ON THE PROBABILITY OF DEADLOCK IN COMPUTER SYSTEMS *

Clarence A. Ellis

Department of Computer Science
University of Colorado
Boulder, Colorado

Report #CU-CS-026-73 August, 1973

* This work supported by National Science Foundation Grant # GJ-660

ABSTRACT

As the number of processes and resources increases within a computer system, does the probability of that system's being in deadlock increase or decrease? This paper introduces Probabilistic Automata as a model of computer systems. This allows investigation of the above question for various scheduling algorithms. A theorem is proved which indicates that, within the types of systems considered, the probability of deadlock increases.

INTRODUCTION

A computer system which allows more than one process to be simultaneously active, holding and requesting resources, may encounter the phenomenon of deadlock (sometimes called deadly embrace). A simple case of this occurs when two processes each wait for a resource held by the other. Usually, these processes will be idle for an indefinite amount of time until operator intervention corrects this situation. The automatic detection and prevention of deadlock has recently received a lot of attention in the literature[1,4,5,6,7,8,9] In practice, these algorithms are time-consuming, and have generally not been implemented on actual computer systems, the philosophy being:

> "Deadlock occurs so infrequently that it is not worthwhile to degrade system performance by executing deadlock prevention algorithms."

The new model presented in this paper provides a tool for observing and measuring deadlock in systems under various resource allocation algorithms. This model is then applied to systems containing many units of a single resource (e.g., memory units) under random, first-come-first-serve, and last-come-first-serve resource allocation algorithms yielding the result that as the number of processes (and resources) becomes larger, the probability of deadlock increases. Thus, in multi-multi-processors of the future, the deadlock problem will need to be explicitly dealt with.

THE MODEL

A computer system consisting of a set of processes and varying units of various types of resources can be described at any particular instant of time as being in some state defined by: (1) the number of units of each type of resource available, (2) the number of units being held by each process, and (3) the number of units being requested by each process. A transition or change to another state corresponds to a new request, an allocation, or some other system action causing any of the above three parameters to change. Holt[8, 9] shows that this type of system can be graphically represented by a System State Diagram. This directed graph can also be viewed as a finite state automaton as shown by Nutt provided that the number of states is finite[12]. Furthermore, for each possible transition out of a given state, one can attach a probability to the occurrence of this event. We further require that the sum of the probabilities associated with transitions out of a given state sum to one for each state in the system yielding a probabilistic automaton model of the computer system. A specific example and formal definitions follow.

Consider a system composed of two processes, P_1 and P_2, and two units of a resource. The following actions may occur in this example and correspond to transitions of the probabilistic automaton.

a. Process P_i may request a unit of resource, denoted r_i in which case the process is suspended (he waits) until a unit is allocated where $i = 1$ or 2. Process P_i can only request one unit at a time and never request more if all units (two in this case) have already been allocated to process P_i.

b. If process P_i is in a suspended state waiting for a unit of resource, the system may allocate a unit to P_i denoted a_i, provided all units of the resource have not already been allocated.

c. If process P_i is not in a suspended state, P_i may de-allocate a unit of resource which was previously allocated to him, denoted d_i, implying that the process no longer needs the unit.

Holt[8,9] has considered the System State Diagram for this system. By making some assumptions concerning the probabilities of state changes, a Probabilistic Automaton can be defined from this. Suppose, for example, that without any apriori knowledge of the system we assign equal probabilities to all transitions out of any given state. Then because these equal probabilities must sum to one, we get the automaton represented graphically by Figure 1. Each circle (node) of Figure 1, denoting a possible state of the system, contains the probability of leaving that state. This representation can be used in this case because of our equal probability assumption. Thus this number is simply the multiplicative inverse of the number of arcs leaving the node. The upper first node in row 00 and column 00 denotes a state (henceforth designated (00,00)) in which process P_i holds no resources and requests no resources, $i = 1,2$. In general, the column labels (row labels) are two digit numbers, the first digit specifying resources held and the second, resources requested by process P_1 (process P_2). The equal probability assumption implicitly imposes a random resource allocation algorithm upon the system because if the system is in a state such that both processes are suspended waiting for a unit of resource, then we randomly choose one or the other. The node in row 01, column 01, of Figure 1, is an example of this. If the system began in the initial state (00,00), (no allocation, no requests) then one way it could get to this state (01,01) is via a request by P_1, $(00,00) \xrightarrow{r1} (01,00)$ followed by a request by P_2, $(01,00) \xrightarrow{r2} (01,01)$. Another possibility is that P_2 could first make a request and then P_1, $(00,00) \xrightarrow{r_2} (00,01) \xrightarrow{r_1} (01,01)$. Thus the arcs are marked to identify the actions of the system (which correspond to transitions of the automaton). The probability of the automaton moving from state (00,00) to (01,01) in two transitions is 1/2, the sum of the probabilities of the paths which allow this to be done:

Path 1: $(00,00) \xrightarrow{1/2} (00,01) \xrightarrow{1/2} (01,01)$, pr(path 1) = 1/4
Path 2: $(00,00) \xrightarrow{1/2} (01,00) \xrightarrow{1/2} (01,01)$, pr(path 2) = 1/4

total pr = 1/2

Note that a path is the conjunction of a sequence of system actions, so the path probability is the product of the transition probabilities of the corresponding transitions composing the path. These concepts are formalized in the definitions which follow. Once the system is in the state (01,01), there is a probability of 1/2 of moving to state (10,01) and probability of 1/2 of moving to (01,10) because the node in row (01,01) has arcs to these two states and contains 1/2. This implies that our resource allocation algorithm allocates to process P_1 with probability 1/2 or to process P_2 with probability 1/2. Two other possible algorithms, first-come-first-serve and last-come-first-serve can be modeled by adding an auxiliary storage to our automaton, forming respectively a probabilistic queue automaton and a probabilistic pushdown automaton. In these cases, whenever a request is made by process P_i, the symbol r_i is put into the auxiliary storage, and whenever an allocation is made, the process P_i to whom this allocation is made is determined by the symbol r_i extracted from the storage. This symbol is then discarded. In the case of a queue, this will be the least recently inserted symbol among the symbols in storage and for a stack, this will be the most recently inserted symbol. The model is formally defined as follows:

Definition: A Probabilistic Language over a vocabulary Σ is a system $\hat{L} = (L,\mu)$ where L is a class of words formed from Σ (we will take this class to be the set Σ^* = all finite strings formed from Σ) and μ is a measure on the set L. If μ is a probability measure, then $\hat{L}$ is a Normalized Probabilistic Language.

