

PROBABILISTIC SEQUENTIAL MACHINE MODELING
OF
COMPUTER SYSTEMS
AND
ITS APPLICATION TO ERROR DETECTION

A Dissertation

Presented to

the Faculty of the Cullen College of Engineering

The University of Houston

in Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy in System Engineering

by

Hitohisa Asai

AUGUST 1975

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his major advisor, Dr. Samuel C. Lee for his advice and continuous encouragements throughout his years of graduate study at the University.

The author also wishes to thank Mr. Chuck E. Voss, Mr. Glenn E. Fisher and Mr. Richard O. Stewart, the staffs of the computing center of the University for their consulting supports.

PROBABILISTIC SEQUENTIAL MACHINE MODELING
OF
COMPUTER SYSTEMS
AND
ITS APPLICATION TO ERROR DETECTION

An Abstract of

A Dissertation

Presented to

the Faculty of the Cullen College of Engineering

The University of Houston

in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in System Engineering

by

Hitohisa Asai

AUGUST 1975

ABSTRACT

The problems of computer system modeling and of error detection in a computer system are investigated in this research. Probabilistic sequential machine modeling of a computer system is proposed by considering input/output flexibilities on the probabilistic sequential machine theory. With the model, a theoretical approach and a practical approach to error detection are presented.

In the theoretical approach, a two-state isolated machine, which contains the well-known completely isolated machine as a subset, is constructed, and a decomposition method of a probabilistic sequential machine into two-state probabilistic sequential machines is studied. Based on the isolated machine and the decomposition method, properties of the isolated machine (which consists of states more than two), such as input traceability, past subsystem activity distribution, and the initial state distribution independence, etc., are discussed. Traceback properties of the machine are used for error detection.

From distribution of input types to and output channel activities from a computer system, the probability of subsystem activities in a steady state is determined in the practical approach by an optimization of a nonlinear programming problem. The nonlinear programming problem is formulated with two phases, a calibration and a monitoring of the computer system. Next, the most likely subsystem which contains an error is determined. A probabilistic sequential machine model of a computer system is built. The advantages of the practical approach are demonstrated on the model which is simulated in a normal computer operation.

TABLE OF CONTENTS

CHAPTER	PAGE
I. INTRODUCTION	1
1.1 Statement of the Problem	1
1.2 Description of Computer System	1
1.3 Past and Present Efforts	11
1.4 Description of the Research	13
II. COMPUTER SYSTEM MODELING	16
2.1 Justification for Modeling Computer Systems by Probabi- listic Sequential Machines	16
2.2 Graphical Representation of a Computer System	21
2.3 Sequential Machine from a System Viewpoint	27
2.4 Deterministic Sequential Machine	30
2.5 Probabilistic (Stochastic) Sequential Machine	35
2.6 Definition of a Generalized Probabilistic Sequential Ma- chine	40
2.7 A Model Used with Generalized Probabilistic Sequential Machine	41
III. TWO-STATE INPUT-TRACEABLE PROBABILISTIC SEQUENTIAL MACHINE . .	48
3. Introduction	48
3.1 Completely Isolated Machines	49
3.2 The Product of 2×2 Stochastic Matrices	52
3.3 Two-State Isolated PSM	58
3.4 An Absolutely Isolated Machine	68
3.5 The k th Approximation of the Absolutely Isolated Machine	69
3.6 Two-State Input-Traceable Machine	84

CHAPTER	PAGE
IV. DECOMPOSITION OF A PROBABILISTIC SEQUENTIAL MACHINE	91
4. Introduction	91
4.1 An Interconnecting Property of N Two-State Machines . . .	92
4.2 Composition of Two-State PSM's	96
4.3 A Decomposition by Two-State PSM's	99
4.4 Decomposition of a Matrix Which Consists of Dependent Relations Among its Rows	124
4.5 Multiplication of Stochastic Matrices by Decomposed Two- State Matrices	126
V. N-STATE INPUT-TRACEABLE PROBABILISTIC SEQUENTIAL MACHINE . . .	139
5. Introduction	139
5.1 Decomposable PSM by Kronecker Product Matrices	140
5.2 Decomposable PSM by $N(N-1)/2$ Two-State PSM's	150
VI. ERROR DETECTION IN A LARGE COMPUTER SYSTEM	153
6. Introduction	153
6.1 Model Building	155
6.2 An Estimation of the Most-Likely Active Subsystem	165
6.3 An Example of the Computer System Modeling	170
6.4 Computer Simulation of the Example	190
VII. CONCLUDING REMARKS AND SUGGESTED FUTURE RESEARCH	197
BIBLIOGRAPHY	201
APPENDIX A	206

CHAPTER 1

INTRODUCTION

1.1 Statement of the Problem

As the range of problems undertaken by digital computer systems increases, the task of ensuring that a computer system is operating correctly has steadily become more important. The rapid growth of computer technology has thrust into prominence: the remarkable advancement of hardware computing capacity and the executive/supervisory software that we refer to as computer operating systems. These prominent advancements have resulted in a massive and extremely complex organization of computer systems. Thus, with the present trends in computer systems, a heavy emphasis on stability/reliability, maintainability, and availability is prevailing rather than on computation capability, speed of computation, memory capacity, and speed of input/output processing.

The expanding size and complexity of the computer systems have made it increasingly difficult to ensure correct machine operation. Moreover, the cost and time for correcting an error in a computer system is multiplying exponentially with the system's size and complexity. Therefore, an efficient and economical error detection method for large computer systems is urgently needed.

