AN ABSTRACT OF THE THESIS OF

Shahla Saidi Baghgandomi for the degree of Master of Science in Computer Science presented on August 11, 1982

Title: PROBABILITY OF DEADLOCK IN A SYSTEM OF PROCESSES AND RESOURCES FOR SHARED AND EXCLUSIVE ACCESS

Abstract approved: Redacted for Privacy

Dr. T. G. Lewis

A process-resource graph is a directed graph with m resource nodes and n process nodes. A request edge is directed from a process to a resource. An assignment edge is directed from a resource to a process. A cycle in the process-resource graph is a necessary and sufficient condition for a deadlock to exist.

This paper presents a method to find the probability of deadlock for shared access (read only) and exclusive access (read/write). Two formulae are given, one to count the total number of graphs, and another to count the total number of graphs which have at least one cycle. The probability of deadlock is then calculated by dividing the total number of graphs into the number of graphs with cycle.

Probability of Deadlock in a System of Processes and Resources for Shared and Exclusive Access

by

Shahla Saidi Baghgandomi

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed August 11, 1982

Commencement June 1983

APPROVED:

Redacted for Privacy

Associate Professor of Computer Science in charge of major

Redacted for Privacy

Chairman of the Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented August 11, 1982

Typed by Opal Grossnicklaus for Shahla Saidi Baghandomi

TABLE OF CONTENTS

LIST OF FIGURES

PROBABILITY OF DEADLOCK IN A SYSTEM OF PROCESSES AND RESOURCES FOR SHARED AND EXCLUSIVE ACCESS

I. INTRODUCTION

The term "resource" is commonly applied to a reusable, relatively stable, and often scarce commodity which is successively requested, used, and released by processes during their activity [6]. A process may have only one outstanding resource request at a time, but it can access as many resources as desired. It is also possible for a process to request a resource exclusively or to share it with other processes.

By having the above properties and using the process-resource table, probability of deadlock can be computed for a system of m resources and n processes.

Deadlock studies are important because dynamic resource sharing, parallel programming, and communicating processes are expected to be operating characteristics of many medium- and large-scale systems of the future at all levels of systems and user programs. The deadlock will increase accordingly [6]. The problem can also be critical in a number of real-time applications, such as computer control of vehicles or monitoring and control of life-supporting system, e.g., during human surgery. So far very few researches have been done to study the theoretical performance of probability of deadlock. This paper provides a method to compute

probability of deadlock, and shows how it varies as the number of processes or resources increases in the system.

1. Process-Resource Graph

A directed graph is defined as a pair <N,E> where N is a set of nodes and E is a set of ordered pairs (a,b), a,bϵN, called edges. A process-resource graph is a directed graph with the following interpretation [6]:

1. N is divided into two mutually exclusive subsets, a set of process-nodes P = $\{P_1,P_2,\ldots,P_n\}$ and a set of resource nodes R = $\{R_1,R_2,\ldots,R_m\}$.
2. The graph is "bipartite" with respect to P and R. Each edge eϵE is directed between a node of P and a node of R. If e = (P_i, R_j), then e is a request edge and is interpreted as a request by P_i for R_j. If e = (R_j,P_i), then e is an assignment edge and indicates an allocation of R_j to P_i.

The state of the processes and resources will represent the allocation status of the various resources in the system (free or allocated), and can be shown by a process-resource graph. For example consider a system with two processes P_1 and P_2, and two resources R_1 and R_2. Assume P_1 and P_2 have gained access to R_1 and R_2 respectively. Now let P_2 request for R_1 and P_1 request for R_2. A graphical representation of the system state can be shown as:

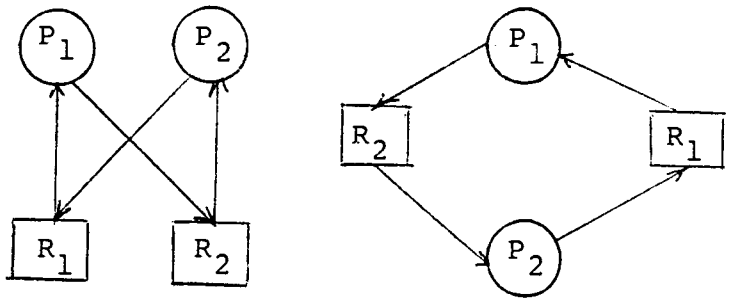

Figure 1. Process-Resource Graph

Circles represent processes, boxes represent resources. An arrow (directed edge) from a resource to a process indicates an allocation, which means the resource is being accessed by the process, while an arrow from a process to a resource is a request which means the process is waiting for the resource. Deadlock is a cycle in a process-resource graph [4]. Figure 1 graphically portrays the deadlock condition. When a deadlock occurs in a system, all the processes involved in the cycle are stopped and cannot proceed any further, unless deadlock is detected and removed. The horizontal-vertical algorithm can be used to detect a cycle and is discussed in [3,4].

2. Process-Resource Table

A process-resource graph can be represented by a table. Each row of a process-resource table is identified with a resource name, and each column of the table is identified with a process name, [4].

	P_1	P_2	. . .	P_n
R_1				
R_2				
.				
.				
.				
R_m				

Figure 2. Process-Resource Table

$$\text{Table } [I,J] = \begin{cases} -1 & \text{empty} \\ 0 & \text{allocated} \\ 1 \ldots N & \text{waiting with rank order Table } [I,J]>0 \end{cases}$$

Table [I,J] = -1 means there is no request by the process P_i for the resource R_j. Table [I,J]=0 means process P_i has gained access to resource R_j. Table [I,J]=k means P_i is the k^{th} in line for access to the resource R_j. The process-resource table for the previous example of process-resource graph is shown in Figure 3.

	P_1	P_2
R_1	0	1
R_2	1	0

Figure 3. Process-Resource Table for Figure 1.

Every time a request is made the rank of the request is entered in the table. If resource is free the request is immediately granted and a rank of 0 is entered; if the resource is not free a rank greater than 0 is entered.

II. COUNTING THE NUMBER OF GRAPHS

Total number of graphs for either shared or exclusive access can be obtained using the process-resource table.

1. Total Number of Graphs in Shared Access

Let's first consider the case of shared access (read only). We are looking for all distinct tables with mn entries which should be filled in with 0's, 1's and -1's subject to:

1. In each column, there can be at most one 1.
2. In each row, if there is a 1, there must be at least one 0.

Consider tables a, b and c as shown in Figure 4:

	P_1	P_2
R_1	0	1
R_2	0	0

"a"

	P_1	P_2
R_1	1	1
R_2	0	-1

"b"

	P_1	P_2
R_1	1	0
R_2	1	

"c"

Figure 4. Examples of Process-Resource Tables

From these three tables, "a" is acceptable, but "b" and "c" do not satisfy the above conditions and are not acceptable, ("b" has a row with all 1's in it and no 0,

"c" has two 1's in a column). For simplicity we are using 1's in place of all positive integers. This replacement has no effect on the computation.

To find all the tables with the above restrictions, let M = number of tables that can be obtained in this way.

let $c = (k_{11}, k_{12}, \ldots, k_{1q_1}, k_{21}, \ldots, k_{2q_2}, \ldots, k_{s1}, k_{s2}, \ldots, k_{sq_s})$

where

$$k_{11} = k_{12} = \ldots = k_{1q_1} = k_1$$
$$k_{21} = k_{22} = \ldots = k_{2q_2} = k_2$$
$$\vdots$$
$$k_{s1} = k_{s2} = \ldots = k_{sq_s} = k_s$$
$$k_1 > k_2 > \ldots > k_s \geq 1$$

s is the number of different groups (discussed later).

We denote a configuration with k_1 1's in each row of q_1, k_2 1's in each row of q_2, ..., and k_s 1's in each row of q's.

Let M_c = number of tables with configuration c. To form a table with configuration c, first choose the $q_1+q_2+ \ldots + q_s$ rows that contain 1's. This is the number of divisions $(q_1, q_2, \ldots, q_s, m-q_1-q_2-\ldots-q_s)$ of m, which is equal to

$$\frac{m!}{q_1!\ q_2! \cdots q_s!(m-q_1-\cdots-q_s)!}$$

and can be denoted by

$$\binom{m}{q_1, q_2, \ldots, q_s, m-q_1-\ldots-q_s}$$

Next, for each row, choose the k_i entries to put 1's in. This is the number of divisions $(k_{11}, \ldots, k_{1q_1}, \ldots, k_{s1} \ldots, k_{sq_s}, n-q_1k_1-\ldots-k_sq_s)$ of n, which is:

$$\binom{n}{\underbrace{k_1, \ldots, k_1}_{q_1}, \ldots \underbrace{, k_s, \ldots, k_s}_{q_s}, n-q_1k_1-\ldots-q_sk_s}$$

Then, for each such row, choose the remaining $n-k_i$ entries to be 0's and -1's but not all -1's (there must be at least one 0 in each of these rows). 2^{n-k_i} gives all the possible arrangements of 0's and -1's in $(n-k_i)$ places. We need to subtract one from this value to extract the case of having -1's in all $n-k_i$ entries. To get all the q_i rows we should compute $(2^{n-k_i}-1)^{q_i}$, and because there are s different groups of these rows, we need to calculate each of them separately and then multiply them together, hence:

$$(2^{n-k_1}-1)^{q_1} \times (2^{n-k_2}-1)^{q_2} \times \cdots \times (2^{n-k_s}-1)^{q_s} = \prod_{i=1}^{s} (2^{n-k_i}-1)^{q_i}$$

Last, for the remaining $m-q_1-\ldots-q_s$ rows, the entries

could be 0's and -1's. This time because there are no 1's in these rows, we also let the entries be -1's, so we get:

$$(2^n)^{(m-q_1-\ldots-q_s)} = 2^{(m-q_1-\ldots-q_s)n}$$

Hence:

2.1.1

$$M_c = \left(\begin{matrix} m \\ q_1, q_2, \ldots, q_s, m-q_1-\ldots-q_s \end{matrix}\right) x$$

$$\left(\begin{matrix} n \\ \underbrace{k_1, \ldots, k_1}_{q_1}, \ldots, \underbrace{k_s, \ldots, k_s}_{q_s}, n-q_1k_1-\ldots-q_sk_s \end{matrix}\right) x$$

$$\prod_{i=1}^{s} (2^{n-k_i}-1)^{q_i} \times 2^{(m-q_1-\ldots-q_s)n}$$

Note:

$$q_1 + q_2 + \ldots + q_s \leq m$$

$$q_1k_1 + q_2k_2 + \ldots + q_sk_s \leq n$$

To calculate M, we generate all configurations of c by assigning to c all positive integers from 1 to n, and add them together.

c=1 means there is one 1 in the table.

c=2 means there are two 1's in the table.

.

.

.

c=n means there are n 1's in the table.

We also need to consider the cases where there

are no 1's in the tables. In these cases the entries could be all -1's and 0's, so the total number of tables with no 1's is 2^{nm}. Hence the total number of tables with 1's, 0's and -1's is:

2.1.2

$$M = \sum_{\text{all } c} M_c + 2^{nm}$$

Now we need a way to calculate the number of groups for each c (s), number of rows in each group (q_i) and number of 1's in each row of q_i (k_i).
Note that each group has equal number of 1's in its rows, therefore each row of q_i has k_i 1's in it.

For example consider a table with m = n = c = 4 as below:

	P_1	P_2	P_3	P_4
R_1	1	1	-1	0
R_2	0	-1	1	0
R_3	-1	0	-1	1
R_4	-1	0	0	-1

Figure 5. A Table with Different Groups of 1's

In this example

s = 2 (group of two 1's in a row and group of one 1 in a row).

q_1 = 1 (number of rows having two 1's).

$q_2 = 2$ (number of rows having one 1).

$k_1 = 2$ (number of 1's in each row of the first group).

$k_2 = 1$ (number of 1's in each row of the second group).

We can get values of s, q's and k's for each c by using a method for partitioning integer c.

1.1. Partition of an Integer

Partition of an integer is discussed in [2,5]. A partition of c is a representation of c as a sum of integers ≥ 1. Then a representation

$$c = r_1 + r_2 + \ldots + r_s$$
$$(r_1 \geq r_2 \ldots \geq r_s)$$

is called a partition of c, where it is understood that the parts $r_1, \ldots, r_s$ are strictly positive integers.

