A GEOMETRIC VIEW OF MUTUAL EXCLUSION AND DEADLOCK IN COMPUTER SYSTEMS

Behrooz Parhami
Computer Science Department
University of Waterloo
Waterloo, Ontario, Canada N26 3G1
on leave from
Sharif University of Technology
Tehran, Iran

Abstract

A set of eight simple diagrams with accompanying explanations has proved quite effective as a tool for teaching the concepts of mutual exclusion and deadlock in an operating systems course. This note presents the diagrams in a manner suitable for direct reproduction as viewgraphs or class handouts and touches upon the major points in explaining the diagrams to students.

1. Processes Sharing a Single Resource

Consider two processes P_1 and P_2 which need the shared resource R. Let us use the horizontal and vertical axes for representing the advancement states (e.g., percentage of completion) for P_1 and P_2, respectively. In addition, let us assume that process P_i needs the resource R from the advancement state r_i up to the advancement state r'_i (Figure 1).

Let the point p, with coordinates x_1 and x_2, represent the advancement states of P_1 and P_2 at a particular instant. Assuming that a process must have exclusive control of a resource in order to use it, the point p can never lie inside the shaded rectangle. This area is appropriately called "the unfeasible region."

On a uniprocessor, the processes P_1 and P_2 are executed one at a time. Thus, the joint advancement state for P_1 and P_2 moves on a staircase-like path (Figure 2). Note that the two axes do not represent time and that under different scheduling policies, either P_1 or P_2 may reach its corresponding r_i point first.

The interval between r_i and r'_i is the critical region of P_i. If P_1 is in its critical region, P_2 is not allowed to enter its critical region (advance past r_2), and vice versa. Given a suitable resource allocation and deallocation mechanism, the joint advancement path will find its way around the unfeasible region in a natural way and no problem arises. The path will lie below or above the unfeasible region, depending on whether P_1 or P_2 enters its critical region first. This mode of operation in the allocation of a shared resource is called "mutual exclusion."

On a multiprocessor, the processes P_1 and P_2 can advance simultaneously. Our geometric representation can handle this case as well, the only difference being that the joint advancement path no longer consists of only horizontal and vertical line segments. Again, with proper resource management, the path finds its way around the unfeasible region automatically (Figure 3).

We can easily generalize Figures 2 and 3 to the case of three processes sharing a single resource. Here, the unfeasible region becomes a rectangular volume, with the joint advancement path (a three-dimensional curve) finding its way around it.

2. Multiple Shared Resources and Deadlock

Now, let us assume that two processes P_1 and P_2 each need the two shared resources R and S, with P_i requiring the use of R from r_i to r'_i and S from s_i to s'_i. Figure 4 shows that when the two unfeasible regions are non-overlapping, the joint advancement path can pass above, below, or between the regions, without presenting any problem.

We can define the critical region of P_i with respect to each resource, as shown in Figure 4. However, an attempt to allocate resources individually based on the mutual exclusion principle would be unwise. Figure 5 shows why. Here, the unfeasible regions overlap in such a way as to create a dead-end corner D. If the joint advancement path ever enters the rectangle ABCD, deadlock at point D will be unavoidable. This rectangle is appropriately called "the unsafe region."

If the joint advancement path happens to go through point N in Figure 5,

deadlock will not occur. Thus, deadlock is a potential danger which can be avoided by proper resource management. Let us look at three deadlock avoidance strategies:

a. Allocating all resources needed by a process before it is allowed to start execution. This is clearly wasteful of resources, since some of the resources may be needed for relatively short periods of time during the process' life.

b. Assigning a unique number to each resource and requiring that a process request needed resources in ascending numerical order. Figure 6 shows that if R is always allocated to a process before S, no deadlock can occur. Judicious numbering of resources can reduce the amount of waste.

c. Preventing the joint advancement path from entering an unsafe region. A rather involved and time consuming algorithm must be applied every time a resource is allocated to ensure that unsafe boundaries are not crossed.

Another philosophy is to let the deadlock occur and then use a deadlock recovery procedure entailing rollback or restart of one or more of the processes. This approach is attractive when rollbacks or restarts are possible (they re impossible for processes with irreversible effects) and the deadlock probability is sufficiently low to justify the wasted time and resources.

In the case of n processes, the joint advancement path will be a curve in the n-dimensional space, with n-dimensional unfeasible and unsafe regions. For example, with three dimensions, the deadlock point is at a corner formed by three pairwise perpendicular surfaces. The geometric visualization of this case is deferred until the end of the next section.