Definition: A Probabilistic Automaton over a vocabulary Σ is a system $\hat{A} = (K, B, \delta, \alpha, \gamma)$ where

K is a finite, nonempty set of states, $s_1, s_2, \ldots, s_n$.

B is a finite set of storage tape symbols, $b_1, b_2, \ldots, b_m$.

$\gamma: K \rightarrow R$ is an initial state vector $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ such that $\gamma_i = \gamma(s_i)$ specifies the probability that s_i is the initial state, where R is the set of real numbers.

$\delta: K \times B \times (\Sigma \cup \lambda) \rightarrow \mathcal{R}(\delta)$ is a mapping called the transition function whose domain is $K \times B \times (\Sigma \cup \lambda)$. The range of this function determines the specific type of probabilistic automaton defined.

$\alpha: K \times B \times (\Sigma \cup \lambda) \times \mathcal{R}(\delta) \rightarrow R$ is a mapping called the transition probability function which associates a real value with each possible map of δ, to designate the probability of that transition.

Several different types of normalization of probabilistic automata have been presented in the literature[13,14]. A classification of normalization types is given in Ellis[2]. The appropriate type of normalization for our application is given next.

A. A Probabilistic Finite Automaton is defined by $\delta : K \times \Sigma \rightarrow P(K) \cup \{\lambda\}$ and $B = \phi$ where ϕ denotes the empty set and $P(K)$ the power set of K. This automaton is normalized if and only if γ is a stochastic vector, i.e., $\sum_{i=1}^{n} \gamma_i = 1$, and $\sum_{a \varepsilon \Sigma} \sum_{s' \varepsilon k \cup \{\lambda\}} \alpha(s, a, s') = 1$ where this implies $\alpha(s, a, s')$ is interpreted as $pr(s', a | s)$.

A transition is a change from some state $S_i \varepsilon K$ under an input $A \varepsilon \Sigma$ to some state $S_{i+1} \varepsilon K$ such that $S_{i+1} \varepsilon\ (S_i, A)$ and can be written $(S_i, A) \rightarrow S_{i+1}$. $\lambda \varepsilon (S_i, A)$ will be written $(S_i, A) \rightarrow$ halt. Associated with each transition is a probability; the product of these transition probabilities is the probability $\bar{p}$ of the sequence $S_0, S_1 \ldots S_t$. A mapping $\delta(S, A) = \phi$ has probability zero associated with it and designates that a transition out of the state S under input A is disallowed. The probability of acceptance of a string $A_1 A_2 \ldots A_n$ is

$$\sum_{i=1}^{m} \gamma\,(S_0)\bar{p}_i p_{if}$$

where m is the number of sequences $S_0 S_1 \ldots S_{n-1}$ such that $S_j \varepsilon (S_{j-1}, A_j)$, $j = 1, 2, \ldots, n-1$, and s_{n-1} satisfies $\lambda \varepsilon (S_{n-1}, A_n)$ with $p_{if} = \alpha(S_{n-1}, A_n, \lambda)$. The probabilistic language accepted by a probabilistic automaton M is (L, μ) where $\mu(z)$ = probability of acceptance of the string z by the automaton M.

A Probabilistic Pushdown Automaton and Probabilistic Queue Automaton can be analogously defined where the set B will be nonempty to represent the auxiliary storage. Simulation results presented in Appendix A indicate that FCFS and LCFS algorithms yield deadlock probabilities which are very close to the probabilities for the random resource allocation algorithm. Thus we concentrate on the random resource allocation case which is most tractible for mathematical analysis and throughout the remainder of the paper, automaton will mean probabilistic finite automaton. Research on models of FCFS and LCFS algorithms is currently in progress. These are non-Markov processes amenable to investigation via probabilistic automata theory rather than via Markov chain techniques. More will be said about these non-Markov processes in the final section of this paper.

<u>Definition</u>: A <u>Probabilistic Grammar</u> (P grammar) over Σ is a system $\hat{G} = (N, P, \Delta)$ where N is the finite set of nonterminals, $A_1, A_2, \ldots, A_n$, Δ is an n-dimensional vector, $(\delta_1 \ldots \delta_n)$ with δ_i being the probability that A_i is chosen as the initial nonterminal, and P is a finite set of probabilistic productions, $\psi_i \overset{p_{ij}}{\rightarrow} \zeta_i$, with $\psi_i \in (N \cup \Sigma)^*$, $\zeta_j \in (N \cup \Sigma)^*$, and $p_{ij} \in R (p_{ij} \neq 0)$. If Δ is stochastic, if $0 < p \leq 1$, and if $\sum_j p_{ij} = 1$ for every generatrix ψ_i contained in productions of P, then $\hat{G}$ is a <u>Normalized Probabilistic Grammar</u> (NP grammar).