The algorithm which we will now discuss, is given in [5] and generates from a given partition

(1) $c = r_1 + r_2 + \ldots + r_s$

its next successor on the list of all partitions of c. Suppose

(2) $c = \bar{r}_1 + \bar{r}_2 + \ldots + \bar{r}_\ell$

is the next successor of (1), ($\bar{r}_1, \ldots, \bar{r}_\ell$ are arbitrary parameters).

To determine $\bar{r}_i$ from r_i suppose first that $r_s > 1$, e.g.,

(3) $9 = 4 + 3 + 2$

To find successor of (3) in the list of partitions of 9,

we need to decrease the last part r_s by 1 and adjoining a new part = 1:

(4) $9 = 4 + 3 + 1 + 1$

Hence, the first rule for obtaining (2) from (1) is:

if $r_s > 1$

set $\bar{r}_1 = r_1$

$\bar{r}_2 = r_2$

.

.

.

$\bar{r}_{s-1} = r_{s-1}$

$\bar{r}_s = r_s - 1$

$\bar{r}_{s+1} = 1.$

Now we need to deal with the case where $r_s = 1$. Suppose

$$r_s = r_{s-1} = \ldots = r_{J-1} = 1,\ r_J > 1$$

as in the example

$9 = 4 + 3 + 1 + 1$

If the first part, 4, were to change to 3, say

(5) $9 = 3 + \ldots$

we would not have the next successor of (4) because any partition

(6) $9 = 4 + 2 + \ldots$

would lie between (4) and (5). Hence the next successor of (4) is of the form

(7) $9 = 4 + \ldots,$

and the dots in (7) constitute a partition of 5, namely, the one which is the successor of

$$5 = 3 + 1 + 1$$

in the list of partition of 5. We need to go from the last partition of 5 whose largest part is 3 to the first partition of 5 whose largest part is 2

$$5 = 2 + 2 + 1$$

The successor of (4) is then

$$9 = 4 + 2 + 2 + 1$$

If we return now to the general case, suppose we have a partition

$$c = r_1 + r_2 + \quad . \ . \ . \quad + r_j + 1 + 1 \ldots + 1$$

In the successor partition, none of the first j-1 parts will change, so

$$\bar{r}_i = r_i \quad (i<j)$$

What remains is the last partition of the number

$$c' = r_j + (s-j)$$

whose largest part is r_j, and we must go to the first partition of c' whose largest part is

$$q = r_j - 1$$

This first partition is made by repeating the part q as often as it will fit into c', namely

$$[c'/q]$$

times, and if there is a positive remainder

$$p = c' - q\,[c'/q]$$

then we adjoin one additional part equal to p.

The algorithm avoids the repeated listing of equal parts by maintaining a list $k_1 > k_2 > \ldots > k_s > 0$ of distinct

parts, and a list $q_1, \ldots, q_s$ of their respective (positive) multiplicities.

Algorithm

(A) [First entry] $k_1 \leftarrow c$; $q_1 \leftarrow 1$; $s \leftarrow 1$; Exit.

(B) [Later entries] (set σ equal to the sum of all parts of size one plus the part preceding them.) If $k_s = 1$, set $\sigma \leftarrow q_s + 1$, $s \leftarrow s-1$; otherwise, $\sigma \leftarrow 1$.

(C) [Remove one part of size k_s] $f \leftarrow k_s - 1$; if $q_s = 1$, to (D); otherwise, $q_s \leftarrow q_s - 1$; $s \leftarrow s+1$.

(D) [Add new parts of size f] $k_s \leftarrow f$; $q_s \leftarrow [\sigma/f]+1$.

(E) [Add positive remainder] $p \leftarrow \sigma \pmod f$; if $p = 0$, to (F); otherwise, $s \leftarrow s+1$; $k_s \leftarrow p$; $q_s \leftarrow 1$.

(F) [Exit] if $q_s = c$, final exit; Exit

We need to apply the algorithm to each value of c ranging between 1 and n, e.g., if $n=4$ then we use the algorithm for $c=1,2,3,4$. Therefore the algorithm is used at least n times. Each time we get different values for s, q's and k's. These can be used in the formula 2.1.1, for computing M_c, to get possible number of tables with c 1's in each of them. From that the total number of tables can be computed by using formula 2.1.2.

EXAMPLE

Let's apply the formulas and the algorithm to the case where m=n=3 and find the total number of graphs (tables).

	P_1	P_2	P_3
R_1			
R_2			
R_3			

Figure 6. A Table of 3 Processes and 3 Resources

First we consider all tables with no 1's:

$$2^{nm} = 2^{3x3} = 512$$

Now let c = 1 then

$$s = 1$$

$$q_1 = 1$$

$$k_{11} = k_1 = 1$$

Hence, using formula 2.1.1 for computing $M_{c=1}$:

$$M_1 = \frac{3!}{1!(3-1)!} \times \frac{3!}{1!(3-1)!} \times (2^{3-1} - 1)^1 \times 2^{(3-1)x3}$$

$$= 1728.$$

For c = 2 we get two partitions of c which are c = 2 = 1+1

For the partition c=2

$$s = 1$$

$$q_1 = 1$$

$$k_{11} = k_1 = 2.$$

And for the second partition c = 1+1

$$s = 1$$

$$q_1 = 2$$

$$k_{11} = k_{12} = k_1 = 1$$

Hence, substituting the above values into the formula 2.1.1 to compute M_2, we can get all tables with two 1's in them, which is:

$$M_2 = \frac{3!}{1!(3-1)!} \times \frac{3!}{2!(3-2)!} \times (2^{3-2} - 1)^1 \times 2^{(3-1)\times 3} +$$

$$\frac{3!}{2!(3-2)!} \times \frac{3!}{1!\times 1!\times(3-2)!} \times (2^{3-1} - 1)^2 \times 2^{(3-2)\times 3}$$

$$= 1872.$$

If c=3 we get three different partitions of c, which are c = 3 = 2+1 = 1+1+1.

For c = 3 then

$$s = 1$$

$$q_1 = 1$$

$$k_{11} = 3.$$

For c = 2+1 then

$$s = 2$$

$$q_1 = 1$$

$$q_2 = 1$$

$$k_{11} = k_1 = 2$$

$$k_{21} = k_2 = 1.$$

For c = 1+1+1 then

$$s = 1$$

$$q_1 = 3$$

$$k_{11} = k_{12} = k_{13} = k_1 = 1$$

Hence computing $M_{c=3}$ by using formula 2.1.1 we get:

$$M_3 = \frac{3!}{1!(3-1)!} \times \frac{3!}{3!(3-3)!} \times (2^{3-3}-1)^1 \times 2^{(3-1)\times 3} +$$

$$\frac{3!}{1!\times 1!\times(3-1-1)!} \times \frac{3!}{2!\times 1!\times(3-2-1)!} \times (2^{3-2}-1 \times$$

$$(2^{3-1}-1)^1 \times 2^{(3-1-1)\times 3} +$$

$$\frac{3!}{3!\times(3-1-1-1)!} \times \frac{3!}{1!\times 1!\times 1!\times(3-3)!} \times (2^{3-1}-1)^3 \times$$

$$2^{(3-1-1-1)\times 3}$$

$$= 0 + 432 + 162 = 594.$$

Now to get the total number of tables with 0, 1, 2, or 3 1's we can use formula 2.1.2.

$$M = \sum_{\text{all } c} M_c + 2^{nm}$$

$$= M_1 + M_2 + M_3 + 2^{nm}$$

$$= 1728 + 1872 + 594 + 512$$

$$= 4706.$$

2. Total Number of Graphs in Exclusive Access

The procedure for finding the total number of graphs in case of exclusive access (read/write) is similar to the previous case (shared access).

We would like this time to fill in the mn entries with 0's, 1's and -1's subject to:

1) In each column, there must be at most one 1,

2) In each row, if there is a 1, there must be one and only one 0.

This case is different from the other one in the number of 0's in each row. Because in exclusive access if one of the processes gained access to a resource, no other processes can have that resource until it is released.

As in the shared access let

$$c = (\underbrace{k_1,\ldots,k_1}_{q_1}, \underbrace{k_2,\ldots,k_2}_{q_2}, \ldots \underbrace{k_s,\ldots,k_s}_{q_s})$$

$$k_1 > k_2 \ldots > k_s \geq 1$$

$\bar{M}_c$ = number of tables with configuration c, with k_1 1's in each row of q_1, k_2 1's in each row of q_2, ... , and k_s 1's in each row of q_s.

To form a table with configuration c,

First, we choose the $q_1 + q_2 + \ldots + q_s$ rows that will contain 1's.

Next, for each row, let's choose the k_i entries to put 1's in.

Next, for each such row, we choose the remaining $n-k_i$ entries to have -1's with one and only one 0, which is $(n-k_i)$ possibilities for each row of q_i.

Last, for the remaining $m-q_1-q_2- \ldots -q_s$ rows. We choose each row to have at most one 0, which is (n+1), and

because there are $(m-q_1-\ldots-q_s)$ of these rows, we get $(n+1)^{(m-q_1-\ldots-q_s)}$ possibilities. Hence, the formula for the number of tables with c 1's in them in exclusive access is:

2.2.1

$$\bar{M}_c = \left(\begin{matrix} m \\ q_1, q_2, \ldots, q_s, m-q_1-\ldots-q_s \end{matrix} \right) \times$$

$$\left(\begin{matrix} n \\ \underbrace{k_1, \ldots, k_1}_{q_1}, \ldots, \underbrace{k_s, \ldots, k_s}_{q_s}, n-q_1k_1-\ldots-k_sq_s \end{matrix} \right) \times$$

$$\prod_{i=1}^{s} (n-k_i)^{q_i} \times (n+1)^{(m-q_1-\ldots-q_s)}$$

For the case where there is no 1 in the table we get $(n+1)^m$, so the total number of graphs (tables) in case of exclusive access is:

2.2.2 $$\bar{M} = \sum_{\text{all } c} \bar{M}_c + (n+1)^m$$

To find s, k's and q's we go exactly through the same procedure as in the case of shared access, by using partitions of integer c.

If we use the previous example where n = m = 3, then we get 976 as the total number of graphs for exclusive access, which is much smaller than what we get for the other case, because the accesses to a resource are restricted to one at a time.

III. NUMBER OF CYCLES IN A GRAPH

Using the process-resource table we can find the distinct number of cycles in a graph.

In cases where there are one resource and n processes (n=1, ...), or one process and m resources (m=1, ...), we don't get any cycle, because a process cannot block itself. So we start with the cases where there are two or more processes and resources.

Length of a cycle is the number of edges between processes and resources in the cycle, e.g., if n = m = 2, we get cycle of length 4 and if n = m = 3 we may get cycles of length 4 or 6 and so on, as in the figure (7).

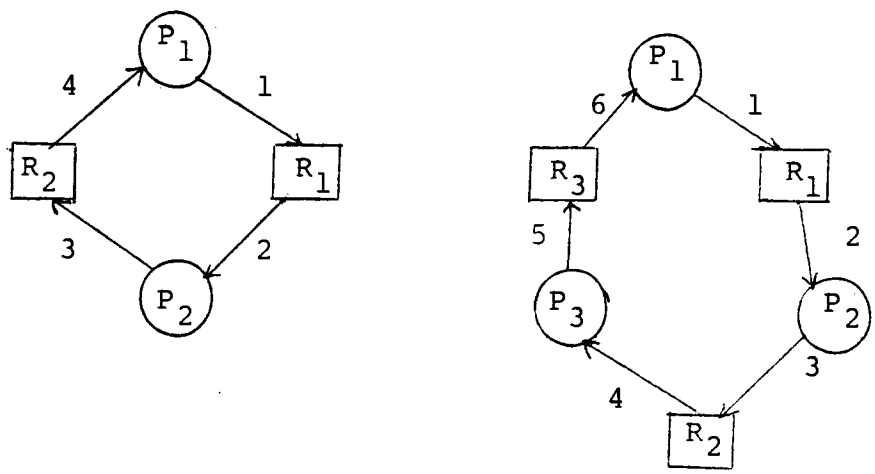

Figure 7. Cycles of Length 4 and Length 6

In a cycle, always the same number of processes and resources are involved, e.g., if n = 2, m = 5, we only get cycles of length 4, because at any instant there could be the 2 processes with 2 of the resources in

a cycle.