3. Multiple Identical Shared Resources

Occurrence of deadlock is not limited to systems in which several independent processes compete for the control of unique resources. Let us substantiate this point by an example.

Consider a computer system having 9 identical tape drives, with processes requesting and releasing drives at particular advancement states. Take as an example, the following sequences of requests and releases (negative requests) by P_1 and P_2:

P_1: +3 +1 +2 -6
P_2: +2 +1 +2 +1 -3 -3

This situation is depicted in Figure 7, with the sum of needs shown for all combinations of advancement steps. Clearly, the unfeasible region corresponds to those combinations of advancement steps where the sum of needs exceeds 9 tape drives. We see from Figure 7 that if P_1 and P_2 are allocated 4 and 5 drives, respectively, the system state is unsafe and deadlock will occur with the next request by either process.

Now, consider three processes with identical sequences of requests and releases in a system with 9 tape drives:

P_i: +3 +3 -6

Here, an unsafe boundary is crossed whenever two processes hold 3 drives each and the third process' request for 3 drives is honored. Figure 8 depicts the unsafe region as a rectangular volume, with the deadlock point D at one of its corners.

Detection of unsafe boundary crossings is straightforward in the case of identical shared resources. A particular allocation is safe if the largest remaining needs of an individual process is not more than the number of free resources after the proposed allocation is completed.

When n processes use m identical shared resources, with a maximum need of k resources per process, the condition $n(k-1) < m$ guarantees deadlock-free operation. The geometric representation of this situation for n = 3 is left as an exercise.

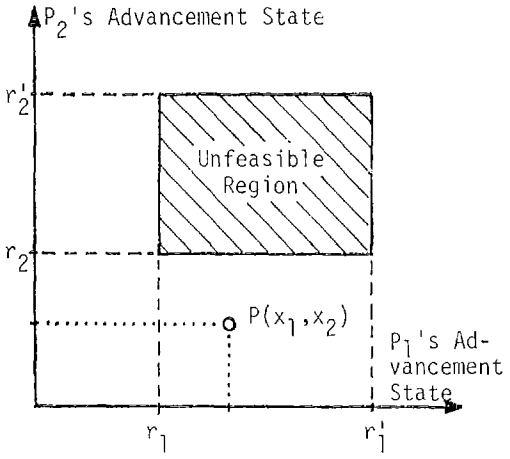

Figure 1. Geometric Representation of Two Processes Using a Single Shared Resource, with the Corresponding Unfeasible Region (Shaded Area).

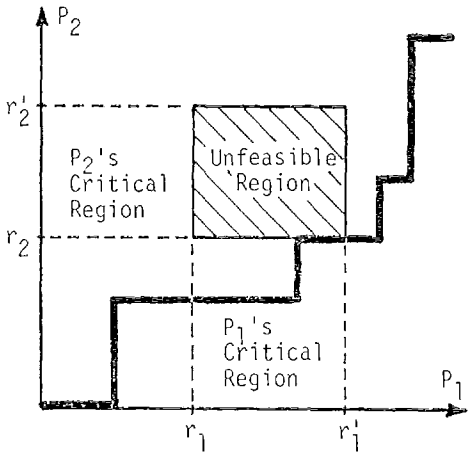

Figure 2. The Joint Advancement Path of Two Processes Using a Single Shared Resource on a Uniprocessor.

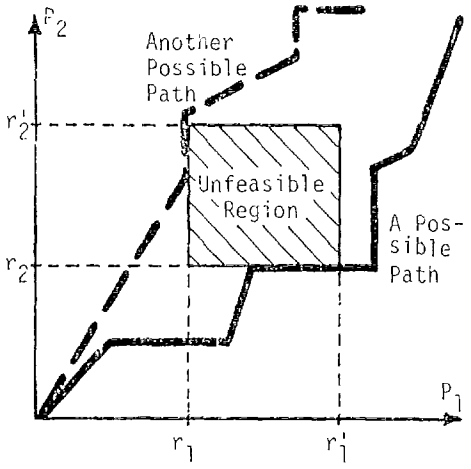

Figure 3. The Joint Advancement Path of Two Processes Using a Single Shared Resource on a Multiprocessor.