If all productions of $\hat{G}$ are of the form $A \overset{p}{\rightarrow} aB$ or $A \overset{p}{\rightarrow} a$, $A \in N$, $B \in N$, $a \in \Sigma$, then $\hat{G}$ is called a <u>left linear P grammar</u>. Then define $pr(\zeta_0 \Rightarrow \zeta_n) = \sum_{i=1}^{k} \prod_{j=1}^{k_i} p_{ij}$ where k is the number of derivations of ζ_n from ζ_0, k_i is the number of derivation steps, $\zeta_{i,j-1} \overset{p_{ij}}{\rightarrow} \zeta_{ij}$ used in the ith derivation, and p_{ij} is the probability associated with the jth step of the ith derivation. The <u>derived probability</u> of a terminal string $x \in \Sigma^*$ with respect to a left linear grammar $\hat{G}$ is $\mu(x) = \sum_{i=1}^{n} (\delta_i \; pr(A_i \Rightarrow x))$ where $N = \{A_1, A_2, \ldots A_n\}$, $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$. The P language generated by $\hat{G}$ is $\hat{L} = (\Sigma^*, \mu)$ where $\mu(x)$ = the derived probability of x.

<u>Theorem 1</u>: The probabilistic language accepted by a normalized probabilistic automaton is necessarily a normalized language, i.e., the sum of the probabilities of all strings accepted by the automaton is one.

<u>Theorem 2</u>: Every left linear P grammar $\hat{G}$ generates a probabilistic language which is accepted by some automaton $\hat{A}$ and conversely, every probabilistic automaton $\hat{A}$ accepts a probabilistic language which is generated by some left linear P grammar $\hat{G}$. Furthermore, $\hat{G}$ is normalized if and only if $\hat{A}$ is normalized.

The theorems stated above which have been proven by the author elsewhere, immediately imply the following:

<u>Theorem 3</u>: The probabilistic language generated by a normalized left linear P grammar is necessarily a normalized language.

However, one must be cautious in defining probabilistic languages because of the following:

<u>Theorem 4</u>: There exists regular (nonprobabilistic) languages, L, with a probability $\mu(x)$ assigned to each $x \in L$ such that no left linear P grammar generates (L, μ).

Consider the application this formalism to the 2 process, 2 resource system described previously. The automaton of Figure 1 has nineteen states plus the deadlock state$^{(11,11)}$ marked X which is not in K. All transitions into deadlock must be λ transitions; there are two of these $(11,10) \xrightarrow{r_2}$ halt and $(10,11) \xrightarrow{r_1}$ halt, each having associated probability 1/2. A typical sequence of actions leading to deadlock could be the following:

1. Process one requests one unit of resource $(00,00) \rightarrow (01,00)$
2. Process one is allocated one unit of resource $(01,00) \rightarrow (10,00)$

3. Process two requests one unit of resource (10,00) → (10,01)

4. Process two is allocated one unit of resource (10,01) → (10,10)

5. Process one requests another one unit of resource (10,10) → (11,10)

6. Process two requests another one unit of resource (11,10) → (11,11)

Thus, the automaton has entered a final state meaning the system has become deadlocked. The probability of this sequence of events is the product of the probabilities of the events:

$$\mathrm{pr}[(00,00) \rightarrow (01,00)] \times \mathrm{pr}\,[(01,00) \rightarrow (10,00)] \times \mathrm{pr}\,[(10,00) \rightarrow (10,01)]$$
$$\times\ \mathrm{pr}\,[(10,01) \rightarrow (10,10)] \times \mathrm{pr}\,[(10,10) \rightarrow (11,10)] \times \mathrm{pr}\,[(11,10) \rightarrow (11,11)] =$$
$$1/2 \times 1/2 \times 1/3 \times 1/3 \times 1/4 \times 1/2 = 1/288.$$

This probability is very low, but there are many other paths to deadlock. How many transitions, on the average, are required before this system will be in deadlock? How does this average change as the number of processes and resources grow? We next consider several methods of arriving at these results.

STOCHASTIC EXPERIMENTS AND THE PROBABILISTIC ANALYZER

One simulation technique employed, called stochastic experimentation, is based on the concept of a grammar, and thus probabilistically generates sequences of system actions until deadlock occurs. Using this method a program (the model builder) was written which would accept as input a number n of processes, and a number r of resources and then build the probabilistic grammar to model this system under a random (or first-come-first-serve) scheduling algorithm. A second program, the simulator, used the model and a random number generator to generate a valid sequence of transitions (requests, allocations, and deallocations) which would terminate by entering a deadlock state. By running a large number of these simulations, the mean number of transitions to a deadlock state and the variance was obtained. This data correlated very well with the data obtained analytically via the methods described later. (See Appendix A.)

An alternative to performing a large number of experiments is the technique of executing the automaton nondeterministically to deadlock. Nondeterministic execution implies traversing all paths simultaneously by storing all possible transitions leaving the first state and their probabilities, then all second transitions from all possible second states with the probabilities of the combined pairs of transitions, etc., noting when deadlocks occur. The finite state automaton model can be classified as a Markov process and thus it is possible to obtain the following mathematical results. The probability of <u>not</u> being in a deadlock state after <u>n transitions</u> can be calculated as follows: Define $x_i^o = 1, s_i \varepsilon K$ and $x_i^n = \sum_{sj\ \varepsilon K} P_{ij}x_j^{n-1}$, $n \geq 1$, where P_{ij} is the probability of a transition from state s_i to state s_j. Then this yields a closed formula for x_i^n:

$$x_i^n = \sum_{j_1, j_2, \dots, j_n} P_{ij_1} P_{j_1 j_2} \dots P_{j_{n-1}, j_n}$$

Observe that x_i^n is the probability that, starting from state s_i, the automaton stays out of deadlock for the next n transitions. In the model of Figure 1, we calculated the smallest value of n such that the probability of staying out of deadlock for n transitions was less than 1/2 and found that n = 18. $P_1^{18} \doteq .4895$ is the probability of not being in deadlock after 18 transitions. Since $x_i^n \leq 1$ for all n > 0, it can be shown using induction that x_i^n is nonincreasing as a function of n. In fact:

$$x_i^2 = \sum_{j\varepsilon(K)} P_{ij}x_j^1 \leq \sum_{j\ \varepsilon\ (K)} P_{ij} = x_i^1$$

Now assuming that $x_j^n \leq x_j^{n-1}$ for all $j \ \varepsilon\ (K)$, we have

$$0 \leq x_i^{n+1} = \sum_{j\ \varepsilon\ (K)} P_{ij}x_j^n \leq P_{ij}x_j^{n-1} = x_i^n$$

Therefore, $x_i^n \downarrow x_i$. i.e., x_i^n decreases to some limit x_i and $x_i = \sum_{j\ \varepsilon\ (K)} P_{ij}x_j$, $i\ \varepsilon\ (K)$.