To find the distinct number of cycles in a system of n processes and m resources, first we have to find i-subsets of n and m ($2 \leq i \leq \min\{n,m\}$) and multiply them, which is $\binom{n}{i} \times \binom{m}{i}$, e.g., let n = 4, m = 3, then

$P = \{P_1, P_2, P_3, P_4\}$,

$R = \{R_1, R_2, R_3\}$.

Now we need to find a set of combinations of each 2 elements of P (a set of pairs) which is

$\{(P_1,P_2), (P_1,P_3), (P_1,P_4), (P_2,P_3), (P_2,P_4), (P_3,P_4)\}$

and has $\binom{4}{2} = 6$ elements, and a set of pairs of each 2 elements of R, which is

$\{(R_1,R_2), (R_1,R_3), (R_2,R_3)\}$

and has $\binom{3}{2} = 3$ elements.

Now combine the elements of these 2 sets which is

$I2 = \{(P_1,P_2,R_1,R_2), (P_1,P_2,R_1,R_3), \ldots\}$

and has $\binom{4}{2} \times \binom{3}{2} = 6 \times 3 = 18$ elements.

Similarly for i=3 we get $\binom{4}{3} \times \binom{3}{3} = 4$ elements in the final set which is

$I3 = \{(P_1,P_2,P_3,R_1,R_2,R_3), (P_1,P_2,P_4,R_1,R_2,R_3), \ldots\}$.

Now we need to find the number of possible cycles for each element of I2 and I3.

For each element of I2 we consider a 2x2 table, so that each column and row contains only one 0 and one 1. For the first column we get 2 ways to put one 0 and one 1 and for each of these cases we get only one way in the second column. So we have 2x1 ways to put 0's and 1's

in the table, which means the number of cycles is 2. The tables with a cycle are shown in figure (8).

	P_1	P_2
R_1	0	1
R_2	1	0

	P_1	P_2
R_1	1	0
R_2	0	1

Figure 8. Different Types of Cycles for 2 Processes and 2 Resources.

Hence, the total number of cycles of length 4 in this system with 4 processes and 3 resources is

$$\binom{4}{2} \times \binom{3}{2} \times 2 \times 1 = 36.$$

For elements of I3 we consider a 3 x 3 table in which for the first column there are 3 x 2 = 6 ways to put one 0 and one 1. For each of these cases there are 2 x 1 = 2 ways in the second column and for each of these 6 x 2 = 12 cases there is one way to put one 0 and one 1 in the third column. So, there are 12 x 1 = 12 ways to arrange the 0's and 1's in the 3 x 3 table to get cycles of length 6. Hence, in this system there are total of 4 x 12 = 48 cycles of length 6. Two of the tables for this case are shown in Figure (9).

	P_1	P_2	P_3
R_1	0	-1	1
R_2	1	0	-1
R_3	-1	1	0

	P_1	P_2	P_3
R_1	1	0	-1
R_2	0	-1	1
R_3	-1	1	0

Figure 9. Two Different Types of Cycles for 3 Processes and 3 Resources.

In general we form an ixi table and compute the number of ways to put 0's and 1's in its entries subject to: In each column and row there must be only one 0 and one 1.

Starting from the first column we get ix(i-1) ways to put one 0 and one 1, in the second column there are (i-1) x (i-2) ways and so on up to the i^{th} column which has one way, so there are

i x (i-1) x (i-1) x(i-2) x -----x 2x1 ways to put 0's and 1's in the table to form cycles of length 2i. Hence, the total number of cycles in the system of n processes and m resources is

$$Cyc = \sum_{i=2}^{\min\{n,m\}} \binom{n}{i} \times \binom{m}{i} \times i \times i \times (i-1) \times (i-1) \times \ldots \times 2 \times 2 \times 1$$

which can be simplified:

$$\sum_{i=1}^{\min\{n,m\}} \binom{n}{i} \times \binom{m}{i} \times i \times i \times (i-1)^2 \times \ldots \times 2^2 \times 1 \quad \text{then}$$

$$3.1 \quad Cyc = \sum_{i=2}^{\min\{n,m\}} \frac{1}{i} \binom{n}{i} \times \binom{m}{i}$$

IV. COUNTING THE NUMBER OF GRAPHS WITH CYCLE

To find the total number of graphs with at least one cycle, we consider the case of shared access and exclusive access.

1. Total Number of Graphs with Cycle in Shared Access

To find the number of graphs with cycle, first we need to get all possible tables with 1's in them, which can be done by using partition of an integer. Then we need to put 0's in those entries, which results in at least one cycle in the table. Therefore, each time a 0 is entered into a row with a 1 already in it, we check for cycle: If there is a cycle, then we count the number of 0's in the table.

Let z_i indicate the number of 0's in group i, then the formula for the graphs with at least one cycle can be written as:

$$4.1.1 \quad DL = \sum_{\text{all c's}} \binom{m}{q_1, \ldots, q_s, m-q_1-\ldots-q_s} x \binom{n}{\underbrace{k_1, \ldots, k_1}_{q_1}, \ldots, \underbrace{k_s, \ldots, k_s}_{q_s}, n-k_1q_1-\ldots-k_sq_s} x \left(\sum_{\substack{\text{all tables} \\ \text{with cycle}}} \prod_{i=1}^{s} (2^{k_i}-1)^{z_i} \times (2^{n-k_1q_1-\ldots-k_sq_s}-1)^{z_{s+1}} \right) x \; 2^{(m-q_1-\ldots-q_s)} \; xn$$

We don't add 2^{nm} to the above formula, because it is the number of tables with no 1's in them. To get tables with a cycle, it is necessary to have at least two rows with 1's in them, so we assign to c all positive integers from 2 to n and add them together. To find z's we first transfer the table to a new one, so that each row contains just one 1 in it e.g., if we have a table as in Figure 10, with m = 3, n = 7

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
R_1	1	1	-1	0	-1	-1	-1
R_2	0	-1	1	1	-1	-1	-1
R_3	-1	-1	0	-1	1	-1	-1

Figure 10. Example of a Table with Cycle.

In this example,

c = 5, number of 1's in the table.

s = 2, number of different groups.

q_1 = 2, number of rows in the first group.

k_1 = 2, number of 1's in each row of q_1.

q_2 = 1, number of rows in the second group.

k_2 = 1, number of 1's in each row of q_2.

Then we can transfer the above table to the table as in Figure 11.

	1	2	3	4
R_1	1	0	-1	-1
R_2	0	1	-1	-1
R_3	-1	0	1	-1
	q_1		q_2	

Figure 11. A Table with only one 1 in each row.

This table has (q_1+q_2) rows, and $(q_1+ q_2 + 1)$ columns. In general, when we transfer the table to a new one with only one 1 in each of its rows, then it has $q_1+....+q_s$ rows and $q_1+....+q_s$, or $q_1+....+q_s+1$ columns, depending on whether $n-k_1q_1-... -k_sq_s = 0$, or $n-k_1q_1-...-k_sq_s > 0$ respectively.

In the above example there is a cycle in the table, so we need to count the number of 0's in the table. By using the table in Figure 11 we get

$z_1 = 3,$

$z_2 = 0$ and $z_3 = 0.$

Hence, the number of ways to put 0's in the original table, Figure 10, to get cycle is:

$$(2^{k_1}-1)^{z_1} \times (2^{k_2}-1)^{z_2} \times (2^{n-k_1q_1-k_2q_2}-1)^{z_3} =$$

$$(2^2-1)^3 \times (2^1-1)^0 \times (2^{7-4-1}-1)^0 = 27$$

Then we find other tables with cycle for the above example, i.e., we may take the 0 in the third row and the second column of Figure 11, and put it in the third row and the fourth column of the table, as in Figure 12.

	1	2	3	4
R_1	1	0	-1	-1
R_2	0	1	-1	-1
R_3	-1	-1	1	0

Figure 12. Another Example of Figure 11

Now, we have another table with cycle, which we count the number of 0's, and add it to what we got before and so on. At the end, when we have the final number for all the possible tables for this example, then we can use formula 4.1.1 to compute the total number of tables with cycle for the above partition.

2. Total Number of Graphs with Cycle in Exclusive Access

In case of exclusive access, where there is one and only one 0 in each row with 1's in it, we have a new formula which is:

$$4.21.\quad \overline{DL} = \sum_{\text{all } c\text{'s}} \binom{m}{q_1, \ldots, q_s, m-q_1-\ldots-q_s} x \binom{n}{k_1, \ldots, k_1, \ldots, k_s, \ldots, k_s, n-q_1k_1-\ldots-q_sk_s} x \sum_{\substack{\text{all tables} \\ \text{with cycle}}} \prod_{i=1}^{s} (k_i)^{z_i} x (n-q_1k_1-\ldots-q_sk_s)^{z_{s+1}} x (n+1)^{(m-q_1-\ldots-q_s)}$$

We assign to c all the positive integers from 2 to n, and add them together. To find z's, number of 0's in the table, we use the same approach as in the shared access case.

3. Cycle Detection

There are many methods to detect a cycle in a system of processes and resources, but most of them are very specific and cannot apply to every situation. The method we present here is discussed in [1], and can detect all cycles of different length in a process-resource table, for both shared and exclusive access.

This method removes all those rows and columns from the table, which have only one type of entry, either 0's or 1's. At the end if the table has more than one row and column, then it indicates a cycle exists in the original table. The -1 entries are not considered in this method because they have no effect on deadlock.

```
Algorithm
    Repeat
        For all columns with only one kind of entry,
            either "0" or "1", do
            begin
                extract the column from the table
                col = col - 1
            end
        For all rows with only one kind of entry,
            either "0" or "1", do
                begin
                    extract the row from the table
                    Rw = Rw - 1
            end
    Until
        there are no more columns or rows to be
        removed from the table.
    If
        col > 1 and Rw > 1
        then
            there is a cycle in the table.
        else
            no cycle
    endif
```

V. PROBABILITY OF DEADLOCK

So far, we have presented a formula to count the total number of graphs, and another formula to count the total number of graphs with cycle. To find the probability of deadlock, we use formula 5.1., which is:

$$5.1 \quad Prob = \frac{\text{total number of graphs with cycle}}{\text{total number of graphs}} \times 100$$

In shared access

$$5.2 \quad Prob = \frac{DL}{M} \times 100$$

and in exclusive access

$$5.3 \quad \overline{prob} = \frac{\overline{DL}}{\overline{M}} \times 100$$

By implementing the formulae, we can verify how the probability of deadlock varies as the number of processes and resources increase for each case.

Experiment shows probability of deadlock increases for both cases as the number of processes and resources increase. Figure 13 shows the probability of deadlock versus the number of processes by holding the number of resources unchanged in case of exclusive access. Figure 14 shows the same experiment for shared access.

Comparing these two cases, we'll see that the probability of deadlock in shared access is higher than in exclusive access, Figures 15 and 16. Figure 17 shows the probability of deadlock versus the number of resources by holding the number of processes unchanged

in exclusive access. Figure 18 is the same experiment for shared access. Comparing these two cases, we'll see that the probability of deadlock increases more rapidly in case of shared access, Figures 19, 20 and 21. All the above experiments are done by using Appendix A, which is the program listing of the formulas and the algorithms for counting the total number of graphs, the total number of graphs with cycle, and the probability of deadlock for shared and exclusive access. These programs are written in BASIC, using HP-85 Microcomputer and its plotter.

