

|  |  |  |  |
| --- | --- | --- | --- |
| The graph consists of a set of vertices and a set of edges. The vertices represent all the active processes in the system {P1, P2, ..., Pn} as well as all the resource types in the system {R1, R2, ..., Rm}. The dots found in each resource square represent the number of instances of that resource. A directed edge from process Pi to resource type Rj (Pi →Rj) signiﬁes that Pi has requested an instance of Rj and is currently waiting for that resource. A directed edge from resource type Rj to process Pi (Rj → Pi) signiﬁes that an  instance of Rj has been allocated to Pi. |  | | |
|  |  |  |
|  | | |

**Resource-Allocation Graph:**

❖***About the graph:***

A process releases its allocated resources only when it acquires all the resources it needs (all its requests are fulfilled). So, if the graph contains no cycles, no process in the system is deadlocked. Otherwise, a deadlock may exist. If each resource type has exactly one instance, then a cycle implies that a deadlock has occurred. On the other hand, if each resource type has several instances, a cycle doesn’t necessarily imply that a deadlock has occurred. In the previous figure, a cycle exists; however, there is no deadlock. This is because process P4 can release its instance of resource R2. That resource can then be allocated to P3, breaking the cycle.

❖***What you are required to do:***

You are required to make aprolog program that determines whether a graph can reach a safe state (with no deadlock) or not and if the graph can reach a safe state, you should output the process execution order that made the graph be in such a state. You will have to define the graph data as facts and define the necessary predicates/rules that detect deadlocks. **For simplicity, assume we only have 2 resource types (R1 and R2).**

❖***A sample test case:***

This test case shows the sample input that should be entered and the sample output for the graph in the previous figure.

?- safe\_state(X).

X = [p2, p4, p1, p3].

Notice that that graph has different solutions.

❖***The input file:***

The input file should look somewhat like the opposite figure, but you can adjust the facts/predicates if you need to.