It follows that since the only bounded solution of this set of equations is the zero vector (0,0,...,0), starting from any nondeadlock state, transition to a deadlock state will eventually occur with probability one.

CALCULATION OF MEAN NUMBER OF STEPS TO DEADLOCK

Consider a generalization of the computer system shown in Figure 1 in which there are arbitrary numbers, m and r, of processes and resources respectively. Simulation results confirm the intuitive statement that as the number of processes increases with a fixed number of resources, the probability of deadlock increases. This statement seems intuitively reasonable due to the fact that adding more processes to a system generally puts the resources into greater demand. Likewise, as the number of resources increases with a fixed number of processes, the probability of deadlock decreases. This leads to the question of what happens to the deadlock probability if the number of processes and resources increase together. Simulation results for small r values (see Appendix B), and the analytic approximation for arbitrary r, which will be presented next, both yield the result that as the number of processes and resources increase together, that is assuming m = r, the probability of deadlock increases.

Define a family of random variables $d_\ell^{(r)}$, $\ell = 1,2,\ldots$, such that the value assumed by $d_\ell^{(r)}$ is the number of transitions to deadlock on the ℓth trial run of a system containing r processes and r identical units of a resource. Let $a_k^{(r)}$ = pr (k steps to deadlock) = pr $(d_\ell^{(r)} = k)$. Then we can define the mean number of transitions to deadlock for an r x r system as the expected value of the random variable $d_\ell^{(r)}$,

$$(1) \qquad E(d_\ell^{(r)}) = \sum_{k=0}^{\infty} k a_k^{(r)}$$

For a given r, the value of $a_k^{(r)}$ can be expressed in terms of x_1^k previously defined as the probability that starting in state 1 (the initial state), the system stays out of deadlock for the next k transitions. We will write x_1^k as $x_{(r)}^k$ since we will always be interested in starting from state 1. The relation is $a_k^{(r)} = (1 - x_{(r)}^k) - (1 - x_{(r)}^{k-1})$ since $x_{(r)}^k$ is cumulative and $a_k^{(r)}$ is not. The expression in (1) becomes $\sum_{k=1}^{\infty} k(x_{(r)}^{k-1} - x_{(r)}^k)$ which is a telescoping sum yielding

$$(2) \qquad E(d_\ell^{(r)}) = \sum_{k=0}^{\infty} x_{(r)}^k$$

This quantity will be used as a measure of the probability of deadlock of a system. Note that a smaller value for E implies a higher probability of deadlock, e.g., Appendix B shows $E(d^{(2)}) = 25.50$ and $E(d^{(3)}) = 23.48$, so a 3 x 3 system has a higher probability of deadlock. In general, an approximate comparison of the mean number of transitions to deadlock for an n x n system under random scheduling and an (n+1) x (n+1) system can be made as follows.

Consider the diagram of Figure 2. The bottom row of states denotes deadlock states of the system. These states are not members of K. The row above denotes states which can lead to deadlock in one request. These are the states, called primary blocked states, of our automaton which have λ transitions.

It can be shown that $\exists N \ni \sum_{k=0}^{N} x_{(n)}^{k} > \sum_{k=0}^{N} x_{(n+1)}^{k}$ by looking at minimal length paths to deadlock. In fact, it can be seen that $x_{(n)}^{N} > x_{(n+1)}^{N}$. Given this we show that $x_{(n)}^{N+1} > x_{(n+1)}^{N+1}$ which implies $\sum_{k=0}^{N+1} x_{(n)}^{k} > \sum_{k=0}^{N+1} x_{(n+1)}^{k}$ and thus the desired result $\sum_{k=0}^{\infty} x_{(n)}^{k} > \sum_{k=0}^{\infty} x_{(n+1)}^{k}$. $x_{(n)}^{N+1} = x_{(n)}^{N}(1-pr(d_{\ell}^{(n)}=N+1))$. As an approximation to $a_{N+1}^{(n)}$, we use $1/2 \frac{|B^{(n)}|}{|K^{(n)}|}$ where $B^{(n)}$ is the set of primary blocked states in an n x n system with 1/2 being the probability of a blocked state's entering deadlock via a single transition, and the notation $|A|$ denotes the cardinality of a set A. $x_{(n)}^{N+1} = x_{(n)}^{N} - x_{(n)}^{N} \frac{|B^{(n)}|}{2|K^{(n)}|} \geq x_{(n+1)}^{N} - x_{(n+1)}^{N} \frac{|B^{(n)}|}{2|K^{(n)}|}$ using the given relation $x_{(n)}^{N} > x_{(n+1)}^{N}$.

Appendix C gives expressions for $|B^{(n)}|$ and $|K^{(n)}|$ illustrating that $\frac{|B^{(n)}|}{|S^{(n)}|} < \frac{|B^{(n+1)}|}{|S^{(n+1)}|}$. Thus

$$x_{(n+1)}^{N} - x_{(n+1)}^{N} \frac{|B^{(n)}|}{2|K^{(n)}|} > x_{(n+1)}^{N} - x_{(n+1)}^{N} \frac{|B^{(n+1)}|}{2|K^{(n+1)}|} = x_{(n+1)}^{N+1}.$$

SUMMARY AND FURTHER RESEARCH

The probability of deadlock within simple systems consisting of a few processes and a few resources has been investigated using a probabilistic automaton model. This probability of deadlock was defined in terms of the expected value of the number of transitions of the automaton before entering a deadlock state. It was shown that as the number of processes and resources became large uniformly, the probability of deadlock increased.