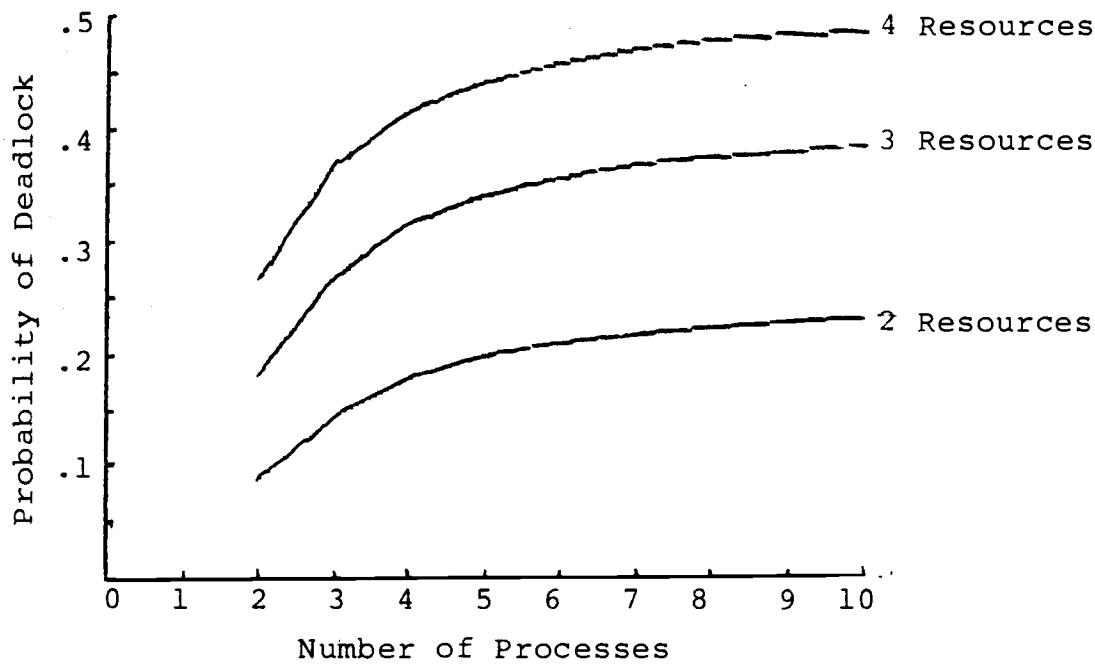

Figure 13. Probability of Deadlock for 2, 3 and 4 Resources in Exclusive Access

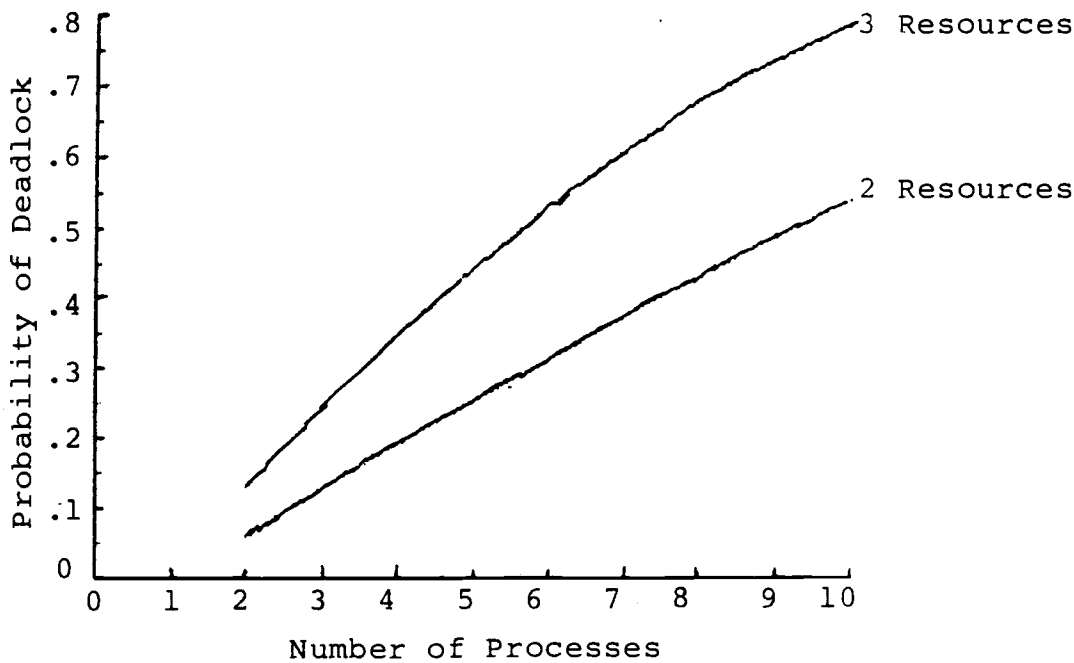

Figure 14. Probability of Deadlock for 2 and 3 Resources in Shared Access

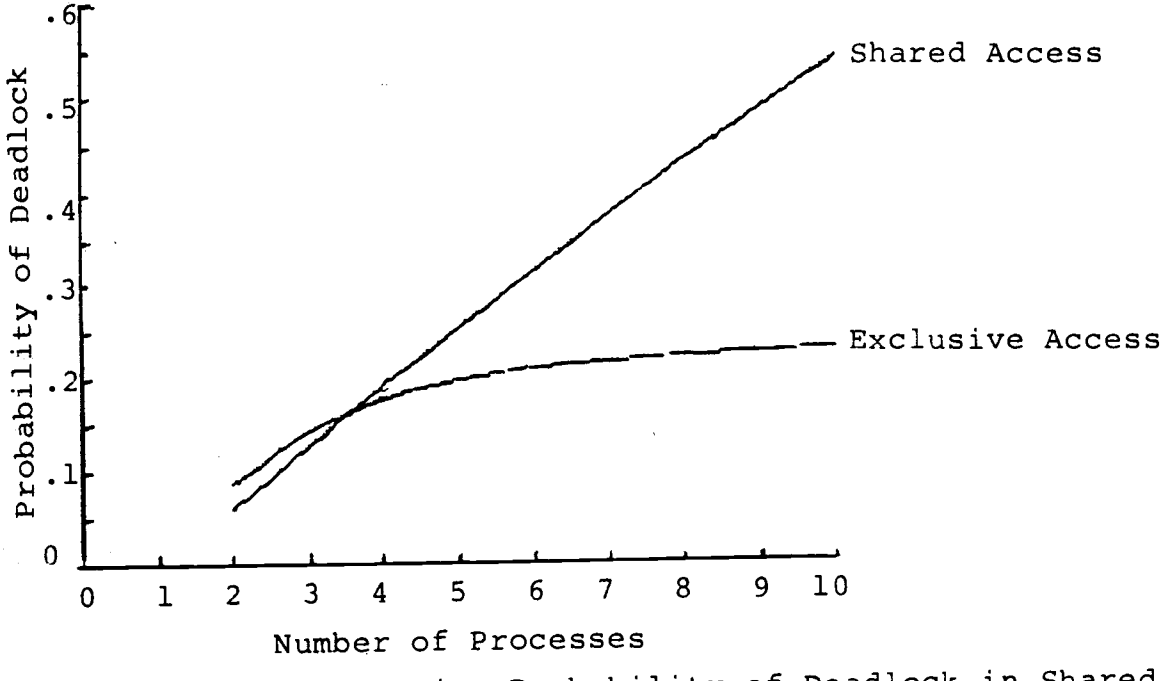

Figure 15. Comparing Probability of Deadlock in Shared and Exclusive Access for 2 Resources

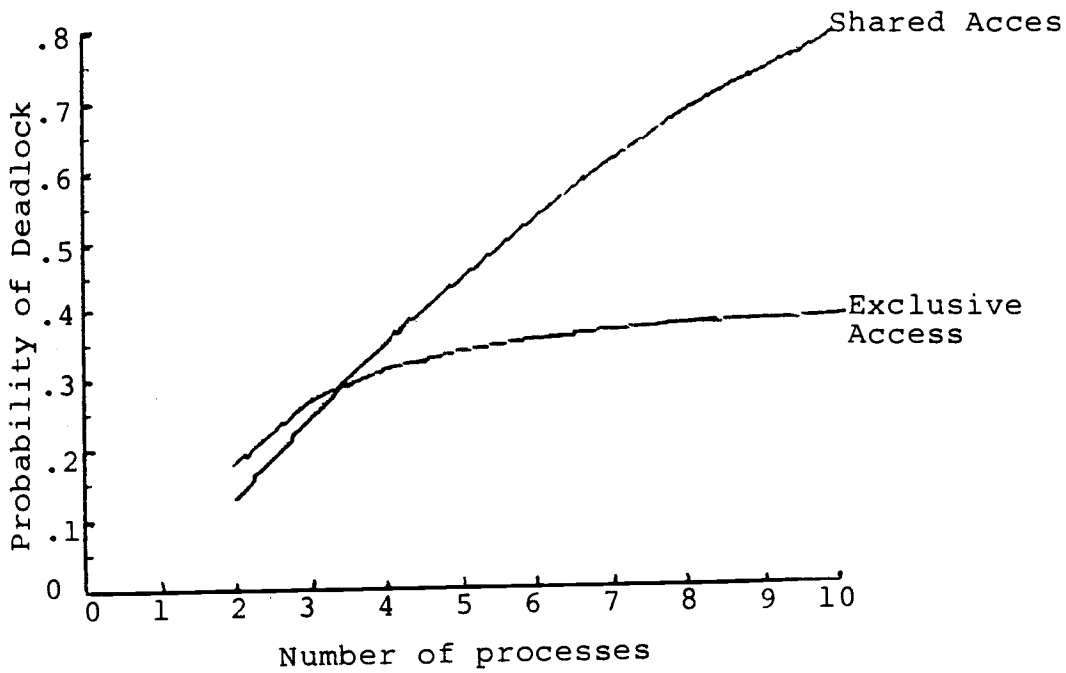

Figure 16. Comparing Probability of Deadlock in Shared and Exclusive Access for 3 Resources

