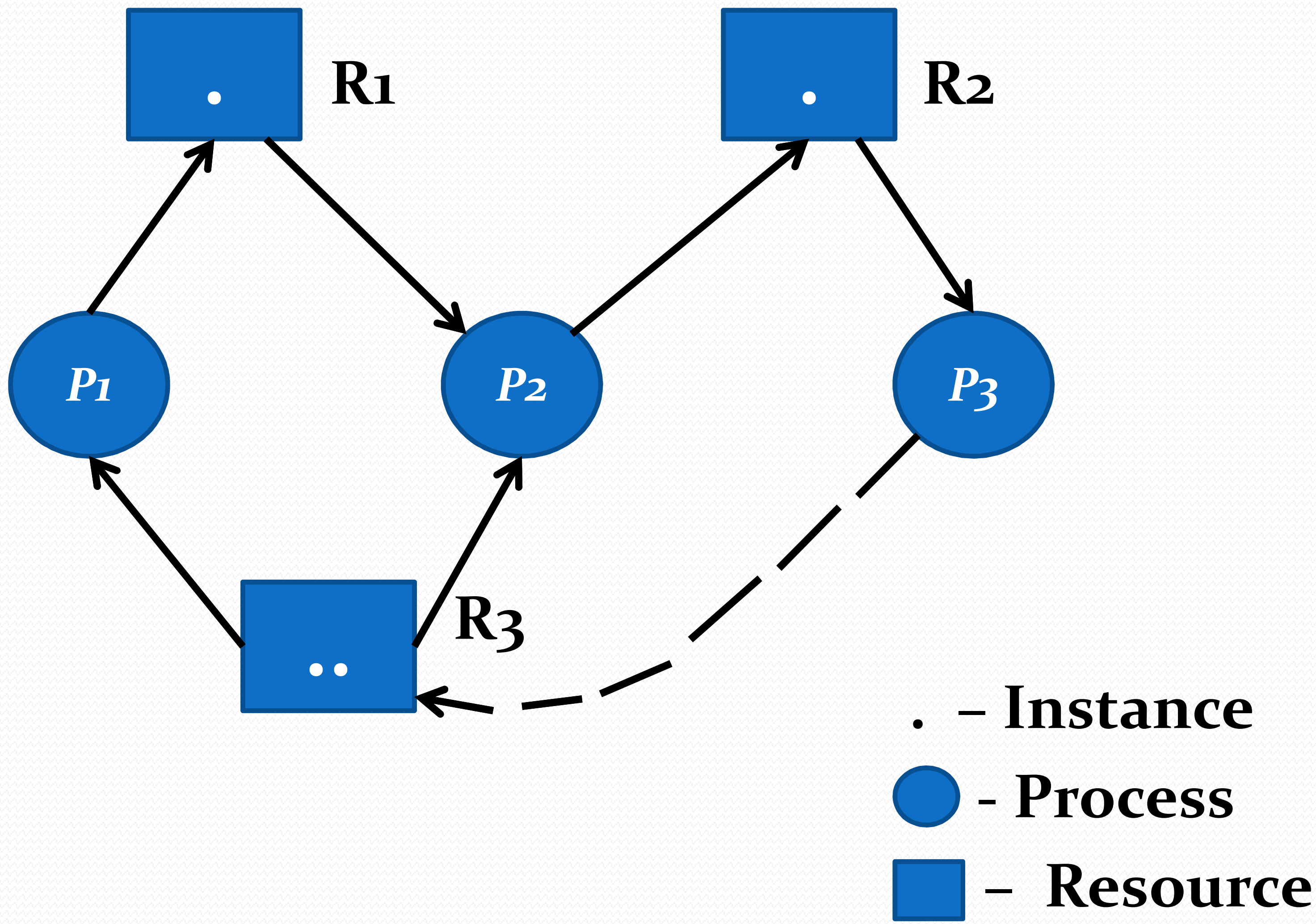
we needed several algorithms which took higher time and space complexity.

- ***Hence we prefer the Graphical methods.***

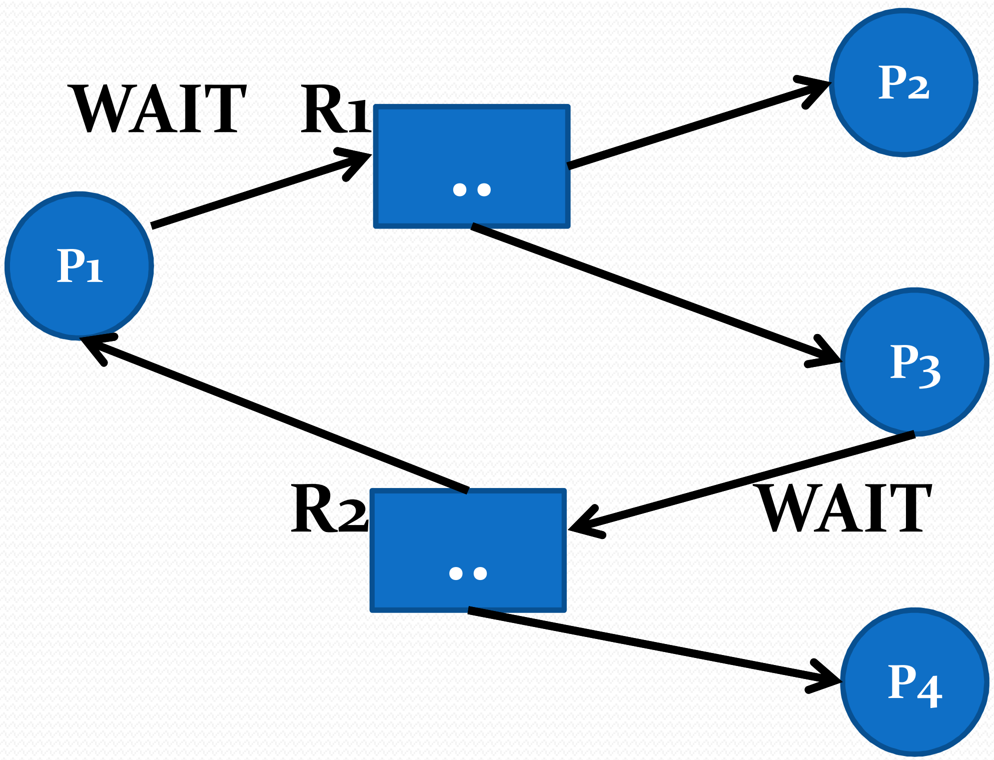
GRAPHICAL METHODS:

- ❑ ***Resource Allocation graph***
- ❑ ***wait for graph***

RESOURCE ALLOCATION GRAPH



No cycle = no deadlock ; Cycle = deadlock



Since there is a cycle it doesn't goes into deadlock.

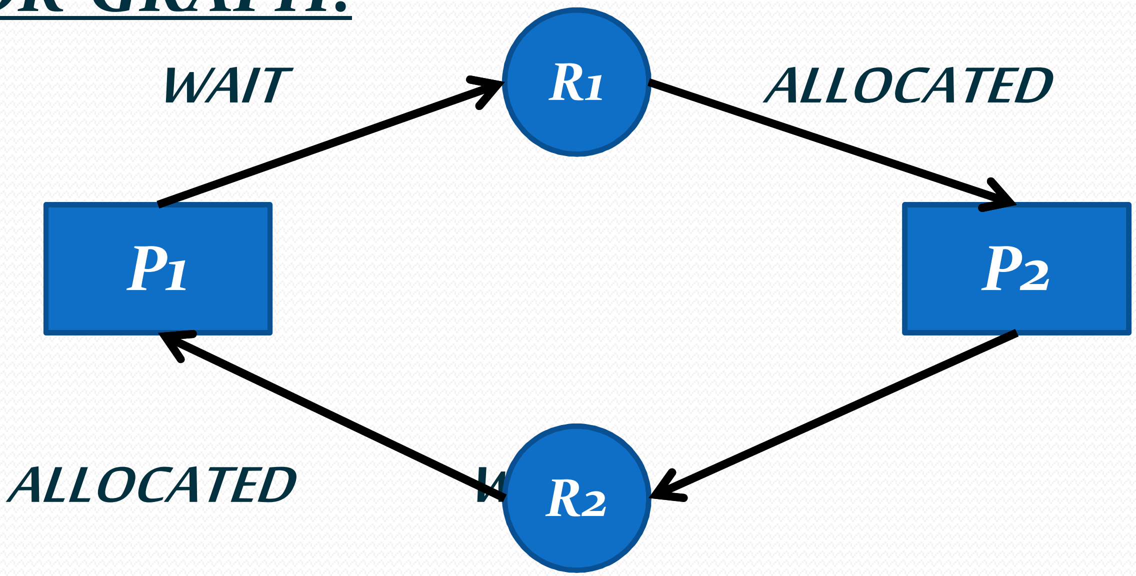
This proves that if the system has a cycle, it may lead to deadlock, but never ensure about deadlock.

RESOURCE ALLOCATION GRAPH:

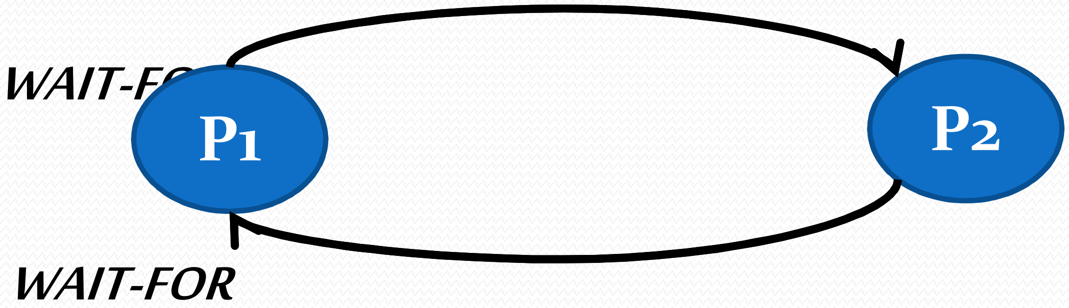
If a graph contains cycle:

- ***No. of instances = 1 , the system is in deadlock.***
- ***No. of instances > 1, the system may lead to deadlock but never ensure about deadlock.***

WAIT-FOR-GRAPH:



- ***Our aim is to find a cycle with the minimum space and time complexity.***
- ***We can say that P1 is waiting for r1 which is allocated to p2,while p2 is waiting for r2 which is allocated to p1.***
- ***we conclude that p1 is waiting for p2 and vice versa. Therefore by reducing the time and space complexity of the graph we get:***



- ***HENCE TIME & SPACE COMPLEXITY OF WAIT-FOR GRAPH < RESOURCE ALLOCATION GRAPH.***


THANK YOU