The probabilistic pushdown automaton and probabilistic queue automaton models are currently being investigated. These models are generalizations of the probabilistic automaton investigated in this paper. Applications of techniques of automata theory to these models seem quite promising.

Useful results are available concerning the probability of termination of probabilistic context-free grammars[3] and concerning the transformation of automata to one state automata by increasing the set of storage tape symbols. It is hoped that this investigation will provide further evidence that probabilistic automata are useful tools in the study of deadlock, and that results concerning the more realistic FCFS and LCFS resource allocation algorithms will be forthcoming.

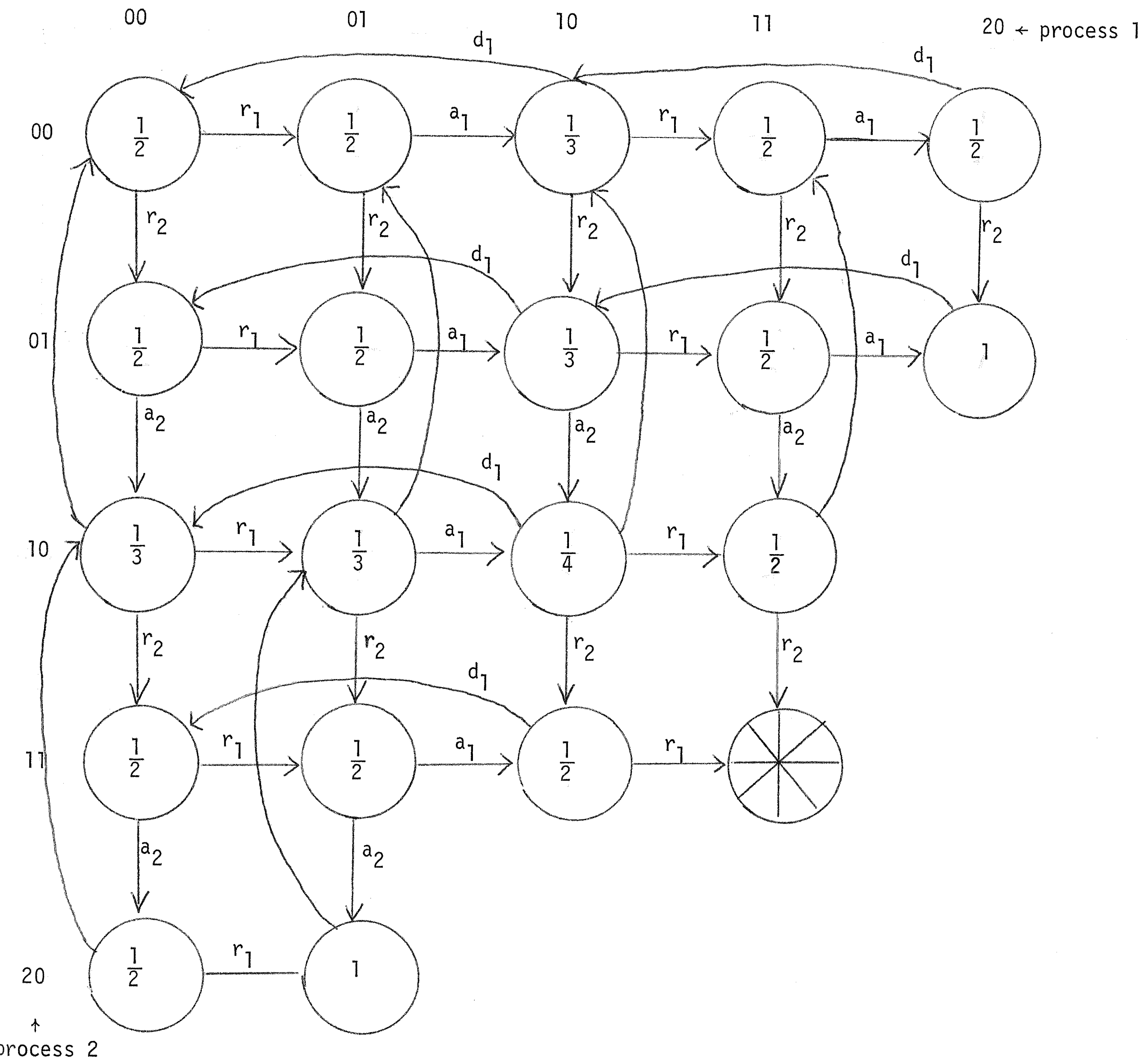


FIGURE 1

A PROBABILISTIC AUTOMATON CORRESPONDING TO A 2 x 2 SYSTEM

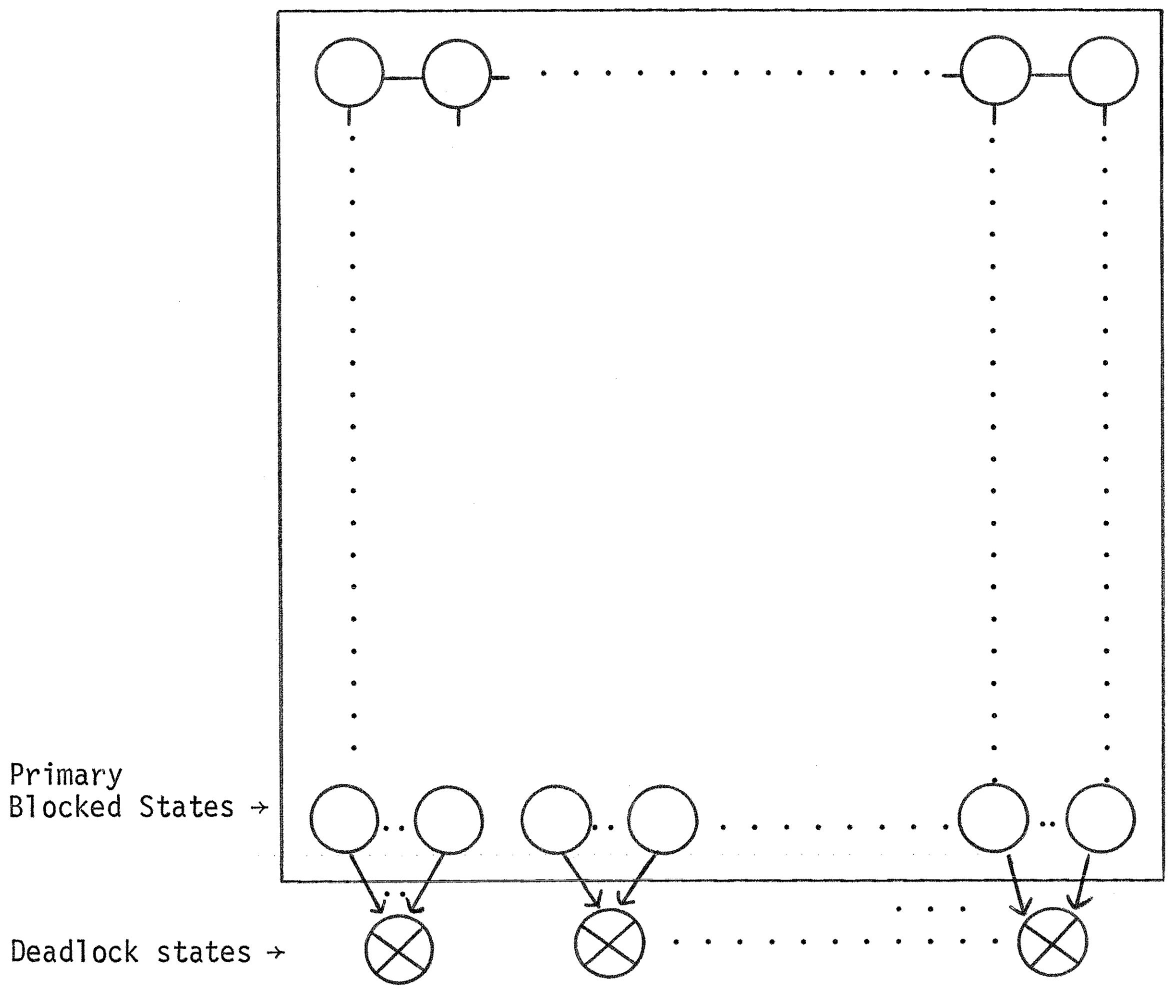


FIGURE 2

A PROBABILISTIC STATE SET CORRESPONDING TO AN m x r SYSTEM

REFERENCES

1. Coffman, E. G., Jr., Elphick, M. J., and Shoshani, A., "System Deadlocks," Computing Surveys, Vol. 3, No. 2, Pages 67-78, June, 1971.

2. Ellis, C. A., "Probabilistic Languages and Automata," Ph.D. Dissertation, Computer Science Department, University of Illinois, 1969.

3. Ellis, C. A., "The Halting Problem for Probabilistic Generators," presented at the Fourth Annual Princeton Conference on Information Sciences and Systems (also in Journ. of ACM, vol. 19, No. 3, July 1972).