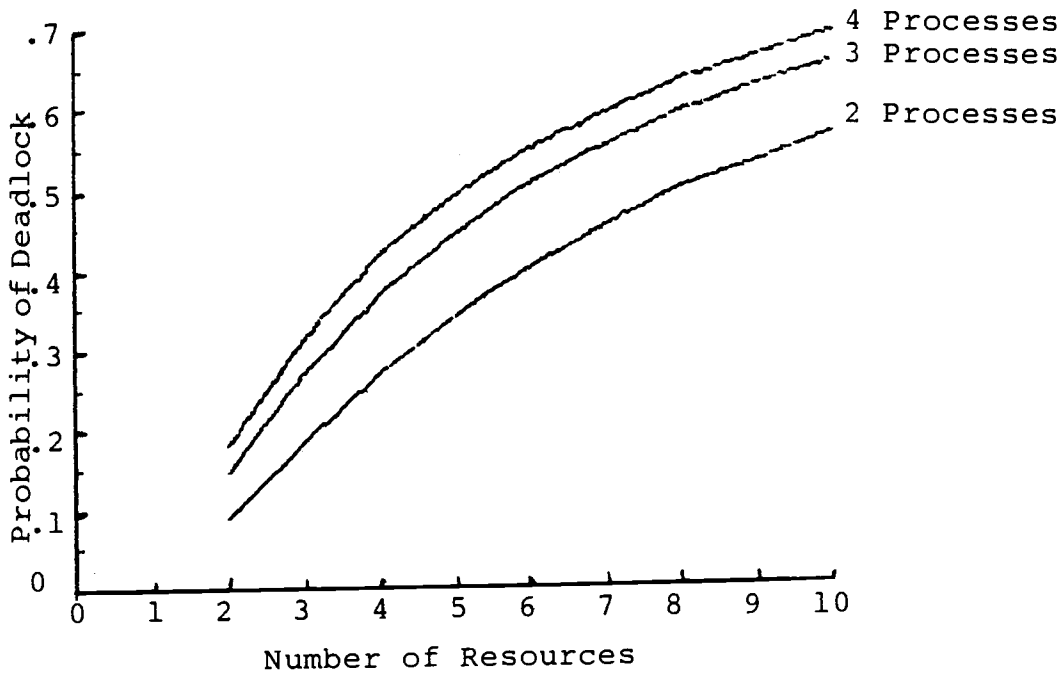

Figure 17. Probability of Deadlock for 2, 3 and 4 Processes in Exclusive Access

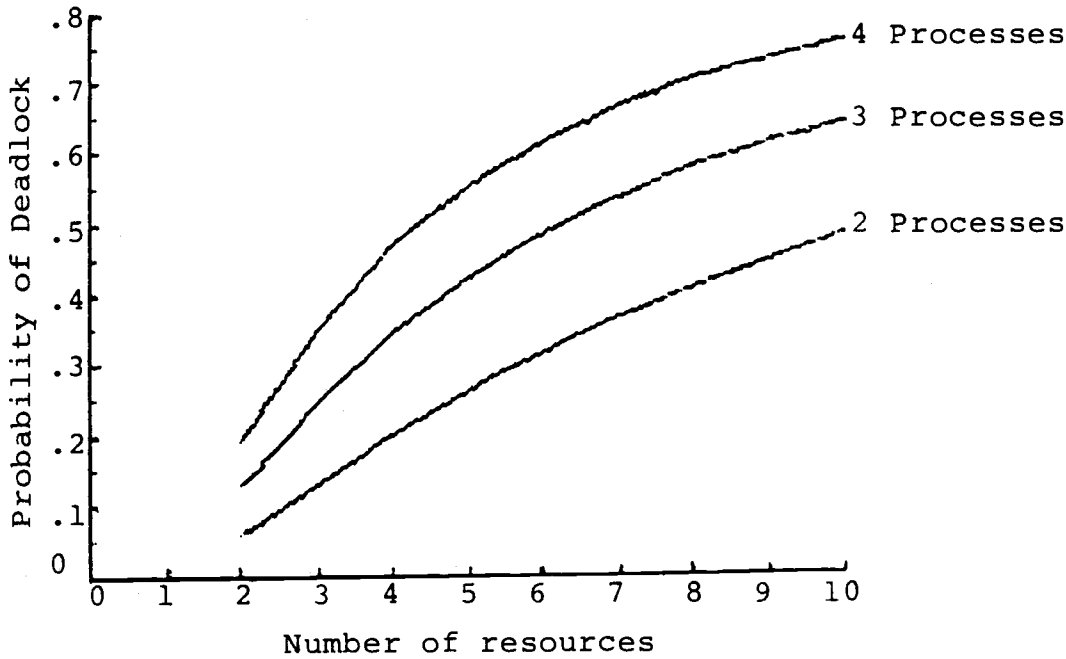

Figure 18. Probability of Deadlock for 2, 3 and 4 Processes in Shared Access

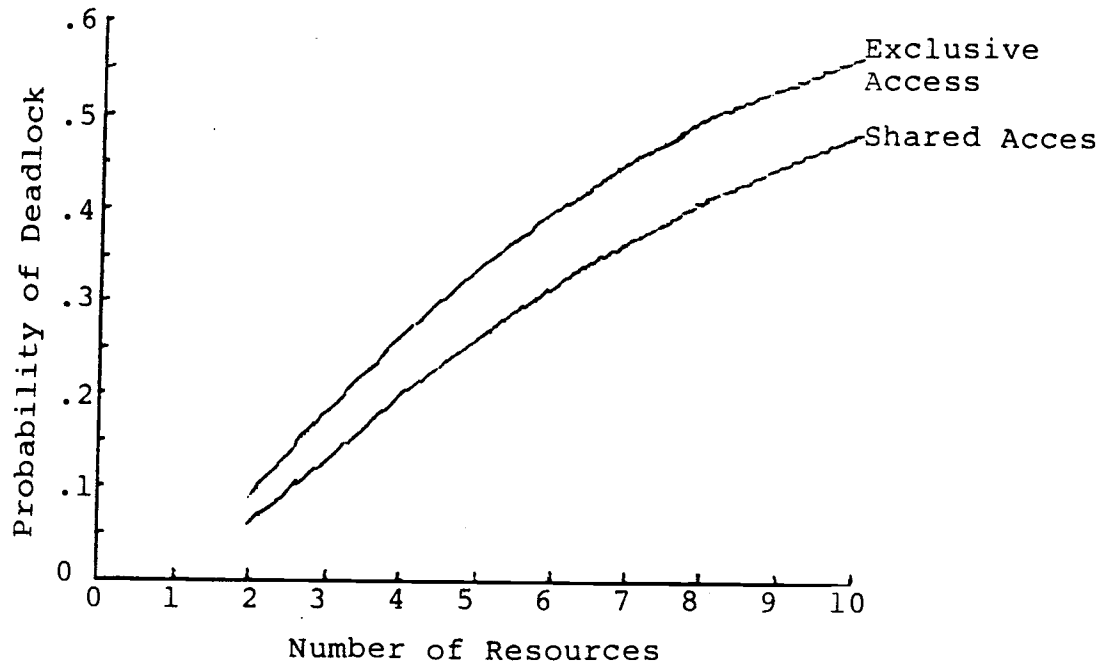

Figure 19. Comparing Probability of Deadlock in Shared and Exclusive Access for 2 Processes

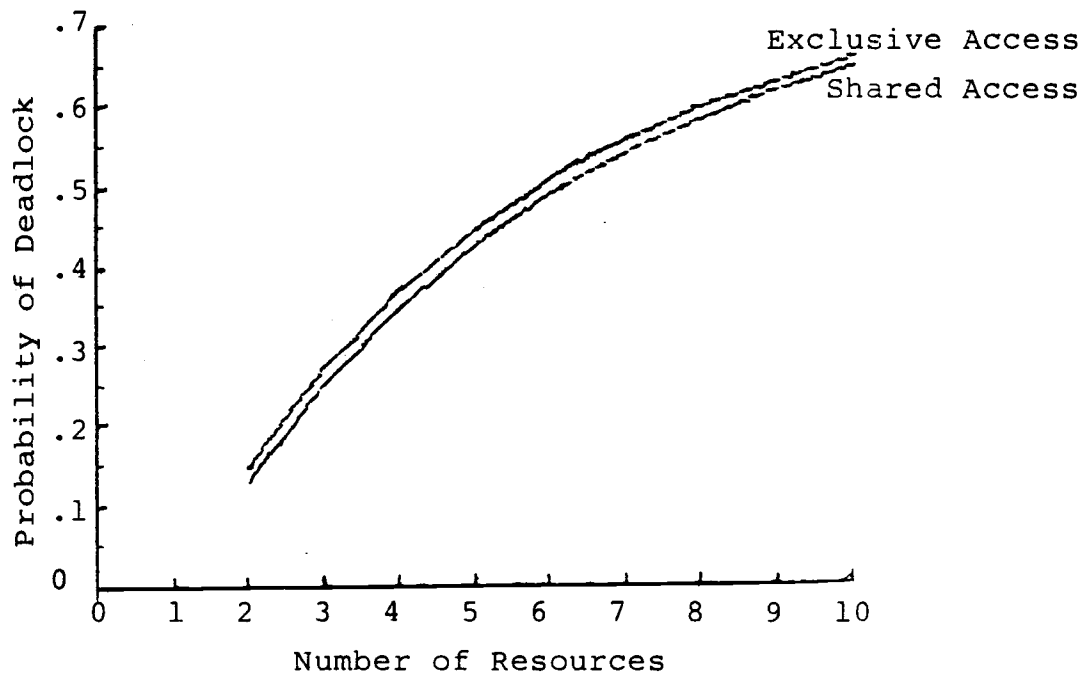

Figure 20. Comparing Probability of Deadlock in Shared and Exclusive Access for 3 Processes

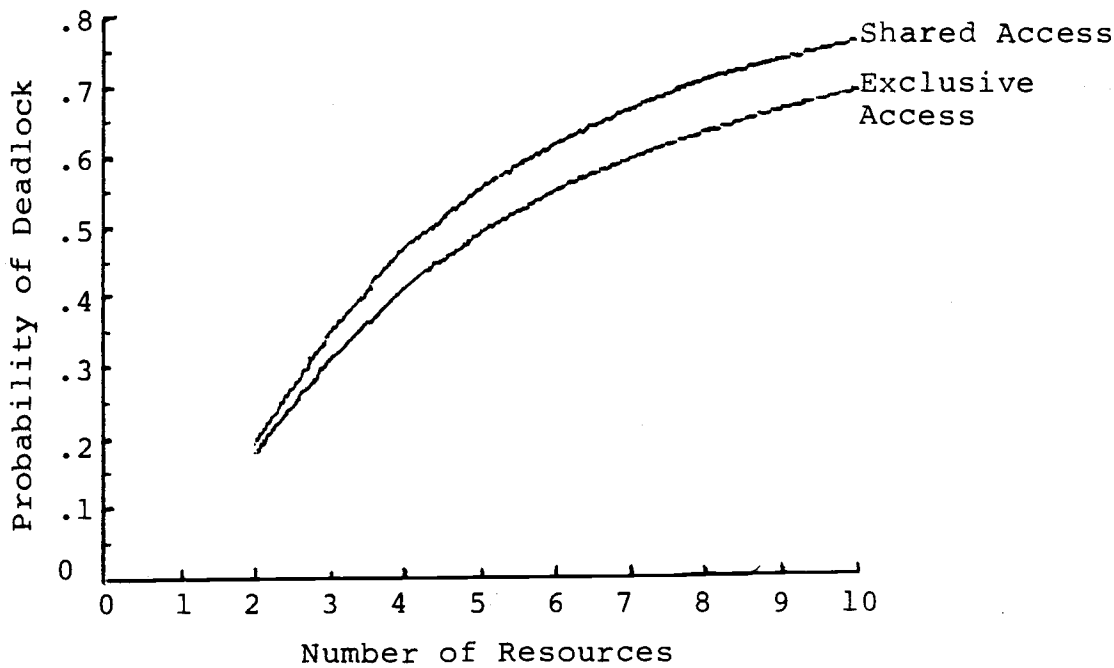

Figure 21. Comparing Probability of Deadlock in Shared and Exclusive Access for 4 Processes

VI. CONCLUSION

Deadlock studies attempt to partially answer one of the most fundamental questions in computer science: can a process progress to completion?

To see the possibility of deadlock in a system of processes and resources, this paper contains a method for computing the probability of deadlock in a process-resource graph. Two cases are considered, shared access and exclusive access. For each case, two formulas are given, one to compute the total number of graphs, another to compute the total number of graphs with cycle. By using these formulas, the probability of deadlock can be computed.

Experiment shows the probability of deadlock increases as the number of processes and resources increase. It is also shown that the probability of deadlock is higher in shared access when compared to exclusive access.

BIBLIOGRAPHY

[1] Birkes, David, Dodge, Yadolah and Seely, Justus, Spanning Sets for Estimable Contrasts in Classification Models, Dept. Stat., Oregon State University, 1976.

[2] Comtet, Louis, Advanced combinatorics, D. Reidel, Boston, 1974, pp 94-115.

[3] Edemenang, Effiong J. and Lewis, T. G., Simulation of On-Line Deadlock Detection Algorithms in a Distributed Computer System, Dept. CS, Oregon State University, Feb. 1982, pp 9-16.

[4] Lewis, T. G., Software Engineering: Analysis and Verification, Reston Books, Reston, VA., 1982, pp 391-430.

[5] Nijenhuis, Albert and Wilf, Herbert S., Combinatorial Algorithms, Academic Press, New York, 1975, pp 63-69.

[6] Shaw, Alan C., The Logical Design of Operating Systems, Prentice-Hall, Englewood Cliffs, N.J., 1974, pp 204-240.