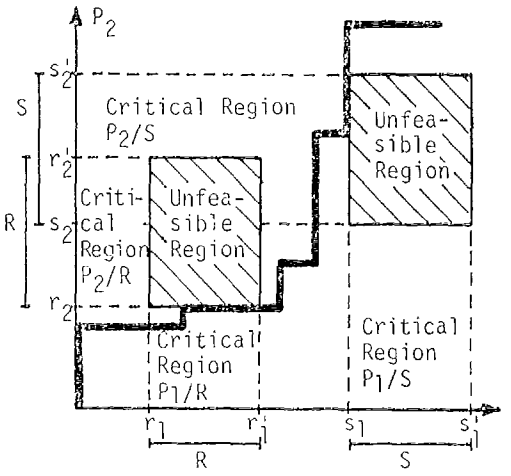

Figure 4. Non-Overlapping Unfeasible Regions for Two Processes Sharing Two Different Resources with No Deadlock.

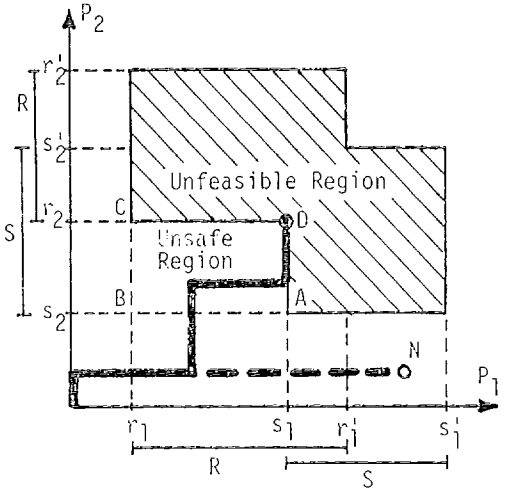

Figure 5. Overlapping Unfeasible Regions for Two Processes Sharing Two Different Resources with Possible Deadlock.

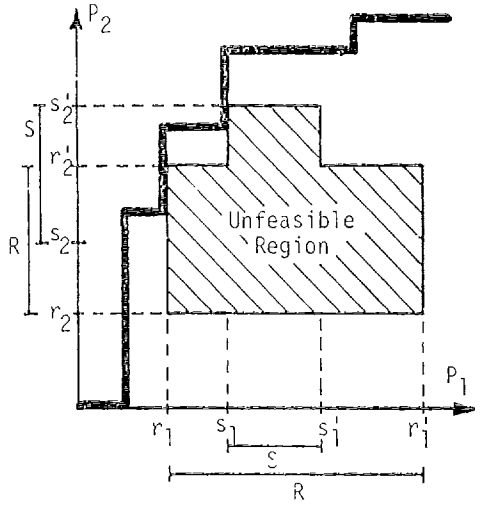

Figure 6. Overlapping Unfeasible Regions for the Case of Resource Allocation in Ascending Numerical Order.

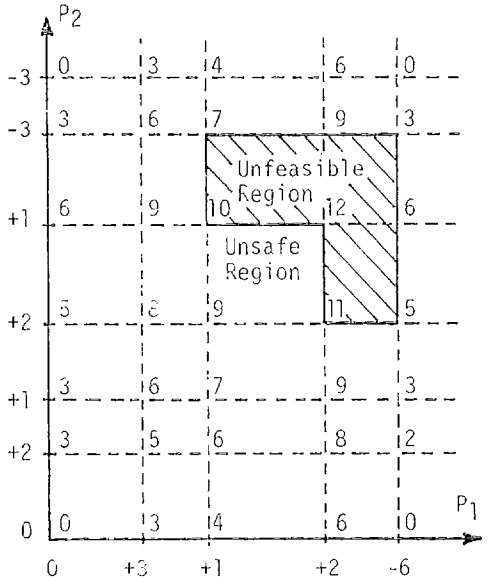

Figure 7. Possible Deadlock for Two Processes Sharing Identical Resources (9 Tape Drives).

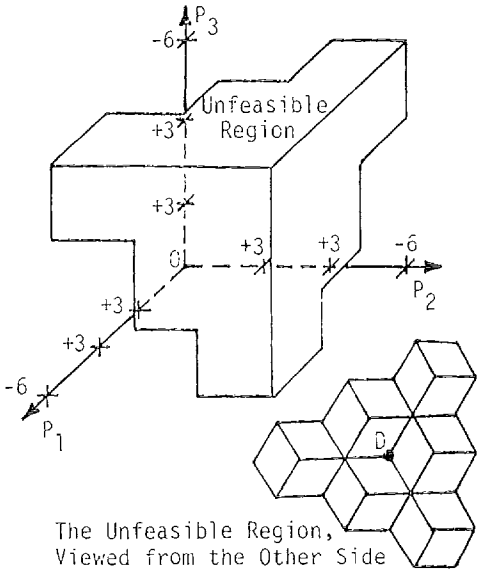

Figure 8. Possible Deadlock for Three Processes Sharing Identical Resources (9 Tape Drives).