4. Fontao, R. O., "A Concurrent Algorithm for Avoiding Deadlocks in Multiprocess Multiple Resource Systems," Proceedings of the Third Symposium on Operating Systems Principles, October, 1971.

5. Habermann, A. N., "Prevention of System Deadlcoks, "Communications of the ACM, Vol. 12, No. 7, Pages 373-377, July, 1969.

6. Havender, J. W., "Avoiding Deadlock in Multitasking Systems," IBM Systems Journal, Vol. 7, No. 2, Pages 74-84, 1968.

7. Hebalkar, P. G., "Deadlock-Free Resource Sharing in Asynchronous Systems," Ph.D. Dissertation, Electrical Engineering Department, MIT, Cambridge, Mass., September, 1960.

8. Holt, R. C., "On Deadlock in Computer Systems," Technical Report CSRG-6, University of Toronto, July, 1972 (also available as Ph.D. Dissertation, Department of Computer Science, Cornell University, Ithaca, New York, January, 1971).

9. Holt, R. C., "Some Deadlock Properties of Computer Systems," Computing Surveys, Vol. 4, No. 3, Pages 179-196, September, 1972.

10. Hopcroft, J. E., and Ullman, J. D., Formal Languages and their Relation to Automata, Addison-Wesley, 1969.

11. Murphy, J. E., "Resource Allocation with Interlock Detection in a Multi-Task System," Proceedings of the AFIPS FJCC, 1968, Vol. 33, Part II, Pages 1169-1176.

12. Nutt, G. J., "Some Applications of Finite State Automata Theory to the Deadlock Problem," Report CU-CS-017-73, University of Colorado, April, 1973.