APPENDIX

APPENDIX A

Program Listing

```
1000 REM ************************************
1020 REM * THIS PROGRAM CALCULATES THE      *
1040 REM * TOTAL NUMBER OF GRAPHS IN CASE*
1060 REM * OF EXCLUSIVE ACCESS FOR N        *
1080 REM * PROCESSES AND M RESOURCES        *
1100 REM ************************************
1120 DIM Q(10),K(10),Q1(10),K1(10)
1140 N=11
1160 M=3
1180 M2=0
1200 REM ************************************
1220 REM * THIS PART GIVES PARTITION OF     *
1240 REM * AN INTEGER , WHICH SHOWS HOW     *
1260 REM * THE PROCESSES MAKE REQUST FOR    *
1280 REM * THE RESOURCES. C IS THE TOTAL    *
1300 REM * NUMBER OF 1'S (REQUESTS) IN       *
1320 REM * THE TABLE. S IS THE NUMBER OF    *
1340 REM * DIFFERENT GROUPS OF 1'S IN THE*
1360 REM * TABLE. EACH ELEMENT OF ARRAY Q*
1380 REM * GIVES THE NUMBER OF ROWS IN A    *
1400 REM * GROUP. EACH ELEMENT OF ARRAY K*
1420 REM * SHOWS THE NUMBER OF 1'S IN       *
1440 REM * EACH ROW OF A GROUP.             *
1460 REM ************************************
1480 FOR C=1 TO N
1500 L1=0
1520 IF C=L1 THEN GOTO 1700
1540 L1=C
1560 P=C
1580 S=0
1600 S=S+1
1620 K(S)=P
1640 Q(S)=1
1660 T0=Q(S)*C
1680 GOTO 1960
1700 IF T0=0 THEN GOTO 1560
1720 S1=1
1740 IF K(S)>1 THEN GOTO 1800
1760 S1=Q(S)+1
1780 S=S-1
1800 F=K(S)-1
1820 IF Q(S)=1 THEN GOTO 1880
1840 Q(S)=Q(S)-1
1860 S=S+1
1880 K(S)=F
1900 Q(S)=1+S1 DIV F
1920 P=S1 MOD F
1940 IF P>0 THEN GOTO 1600 ELSE GOTO 1660
1960 Q9=0
1980 FOR J=1 TO S
2000 Q9=Q9+Q(J)
2020 NEXT J
2040 IF Q9>M THEN GOTO 3600
```

```
2060 REM ********************************
2080 REM * COMPUTE THE NUMBER OF WAYS   *
2100 REM * TO PUT 1'S IN THE ROWS OF    *
2120 REM * THE TABEL.                   *
2140 REM ********************************
2160 M3=1
2180 FOR I=1 TO M
2200 M3=M3*I
2220 NEXT I
2240 FOR I=1 TO S
2260 Q1(I)=1
2280 NEXT I
2300 Q2=1
2320 FOR I=1 TO S
2340 FOR J=1 TO Q(I)
2360 Q1(I)=Q1(I)*J
2380 NEXT J
2400 Q2=Q2*Q1(I)
2420 NEXT I
2440 M4=M
2460 FOR I=1 TO S
2480 M4=M4-Q(I)
2500 NEXT I
2520 M5=1
2540 FOR I=1 TO M4
2560 M5=M5*I
2580 NEXT I
2600 M6=M3 DIV (Q2*M5)
2620 REM ********************************
2640 REM * COMPUTE THE NUMBER OF WAYS   *
2660 REM * TO PUT 1'S IN THE COLUMNS    *
2680 REM * OF THE TABEL.                *
2700 REM ********************************
2720 N2=N
2740 FOR I=1 TO S
2760 N2=N2-K(I)*Q(I)
2780 NEXT I
2800 N3=1
2820 FOR I=1 TO N2
2840 N3=N3*I
2860 NEXT I
2880 FOR I=1 TO S
2900 K1(I)=1
2920 NEXT I
2940 K2=1
2960 FOR I=1 TO S
2980 FOR J=1 TO K(I)
3000 K1(I)=K1(I)*J
3020 NEXT J
3040 K1(I)=K1(I)^Q(I)
3060 K2=K1(I)*K2
3080 NEXT I
3100 N4=1
```

```
3120 FOR I=1 TO N
3140 N4=N4*I
3160 NEXT I
3180 N5=N4 DIV (K2*N3)
3200 REM ********************************
3220 REM * COMPUTE THE NUMBER OF WAYS   *
3240 REM * TO PUT 0'S IN THE REMAINING *
3260 REM * COLUMNS.                     *
3280 REM ********************************
3300 N6=1
3320 FOR I=1 TO S
3340 N6=(N-K(I))^Q(I)*N6
3360 NEXT I
3380 REM ********************************
3400 REM * COMPUTE THE NUMBER OF WAYS   *
3420 REM * TO PUT 0'S IN THE ROWS WITH *
3440 REM * NO 1'S IN THEM.              *
3460 REM ********************************
3480 M7=M
3500 FOR I=1 TO S
3520 M7=M7-Q(I)
3540 NEXT I
3560 N7=(N+1)^M7
3580 M2=M2+M6*N5*N6*N7
3600 IF T0=1 THEN GOTO 1520
3620 NEXT C
3640 PRINT USING 3660
3660 IMAGE "N",3X,"M",10X,"# OF GRAPHS"
3680 REM ********************************
3700 REM * M2 GIVES THE TOTAL NUMBER   *
3720 REM * OF TABELS (GRAPHS.)          *
3740 REM ********************************
3760 M2=M2+(N+1)^M
3780 PRINT USING 3800 ; N,M,M2
3800 IMAGE 7D,10X,7D,10X,18D
3820 END
```

```
1000 REM ************************************
1020 REM * THIS PROGRAM CALCULATES THE     *
1040 REM * NUMBER OF GRAPHS WITH NO CYCLE *
1060 REM * IN THE CASE OF EXCLUSIVE        *
1080 REM * ACCESS (READ/WRITE).            *
1100 REM ************************************
1120 DIM Q(10),K(10),Q1(10),K1(10),A(10,10)
1140 DIM A1(10),B(10,10),B1(10,10),P1(10)
1160 PRINT USING 1180
1180 IMAGE "N",3X,"M",10X,"# OF GRAPHS WITH NO CYCLE"
1200 N=2
1220 M=3
1240 G0=0
1260 IF N=2 THEN GOTO 7320
1280 C1=N-1
1300 REM ************************************
1320 REM * THIS PART GIVES PARTITION OF *
1340 REM * AN INTEGER, WHICH SHOWS THE  *
1360 REM * POSSIBLE WAYS FOR PROCESSES  *
1380 REM * TO MAKE REQUEST FOR          *
1400 REM * DIFFERENT TYPES OF RESOURCES *
1420 REM ************************************
1440 FOR C=2 TO C1
1460 L1=0
1480 IF C=L1 THEN GOTO 1760
1500 L1=C
1520 P=C
1540 S=0
1560 S=S+1
1580 K(S)=P
1600 Q(S)=1
1620 T0=Q(S)*C
1640 Q9=0
1660 FOR J=1 TO S
1680 Q9=Q(J)+Q9
1700 NEXT J
1720 IF Q9>M THEN GOTO 6760
1740 GOTO 2020
1760 IF T0=0 THEN GOTO 1520
1780 S1=1
1800 IF K(S)>1 THEN GOTO 1860
1820 S1=Q(S)+1
1840 S=S-1
1860 F=K(S)-1
1880 IF Q(S)=1 THEN GOTO 1940
1900 Q(S)=Q(S)-1
1920 S=S+1
1940 K(S)=F
1960 Q(S)=1+S1 DIV F
1980 P=S1 MOD F
2000 IF P>0 THEN GOTO 1560 ELSE GOTO 1620
2020 Q3=0
2040 FOR I=1 TO S
```

```
2060 Q3=Q(I)+Q3
2080 NEXT I
2100 IF Q3=1 THEN GOTO 6920
2120 Q4=Q3+1
2140 N6=0
2160 REM ***********************************
2180 REM * TRANSFERING THE TABLE TO A      *
2200 REM * NEW ONE WITH ONLY ONE "1" IN    *
2220 REM * EACH OF ITS ROWS.               *
2240 REM ***********************************
2260 FOR I=1 TO Q3
2280 FOR J=1 TO Q4
2300 IF I=J THEN A(I,J)=1 ELSE A(I,J)=2
2320 NEXT J
2340 NEXT I
2360 REM ***********************************
2380 REM * THIS PART SHOWS THE WAY "0"     *
2400 REM * COULD BE PUT IN EACH ROW OF     *
2420 REM * THE TABEL BY USING THE          *
2440 REM * METHOD OF BACKTRACKING.         *
2460 REM ***********************************
2480 I=Q3
2500 FOR J=2 TO Q3
2520 A1(J)=1
2540 NEXT J
2560 A1(1)=2
2580 FOR J=1 TO Q3
2600 A(J,A1(J))=0
2620 NEXT J
2640 GOTO 3120
2660 FOR I1=1 TO Q3
2680 FOR J=1 TO Q4
2700 IF I1=J THEN A(I1,J)=1 ELSE A(I1,J)=2
2720 NEXT J
2740 NEXT I1
2760 FOR J=1 TO Q3
2780 A(J,A1(J))=0
2800 NEXT J
2820 GOTO 3120
2840 A1(I)=A1(I)+1
2860 IF A1(I)=I THEN GOTO 2840
2880 IF A1(I)(=Q4 THEN GOTO 2660
2900 I=I-1
2920 IF I=0 THEN GOTO 5460
2940 A1(I)=A1(I)+1
2960 IF A1(I)=I THEN GOTO 2940
2980 IF A1(I)>Q4 THEN GOTO 2900
3000 I=I+1
3020 FOR J=I TO Q3
3040 A1(J)=1
3060 NEXT J
3080 I=Q3
3100 GOTO 2660
```

```
3120 FOR J=1 TO Q3
3140 IF A(J,Q4)=0 THEN GOTO 3420
3160 NEXT J
3180 GOTO 2840
3200 REM ********************************
3220 REM * THIS PART CHECK FOR CYCLE     *
3240 REM * BY CROSSING OUT THOSE ROWS    *
3260 REM * AND COLUMNS WHICH HAVE ONLY   *
3280 REM * ONE TYPE OF ENTRY, EITHER     *
3300 REM * "0" OR "1". AT THE END IF     *
3320 REM * THERE ARE MORE THAN ONE ROW   *
3340 REM * AND COLUMN IT INDICATES       *
3360 REM * THERE IS A CYCLE IN THE       *
3380 REM * TABEL.                        *
3400 REM ********************************
3420 FOR I1=1 TO Q3
3440 FOR J=1 TO Q4
3460 B1(I1,J)=A(I1,J)
3480 NEXT J
3500 NEXT I1
3520 U=Q4
3540 U1=Q4
3560 U2=Q4
3580 V=Q3
3600 V1=Q3
3620 V2=Q3
3640 I2=0
3660 FOR I1=1 TO V
3680 N0=0
3700 FOR J=1 TO U
3720 IF B1(I1,J)=0 OR B1(I1,J)=1 THEN N0=N0+1
3740 IF N0<=1 THEN GOTO 3880
3760 I2=I2+1
3780 FOR K2=1 TO U
3800 B(I2,K2)=B1(I1,K2)
3820 NEXT K2
3840 NEXT I1
3860 GOTO 3940
3880 NEXT J
3900 IF N0<=1 THEN V2=V2-1
3920 NEXT I1
3940 IF V2>=2 THEN GOTO 3980
3960 GOTO 4680
3980 V=V2
4000 U=U2
4020 FOR I1=1 TO V
4040 FOR J=1 TO U
4060 B1(I1,J)=B(I1,J)
4080 NEXT J
4100 NEXT I1
4120 J1=0
4140 FOR J=1 TO U
4160 N0=0
```

```
4180 FOR I1=1 TO V
4200 IF B1(I1,J)=0 OR B1(I1,J)=1 THEN N0=N0+1
4220 IF N0<=1 THEN GOTO 4360
4240 J1=J1+1
4260 FOR K2=1 TO V
4280 B(K2,J1)=B1(K2,J)
4300 NEXT K2
4320 NEXT J
4340 GOTO 4420
4360 NEXT I1
4380 IF N0<=1 THEN U2=U2-1
4400 NEXT J
4420 IF U2>=2 THEN GOTO 4460
4440 GOTO 4680
4460 U=U2
4480 V=V2
4500 IF U=U1 AND V=V1 THEN GOTO 2840
4520 U1=U
4540 V1=V
4560 FOR I1=1 TO V
4580 FOR J=1 TO U
4600 B1(I1,J)=B(I1,J)
4620 NEXT J
4640 NEXT I1
4660 GOTO 3640
4680 S2=S+1
4700 REM ************************************
4720 REM * CALCULATE THE NUMBER OF 0'S   *
4740 REM * IN THE TABEL. EACH ELEMENT    *
4760 REM * OF ARRAY "P1" GIVES THE       *
4780 REM * NUMBER OF 0'S IN EACH GROUP.  *
4800 REM ************************************
4820 FOR I1=1 TO S2
4840 P1(I1)=0
4860 NEXT I1
4880 J1=1
4900 J2=0
4920 FOR I1=1 TO S
4940 J2=J2+Q(I1)
4960 FOR J=J1 TO J2
4980 FOR K2=1 TO Q3
5000 IF A(K2,J)=0 THEN P1(I1)=P1(I1)+1
5020 NEXT K2
5040 NEXT J
5060 J1=J1+Q(I1)
5080 NEXT I1
5100 FOR K2=1 TO Q3
5120 IF A(K2,Q4)=0 THEN P1(S2)=P1(S2)+1
5140 NEXT K2
5160 N8=1
5180 N9=0
5200 FOR I1=1 TO S
5220 N8=K(I1)^P1(I1)*N8
```

```
5240 N9=N9+Q(I1)*K(I1)
5260 NEXT I1
5280 N8=N8*(N-N9)^P1(S2)
5300 N6=N6+N8
5320 GOTO 2840
5340 REM ***************************************
5360 REM * COMPUTE                             *
5380 REM * M!/((Q1!*...*QS!)*(M-Q1-...-QS)!)  *
5400 REM * WHICH IS THE NUMBER OF WAYS TO      *
5420 REM * CHOOSE ROWS WITH 1'S IN THEM.       *
5440 REM ***************************************
5460 M3=1
5480 FOR I=2 TO M
5500 M3=M3*I
5520 NEXT I
5540 FOR I=1 TO S
5560 Q1(I)=1
5580 NEXT I
5600 Q2=1
5620 FOR I=1 TO S
5640 FOR J=1 TO Q(I)
5660 Q1(I)=Q1(I)*J
5680 NEXT J
5700 Q2=Q2*Q1(I)
5720 NEXT I
5740 M4=M
5760 FOR I=1 TO S
5780 M4=M4-Q(I)
5800 NEXT I
5820 M5=1
5840 FOR I=1 TO M4
5860 M5=M5*I
5880 NEXT I
5900 M6=M3 DIV (Q2*M5)
5920 REM ***************************************************
5940 REM * COMPUTE                                         *
5960 REM * N!/((K1!^Q1*...*KS!^QS)*(N-K1Q1-...-KSQS)!)*
5980 REM * WHICH IS THE NUMBER OF WAYS TO PUT 1'S          *
6000 REM * IN THE K(I) ENTRIES OF EACH ROW.               *
6020 REM ***************************************************
6040 N2=N
6060 FOR I=1 TO S
6080 N2=N2-K(I)*Q(I)
6100 NEXT I
6120 N3=1
6140 FOR I=1 TO N2
6160 N3=N3*I
6180 NEXT I
6200 FOR I=1 TO S
6220 K1(I)=1
6240 NEXT I
6260 K3=1
6280 FOR I=1 TO S
```

```
6300 FOR J=1 TO K(I)
6320 K1(I)=K1(I)*J
6340 NEXT J
6360 K1(I)=K1(I)^Q(I)
6380 K3=K1(I)*K3
6400 NEXT I
6420 N4=1
6440 FOR I=1 TO N
6460 N4=N4*I
6480 NEXT I
6500 N5=N4 DIV (K3*N3)
6520 REM ***********************************
6540 REM * COMPUTE THE NUMBER OF WAYS TO *
6560 REM * PUT 0'S IN THE REMAINING      *
6580 REM * ROWS OF THE TABLE:            *
6600 REM * (N-1)^(M-Q1-Q2-...-QS)        *
6620 REM ***********************************
6640 M7=M
6660 FOR I=1 TO S
6680 M7=M7-Q(I)
6700 NEXT I
6720 N7=(N+1)^M7
6740 G0=G0+M6*N5*N6*N7
6760 IF T0=1 THEN GOTO 1480
6780 NEXT C
6800 GOTO 7320
6820 REM ***********************************
6840 REM * COMPUTE THE NUMBER OF TABELS  *
6860 REM * IN CASE THERE IS ONLY ONE ROW *
6880 REM * IN THE TABEL WITH 1'S IN IT.  *
6900 REM ***********************************
6920 N4=1
6940 FOR I=1 TO N
6960 N4=N4*I
6980 NEXT I
7000 K4=1
7020 FOR I=1 TO K(S)
7040 K4=K4*I
7060 NEXT I
7080 N5=N-K(S)
7100 N6=1
7120 FOR I=1 TO N5
7140 N6=N6*I
7160 NEXT I
7180 N7=N4 DIV (K4*N6)
7200 G0=G0+M*N7*(N-K(S))*(N+1)^(M-1)
7220 GOTO 6760
7240 REM ***********************************
7260 REM * G0 GIVES THE TOTAL NUMBER OF  *
7280 REM * GRAPHS WITH NO CYCLE.         *
7300 REM ***********************************
7320 G0=G0+(N+1)^M+M*N*(N-1)*(N+1)^(M-1)
7340 PRINT USING 7360 ; N,M,G0
```

```
7360 IMAGE 7D,10X,7D,10X,18D
7380 END
```

```
1000 REM ***********************************
1020 REM * THIS PROGRAM CALCULATES THE     *
1040 REM * TOTAL NUMBER OF GRAPHS IN CASE*
1060 REM * OF SHARED ACCESS (READ ONLY)    *
1080 REM * FOR N PROCESSES AND M           *
1100 REM * RESOURCES.                      *
1120 REM ***********************************
1140 DIM Q(10),K(10),Q1(10),K1(10),M2(10,10)
1160 T=10
1180 T1=3
1200 FOR N=2 TO T
1220 FOR M=T1 TO T1
1240 M2(N,M)=0
1260 NEXT M
1280 NEXT N
1300 FOR N=2 TO T
1320 REM ***********************************
1340 REM * THIS PART GIVES PARTITION OF    *
1360 REM * AN INTEGER , WHICH SHOWS HOW    *
1380 REM * THE PROCESSES MAKE REQUEST      *
1400 REM * FOR THE RESOURCES. C IS THE     *
1420 REM * NUMBER OF 1'S IN THE TABEL.     *
1440 REM * S IS THE NUMBER OF DIFFERANT    *
1460 REM * GROUPS OF 1'S IN THE TABEL.     *
1480 REM * EACH ELEMENT OF ARRAY Q GIVES*
1500 REM * THE NUMBER OF ROWS IN A GROUP*
1520 REM * EACH ELEMENT OF ARRAY K SHOWS*
1540 REM * THE NUMBER OF 1'S IN EACH ROW*
1560 REM * OF A GROUP.                     *
1580 REM ***********************************
1600 FOR C=1 TO N
1620 L1=0
1640 IF C=L1 THEN GOTO 1820
1660 L1=C
1680 P=C
1700 S=0
1720 S=S+1
1740 K(S)=P
1760 Q(S)=1
1780 T0=Q(S)*C
1800 GOTO 2080
1820 IF T0=0 THEN GOTO 1680
1840 S1=1
1860 IF K(S)>1 THEN GOTO 1920
1880 S1=Q(S)+1
1900 S=S-1
1920 F=K(S)-1
1940 IF Q(S)=1 THEN GOTO 2000
1960 Q(S)=Q(S)-1
1980 S=S+1
2000 K(S)=F
2020 Q(S)=1+S1 DIV F
2040 P=S1 MOD F
```

```
2060 IF P>0 THEN GOTO 1720 ELSE GOTO 1780
2080 FOR M=T1 TO T1
2100 Q9=0
2120 FOR J=1 TO S
2140 Q9=Q9+Q(J)
2160 NEXT J
2180 IF Q9>M THEN GOTO 3760
2200 REM ***********************************
2220 REM * COMPUTE THE NUMBER OF WAYS TO*
2240 REM * PUT 1'S IN THE ROWS OF THE    *
2260 REM * TABEL.                        *
2280 REM ***********************************
2300 M3=1
2320 FOR I=1 TO M
2340 M3=M3*I
2360 NEXT I
2380 FOR I=1 TO S
2400 Q1(I)=1
2420 NEXT I
2440 Q2=1
2460 FOR I=1 TO S
2480 FOR J=1 TO Q(I)
2500 Q1(I)=Q1(I)*J
2520 NEXT J
2540 Q2=Q2*Q1(I)
2560 NEXT I
2580 M4=M
2600 FOR I=1 TO S
2620 M4=M4-Q(I)
2640 NEXT I
2660 M5=1
2680 FOR I=1 TO M4
2700 M5=M5*I
2720 NEXT I
2740 M6=M3 DIV (Q2*M5)
2760 REM ***********************************
2780 REM * COMPUTE THE NUMBER OF WAYS TO*
2800 REM * PUT 1'S IN THE COLUMNS OF THE*
2820 REM * TABEL.                        *
2840 REM ***********************************
2860 N2=N
2880 FOR I=1 TO S
2900 N2=N2-K(I)*Q(I)
2920 NEXT I
2940 N3=1
2960 FOR I=1 TO N2
2980 N3=N3*I
3000 NEXT I
3020 FOR I=1 TO S
3040 K1(I)=1
3060 NEXT I
3080 K2=1
3100 FOR I=1 TO S
```

```
3120 FOR J=1 TO K(I)
3140 K1(I)=K1(I)*J
3160 NEXT J
3180 K1(I)=K1(I)^Q(I)
3200 K2=K1(I)*K2
3220 NEXT I
3240 N4=1
3260 FOR I=1 TO N
3280 N4=N4*I
3300 NEXT I
3320 N5=N4 DIV (K2*N3)
3340 REM ********************************
3360 REM * COMPUTE THE NUMBER OF WAYS   *
3380 REM * TO PUT 0'S IN THE REMAINING *
3400 REM * COLUMNS OF THE TABEL.        *
3420 REM ********************************
3440 N6=1
3460 FOR I=1 TO S
3480 N6=(2^(N-K(I))-1)^Q(I)*N6
3500 NEXT I
3520 REM ********************************
3540 REM * COMPUTE THE NUMBER OF WAYS   *
3560 REM * TO PUT 0'S IN THE ROWS WITH *
3580 REM * NO 1'S IN THEM.              *
3600 REM ********************************
3620 M7=M
3640 FOR I=1 TO S
3660 M7=M7-Q(I)
3680 NEXT I
3700 N7=2^(M7*N)
3720 M2(N,M)=M2(N,M)+M6*N5*N6*N7
3740 NEXT M
3760 IF T0=1 THEN GOTO 1640
3780 NEXT C
3800 NEXT N
3820 PRINT USING 3840
3840 IMAGE "N",3X,"M",10X,"# OF GRAPHS"
3860 REM ********************************
3880 REM * M2 GIVES THE TOTAL NUMBER    *
3900 REM * OF TABELS (GRAPHS.)          *
3920 REM ********************************
3940 FOR N=2 TO T
3960 FOR M=T1 TO T1
3980 M2(N,M)=M2(N,M)+2^(N*M)
4000 PRINT USING 4020 ; N,M,M2(N,M)
4020 IMAGE 7D,10X,7D,10X,18D
4040 NEXT M
4060 NEXT N
4080 END
```

```
1000 REM ************************************
1020 REM * THIS PROGRAM CALCULATES THE      *
1040 REM * NUMBER OF GRAPHS WITH NO CYCLE*
1060 REM * IN THE CASE OF SHARED ACCESS     *
1080 REM * (READ ONLY).                     *
1100 REM ************************************
1120 DIM Q(10),Q1(10),K(10),K1(10),A(10,10)
1140 DIM A1(10),B(10,10),B1(10,10),P1(10),W(10)
1160 PRINT USING 1180
1180 IMAGE "N",3X,"M",10X,"# OF GRAPHS WITH NO CYCLE"
1200 N=2
1220 M=3
1240 G0=0
1260 IF N=2 THEN GOTO 7260
1280 C1=N-1
1300 REM ************************************
1320 REM * THIS PART GIVES PARTITION OF     *
1340 REM * AN INTEGER.                      *
1360 REM ************************************
1380 FOR C=2 TO C1
1400 L1=0
1420 IF C=L1 THEN GOTO 1700
1440 L1=C
1460 P=C
1480 S=0
1500 S=S+1
1520 K(S)=P
1540 Q(S)=1
1560 T0=Q(S)*C