13. Paz, A., Introduction to Probabilistic Automata, Academic Press, 1971.

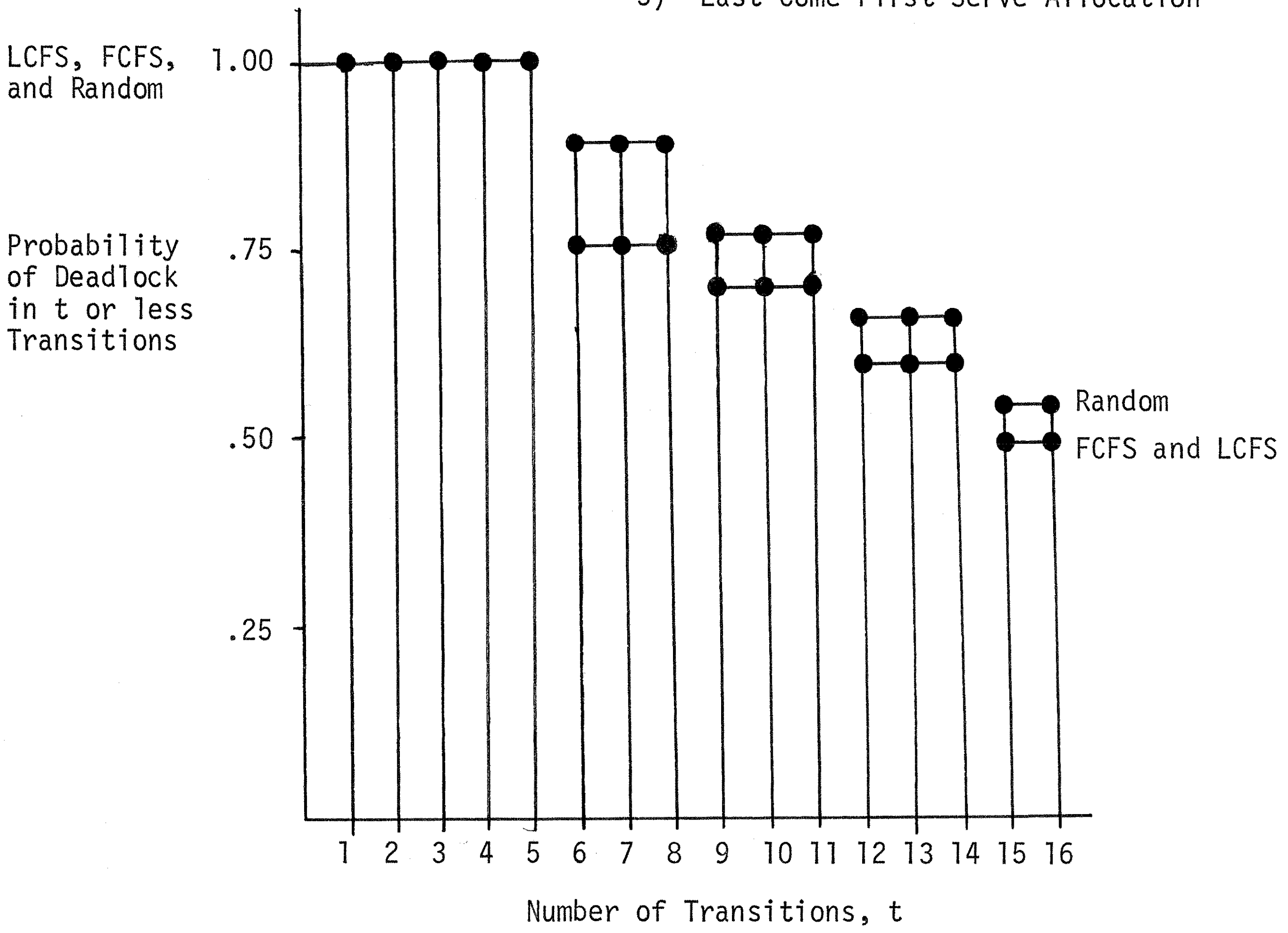
14. Rabin, M. O., "Probabilistic Automata," Information and Control, Vol. 6, No. 3, Pages 230-245.

CAE:Cah

APPENDIX A

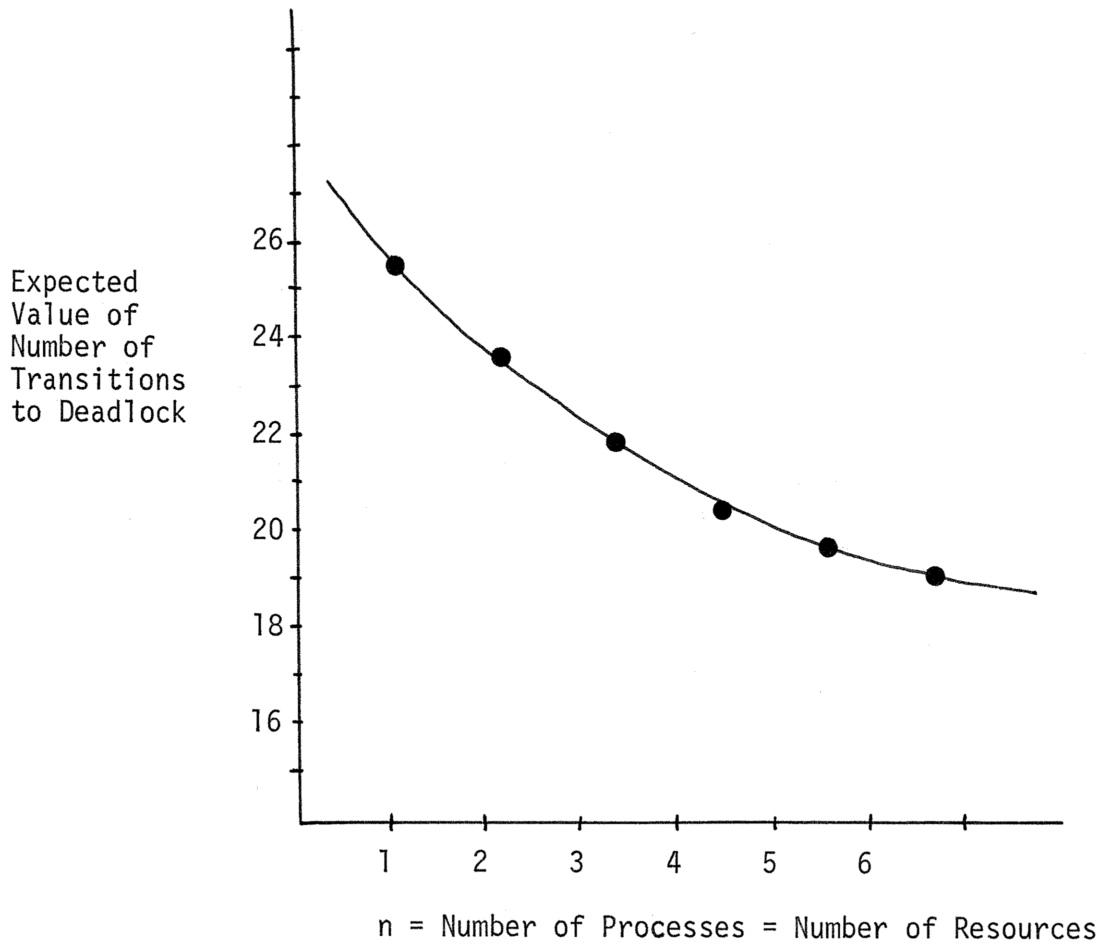
PROBABILITIES OF DEADLOCK IN A 2 x 2 SYSTEM FOR:

1) Random Resource Allocation
2) First-Come-First-Serve Allocation
3) Last-Come-First-Serve Allocation



APPENDIX B

MEAN NUMBER OF TRANSITIONS TO DEADLOCK FOR n x n SYSTEM



APPENDIX C

COMBINATORIAL ANALYSIS OF $|K|$ AND $|B|$

1. Counting of $|B|$ = Number of primary blocked states for m x r system.
   a. For each deadlock state, there are uncommitted processes out of a total of m processes. There are $\binom{m}{\mu}$ ways to choose processes. Uncommitted means the process does not hold any resources and thus does not contribute to the deadlock directly.
   b. For each of these choices, there are m-μ processes which must each hold an allocation of at least one resource and the sum of their allocations must equal the total number r of resources in the systems. This can be done in $\binom{r-1}{m-\mu-1}$ ways.
   c. For each of these ways, there are 2 permutations of the uncommitted processes between request = 0 and request = 1.
   d. The above imply that there are $\underset{\mu=0}{\overset{m-2}{}} \binom{m}{\mu} \binom{r-1}{m-\mu-1}$ 2 deadlock states.
   e. Each deadlock state, having μ uncommitted processes, has m-μ nondeadlock states which enter it corresponding to a request by each of the m-committed processes. These states are blocked states for m-μ-1 of the processes (i.e., these processes are in a wait state) but these states are not in deadlock unless the one transition which could lead to deadlock is the next system action. Thus these primary blocked states are distinct so we have not over-counted.

$$|B| = \sum_{\mu=0}^{m-2} (m-\mu)\binom{m}{\mu}\binom{r-1}{m-\mu-1} 2^{\mu}$$

2. Counting of total number of states for m x r system.
   a. First we count the number of states for a 2 x r system, and

then count for arbitrary m x r system. Our justification of the formula is by double induction. In the 2 x 2 case, we see there are 20 states = $2r^2 + 6r$. We use this as our induction hypothesis, which is true in the case r = 2.

b. The pattern for adding states to get from r-1 to r resources is to add two states per row (for the request and allocation of the added resource) plus an extra two states due to the fact that one process cannot hold all resources and request more. Thus assuming the induction hypothesis that in a 2x(r-1) system there are $s(r-1)^2 + 6(r-1)$ states, we can calculate the number of states in a 2xr system to be

$$\begin{aligned}[2(r-1)^2+6(r-1)]+2(2r-1)+2 &= 2r^2-4r+2+6r-6+4r+2+2\\ &= 2r^2+6r. \quad \text{q.e.d.}\end{aligned}$$

c. For a given r, our induction hypothesis claims that an m process system ($m \leq r$) will have the following number of states:

$$s_m^r = 2^{m-1} \sum_{i_{m-1}=1}^{r} \sum_{i_{m-2}=1}^{i_{m-1}} \cdots \sum_{i_1=1}^{i_2} i_1 + \sum_{i_{m-2}=1}^{r} \cdots \sum_{i_1=1}^{i_2} i_1 + \cdots +$$

$$\sum_{i_1=1}^{r} i_1 + (2r+1) + \sum_{i_{m-1}=2}^{r} \sum_{i_{m-2}=1}^{i_{m-1}} \cdots \sum_{i_1=1}^{i_2} i_1 +$$

$$\sum_{i_{m-2}=2}^{r} \cdots \sum_{i_1=1}^{i_2} i_1 + \ldots + \sum_{i_1=2}^{r} i_1 \ .$$

This equation can, for a particular (m,r) pair be reduced to a short form. Certainly for the case m=1, this holds because the number of states possible is 2r+1. Also in the 2 x r case, this reduces to

$$2\left(\sum_{i_1=1}^{r} i_1 + \sum_{i_1=2}^{r} i_1 + 2r+1\right) = 2\left(2\sum_{i_1=1}^{r} i_1 + 2r\right) = 2[(r^2+r) + 2r]$$

$$= 2r^2 + 6r$$

d. Given that the induction hypothesis holds for (m-1) x r systems, then for an m x r system we can partition the states into equivalence classes such that all states which have the same value for the number of resources held and requested by process m are in the same class ${}_mc_i$. The class ${}_mc_{00}$ has exactly the same number of states as an (m-1) x r system (s^r_{m-1}) and so does ${}_mc_{01}$ which can be seen by ignoring the mth process in these cases where process m holds no resources. Thus by counting the number of states in each of the remaining 2r-1 classes whose cardinality decreases uniformly, and adding to $2s^r_{m-1}$, we get s^r_m:

$$s^r_m = 2^{m-1} \quad 2\sum_{i_{m-1}=1}^{r} \; \sum_{i_{m-2}=1}^{i_{m-1}} \cdots \sum_{i_1=1}^{i_2} \left(i_1-1\right) + 2s^r_{m-1}$$

$$= 2^{m-1} \quad \sum_{i_{m-1}=1}^{r} \; \sum_{i_{m-2}=1}^{i_{m-1}} \cdots \sum_{i_1=1}^{i_2} i_1 - \sum_{i_{m-1}=2}^{r} \; \sum_{i_{m-2}=1}^{i_{m-1}} \cdots \sum_{i_1=1}^{i_2} i_1 +$$

$$2s^r_{m-1}.$$

By inserting the value of s^r_{m-1} implied by the induction hypothesis for m-1 processes, we get the correct corresponding value for m processes.