1580 Q9=0
1600 FOR J=1 TO S
1620 Q9=Q(J)+Q9
1640 NEXT J
1660 IF Q9>M THEN GOTO 6700
1680 GOTO 1960
1700 IF T0=0 THEN GOTO 1460
1720 S1=1
1740 IF K(S)>1 THEN GOTO 1800
1760 S1=Q(S)+1
1780 S=S-1
1800 F=K(S)-1
1820 IF Q(S)=1 THEN GOTO 1880
1840 Q(S)=Q(S)-1
1860 S=S+1
1880 K(S)=F
1900 Q(S)=1+S1 DIV F
1920 P=S1 MOD F
1940 IF P>0 THEN GOTO 1500 ELSE GOTO 1560
1960 Q3=0
1980 FOR I=1 TO S
2000 Q3=Q(I)+Q3
2020 NEXT I
2040 IF Q3=1 THEN GOTO 6860
```

```
2060 Q4=Q3+1
2080 N6=0
2100 REM ************************************
2120 REM * THIS PART SHOWS THE WAY 0'S     *
2140 REM * COULD BE PUT IN THE TABEL.      *
2160 REM ************************************
2180 I=1
2200 FOR I1=1 TO Q3
2220 A1(I1)=0
2240 NEXT I1
2260 A1(I)=A1(I)+1
2280 IF A1(I)<2^Q4-1 THEN GOTO 2400
2300 GOTO 2800
2320 REM ************************************
2340 REM * CONVERT DECIMAL NUMBERS INTO     *
2360 REM * BINARY NUMBERS.                  *
2380 REM ************************************
2400 FOR J=1 TO Q4
2420 W(J)=0
2440 NEXT J
2460 A2=A1(I)
2480 FOR J=Q4 TO 1 STEP -1
2500 W(J)=A2 MOD 2
2520 A2=A2 DIV 2
2540 IF A2>1 THEN GOTO 2600
2560 W(J-1)=A2
2580 GOTO 2620
2600 NEXT J
2620 IF W(I)=1 THEN GOTO 2640 ELSE GOTO 2260
2640 FOR J=1 TO Q4
2660 IF J=I THEN GOTO 2700
2680 IF W(J)=1 THEN A(I,J)=2 ELSE A(I,J)=0
2700 NEXT J
2720 A(I,I)=1
2740 IF I>=Q3 THEN GOTO 2940
2760 I=I+1
2780 GOTO 2260
2800 I=I-1
2820 IF I=0 THEN GOTO 5420
2840 IF A1(I)<2^Q4-1 THEN GOTO 2860 ELSE GOTO 2800
2860 FOR J=I+1 TO Q3
2880 A1(J)=0
2900 NEXT J
2920 GOTO 2260
2940 FOR J=1 TO Q3
2960 IF A(J,Q4)=0 THEN GOTO 3200
2980 NEXT J
3000 GOTO 2260
3020 REM ************************************
3040 REM * THIS PART CHECKS FOR CYCLE BY    *
3060 REM * CROSSING OUT ROWS AND COLUMNS    *
3080 REM * WITH ONLY ONE TYPE OF ENTRY,     *
3100 REM * EITHER "0" OR "1". AT THE END IF *
```

```
3120 REM * THERE ARE MORE THAN ONE ROW AND  *
3140 REM * COLUMN , IT INDICATES THERE IS A *
3160 REM * CYCLE IN THE TABEL.               *
3180 REM ************************************
3200 FOR I1=1 TO Q3
3220 FOR J=1 TO Q4
3240 B1(I1,J)=A(I1,J)
3260 NEXT J
3280 NEXT I1
3300 U=Q4
3320 U1=Q4
3340 U2=Q4
3360 V=Q3
3380 V1=Q3
3400 V2=Q3
3420 I2=0
3440 FOR I1=1 TO V
3460 N0=0
3480 FOR J1=1 TO U
3500 IF B1(I1,J1)<>0 THEN GOTO 3560
3520 N0=N0+1
3540 GOTO 3580
3560 NEXT J1
3580 FOR J=1 TO U
3600 IF B1(I1,J)=1 THEN N0=N0+1
3620 IF N0<=1 THEN GOTO 3760
3640 I2=I2+1
3660 FOR K2=1 TO U
3680 B(I2,K2)=B1(I1,K2)
3700 NEXT K2
3720 NEXT I1
3740 GOTO 3820
3760 NEXT J
3780 IF N0<=1 THEN V2=V2-1
3800 NEXT I1
3820 IF V2>=2 THEN GOTO 3860
3840 GOTO 4780
3860 V=V2
3880 U=U2
3900 FOR I1=1 TO V
3920 FOR J=1 TO U
3940 B1(I1,J)=B(I1,J)
3960 NEXT J
3980 NEXT I1
4000 J1=0
4020 FOR J=1 TO U
4040 N0=0
4060 FOR I2=1 TO V
4080 IF B1(I2,J)<>0 THEN GOTO 4140
4100 N0=N0+1
4120 GOTO 4160
4140 NEXT I2
4160 FOR I1=1 TO V
```

```
4180 IF B1(I1,J)=1 THEN N0=N0+1
4200 IF N0<=1 THEN GOTO 4340
4220 J1=J1+1
4240 FOR K2=1 TO V
4260 B(K2,J1)=B1(K2,J)
4280 NEXT K2
4300 NEXT J
4320 GOTO 4400
4340 NEXT I1
4360 IF N0<=1 THEN U2=U2-1
4380 NEXT J
4400 IF U2>=2 THEN GOTO 4440
4420 GOTO 4780
4440 U=U2
4460 V=V2
4480 IF U=U1 AND V=V1 THEN GOTO 2260
4500 U1=U
4520 V1=V
4540 FOR I1=1 TO V
4560 FOR J=1 TO U
4580 B1(I1,J)=B(I1,J)
4600 NEXT J
4620 NEXT I1
4640 GOTO 3420
4660 REM ************************************
4680 REM * CALCULATE THE NUMBER OF 0'S     *
4700 REM * IN THE TABEL. EACH ELEMENT OF   *
4720 REM * ARRAY "P1" GIVES THE NUMBER     *
4740 REM * OF 0'S IN EACH GROUP.           *
4760 REM ************************************
4780 S2=S+1
4800 FOR I1=1 TO S2
4820 P1(I1)=0
4840 NEXT I1
4860 J1=1
4880 J2=0
4900 FOR I1=1 TO S
4920 J2=J2+Q(I1)
4940 FOR J=J1 TO J2
4960 FOR K2=1 TO Q3
4980 IF A(K2,J)=0 THEN P1(I1)=P1(I1)+1
5000 NEXT K2
5020 NEXT J
5040 J1=J1+Q(I1)
5060 NEXT I1
5080 FOR K2=1 TO Q3
5100 IF A(K2,Q4)=0 THEN P1(S2)=P1(S2)+1
5120 NEXT K2
5140 N8=1
5160 N9=0
5180 FOR I1=1 TO S
5200 N8=(2^K(I1)-1)^P1(I1)*N8
5220 N9=N9+Q(I1)*K(I1)
```

```
5240 NEXT I1
5260 N8=N8*(2^(N-N9)-1)^P1(S2)
5280 N6=N6+N8
5300 GOTO 2260
5320 REM ***************************************
5340 REM * COMPUTE THE NUMBER OF WAYS TO CHOOSE  *
5360 REM * ROWS OF THE TABEL WITH 1'S IN THEM:   *
5380 REM * M!/((Q1!*...*QS!)*(M-Q1-...-QS)!)     *
5400 REM ***************************************
5420 M3=1
5440 FOR I=2 TO M
5460 M3=M3*I
5480 NEXT I
5500 FOR I=1 TO S
5520 Q1(I)=1
5540 NEXT I
5560 Q2=1
5580 FOR I=1 TO S
5600 FOR J=1 TO Q(I)
5620 Q1(I)=Q1(I)*J
5640 NEXT J
5660 Q2=Q2*Q1(I)
5680 NEXT I
5700 M4=M
5720 FOR I=1 TO S
5740 M4=M4-Q(I)
5760 NEXT I
5780 M5=1
5800 FOR I=1 TO M4
5820 M5=M5*I
5840 NEXT I
5860 M6=M3 DIV (Q2*M5)
5880 REM *****************************************************
5900 REM * COMPUTE THE NUMBER OF WAYS TO PUT 1'S IN     *
5920 REM * THE K(I) ENTRIES OF EACH ROW:                *
5940 REM * N!/((K1!^Q1*...*KS!^QS)*(N-K1*Q1-...KS*QS)!) *
5960 REM *****************************************************
5980 N2=N
6000 FOR I=1 TO S
6020 N2=N2-K(I)*Q(I)
6040 NEXT I
6060 N3=1
6080 FOR I=1 TO N2
6100 N3=N3*I
6120 NEXT I
6140 FOR I=1 TO S
6160 K1(I)=1
6180 NEXT I
6200 K3=1
6220 FOR I=1 TO S
6240 FOR J=1 TO K(I)
6260 K1(I)=K1(I)*J
6280 NEXT J
```

```
6300 K1(I)=K1(I)^Q(I)
6320 K3=K1(I)*K3
6340 NEXT I
6360 N4=1
6380 FOR I=1 TO N
6400 N4=N4*I
6420 NEXT I
6440 N5=N4 DIV (K3*N3)
6460 REM ****************************************
6480 REM * COMPUTE THE NUMBER OF WAYS TO PUT    *
6500 REM * 0'S IN THE REMAINING ROWS OF THE     *
6520 REM * TABEL WHICH IS:                       *
6540 REM * 2^((M-Q1-...-QS)*N)                  *
6560 REM ****************************************
6580 M7=M
6600 FOR I=1 TO S
6620 M7=M7-Q(I)
6640 NEXT I
6660 N7=2^(M7*N)
6680 G0=G0+M6*N5*N6*N7
6700 IF T0=1 THEN GOTO 1420
6720 NEXT C
6740 GOTO 7260
6760 REM ************************************
6780 REM * COMPUTE THE NUMBER OF TABELS IN   *
6800 REM * CASE THERE IS ONLY ONE ROW IN     *
6820 REM * THE TABEL WITH 1'S IN IT.         *
6840 REM ************************************
6860 N4=1
6880 FOR I=1 TO N
6900 N4=N4*I
6920 NEXT I
6940 K4=1
6960 FOR I=1 TO K(S)
6980 K4=K4*I
7000 NEXT I
7020 N5=N-K(S)
7040 N6=1
7060 FOR I=1 TO N5
7080 N6=N6*I
7100 NEXT I
7120 N7=N4 DIV (K4*N6)
7140 G0=G0+M*N7*(2^(N-K(S))-1)*2^((M-1)*N)
7160 GOTO 6700
7180 REM ************************************
7200 REM * G0 GIVES THE TOTAL NUMBER OF      *
7220 REM * GRAPHS WITH NO CYCLE.              *
7240 REM ************************************
7260 G0=G0+2^(M*N)+M*N*(2^(N-1)-1)*2^((M-1)*N)
7280 PRINT USING 7300 ; N,M,G0
7300 IMAGE 7D,10X,7D,10X,18D
7320 END
```

```
1000 REM ***********************************
1020 REM * THIS PROGRAM CALCULATES THE    *
1040 REM * PROBABILITY OF DEADLOCK AND    *
1060 REM * PLOTS IT VERSUS DIFFERENT      *
1080 REM * NUMBER OF PROCESSES AND         *
1100 REM * RESOURCES.                     *
1120 REM ***********************************
1140 DIM P(11)
1160 T=11
1180 FOR I=2 TO T
1200 READ A,B
1220 P(I)=(A-B)/A
1240 NEXT I
1260 DATA 99,81,976,712,8021,5489,58176
1270 DATA 38316,385855,248005,2393462,1513436
1280 DATA 14095017,8811585,79635484
1290 DATA 49376980,434986451,268064261
1300 DATA 2310320706,1417123236
1320 PEN 1
1340 GCLEAR
1360 SCALE -4,12,-.2,.6
1380 XAXIS 0,1,0,10
1400 YAXIS 0,.05,0,.5
1420 LDIR 0
1440 MOVE 1,-.15
1460 LABEL "NUMBER OF PROCESSE"
1510 LDIR 90
1520 MOVE -2,.05
1540 LABEL "PROB. OF DEADLOCK"
1560 REM ***********************************
1580 REM * LABEL X-AXIS.                   *
1600 REM ***********************************
1620 LDIR 0
1640 FOR X=0 TO 10
1660 MOVE X,-.05
1680 LABEL VAL$(X)
1700 NEXT X
1720 REM ***********************************
1740 REM * LABEL Y-AXES.                   *
1760 REM ***********************************
1780 FOR Y=0 TO .5 STEP .1
1800 MOVE -1.5,Y
1820 LABEL VAL$(Y)
1840 NEXT Y
1860 MOVE 2,P(2)
1880 FOR I=3 TO T
1900 DRAW I,P(I)
1920 NEXT I
1940 GRAPH
1960 COPY
1980 END
```