1.2 Description of Computer System

The term "computer system" means a combination of hardware and software components that provides a definite service. A computer system provides representations for certain data types and information structures, and it implements a set of primitive operations on these data types and structures.

Data and information in the computer system are simply all possible contents of the main memory. Selection of a desired component of these information structures is accomplished through indexing or memory addressing by base registers. The interpretation of memory words as data types and information is performed implicitly by a central processing unit. Data manipulation on selected information (usually addition, subtraction, multiplication, division, logical operation of bits-AND, OR, etc., and shift bits) is executed within registers of the arithmetic unit. The services provided by the computer system are furnished through combinations of hardware and software components.

Computer system hardware operates with information, numerical or otherwise, represented in digital form. From a viewpoint of the hardware structure, the computer system can be ideally described as an aggregate of two-valued memory devices, functionally connected by logical networks. The states of these memory devices undergo discrete changes at certain instants of time. Their initial states are set up by software.

The operating system, known as software, is a sequence of instructions by which a given, specified function is performed in a sophisticated manner by the central processing unit. Thus, the computer system works as if it is a high-level organism. Conceivably, the internal organization of the computer system may be very complicated. The system actually consists of many individual elements (or subsystems), which are interconnected and must run simultaneously.

The structures of a computer system have continued to grow in complexity, size, and diversity. The classical four-box picture of a computer

system (arithmetic unit, memory, input/output unit, and control unit) is certainly an effective organization of components to process information. However, multiprogramming, multiprocessing, hierarchies of memories, virtual memory, and remote communication to computer systems force the classical four-box representation of the organization into distinct levels of analysis of each component. Each level originates from the abstraction of the levels below it. Each higher level does a descriptive job in a simpler way, which the lower level could not because of the unnecessary detail that could force function looping. A system (at any level) is characterized by a set of components (of which certain properties are posited) and a set of ways of combining components to present systems. When the sets are formalized appropriately, the behavior of the systems can be determined by the behavior of its components and specific modes of the component combination used.

There is a recursive feature to most system descriptions. A system, composed of components structured in a given way, may be considered a component in the construction of yet other systems. There are, of course, some primitive components whose properties can not be explained as the resultants of a system of the same type. For example, a resistor, which can not be explained by a subcircuit, can be taken as a primitive in hardware, and certain instructions in software can not be replaced by other software instructions. However, sometimes there are no absolute primitives, and what basic element is taken is a matter of convention. For example, we can build logical design systems from many different primitive sets of logical operations (AND and NOT, NAND, OR and NOT, etc., in hardware and Execute, Test, Set, Erase, Find, Move, Locate, insert, Read, Write and Print, etc., in software).

Level		Hardware	Software
Major Component Level	Structure	Computer System Configuration	System Generation
	Component	Processors, Memories, I/O facilities	Processors, Memories, I/O facilities
Programming Level	Structure	Programs, Subprograms,	Job Control Language
	Component	State (Memory Cell), Instruction, Operation, Control, Interpreter	File Assign, File Manipulation, Software Processor Execution, Test and Control
Logical Design Level	Structure	Major Physical Unit	Major Task of Programs (Structured Programming)
	Component	Module of Subfunctional Units	Module of Subtask, Control of Subtask, Subroutine Groups
Prime Task Design Level	Structure	Switching Circuit, Combinational and Sequential Circuits	Problem Oriented Programming Languages FORTRAN, COBOL, Assembly
	Component	Flip-flop, Reset-set, Toggle, Delay, AND, OR, NOT, NAND and NOR	Main Program, Subroutine
Primitive Level	Structure	Amplifier, Delay unit, Attenuation, Clock, Gate, Differentiator,	Flowchart of Instruction Execution
	Component	Resistor, Capacitor, Inductor, Diode, Transistor	Machine Instruction, Logical Operation

Table 1.1 Hierarchy of Levels of Computer Systems

The computer system can be considered as a multilevel structure, where each level can be analyzed in the same manner. Table 1.1 shows several structure levels of the computer system. Each level is characterized by a distinct language for representing the system which is constituted of components, modes of combination, and laws of behavior. These distinct languages reflect special properties of the component types and of the way they combine.

Each level in Table 1.1 actually has two languages or representations associated with it: an algebraic one and a graphical one. These are isomorphic to each other; the same entries, properties, and relations are given in both. The lowest level, the primitive level, of the hardware in Table 1.1 is a circuit level. The components are R's, L's, C's, voltage sources, and nonlinear devices. The behavior of the system is measured in terms of voltage, current, and magnetic flux. These are continuously varying quantities associated with various components; therefore, there is continuous behavior through time. The structure of the circuit level can be described symbolically by first writing the relationship which defines each of the components (i.e., Ohm's law, Faraday's law, etc.) and then by writing the equation which defines the interconnection of the components (i.e., Kirchhoff's law). We observe circuit behavior by applying an input and observing the output. Alternatively, if we solve the equations which specify the structure, we obtain expressions which describe the behavior explicitly. Actually, even at an abstract level, circuit theory is not quite adequate to describe computer system technology.

The next hardware level is the switching circuit. The behavior at this level is described by discrete variables. Discrete variables take on

only two values, called 0 and 1, + and -, true and false, or high and low. The components perform logical functions: AND, OR, NOT, NAND, etc. The laws of Boolean algebra are used to compute the behavior of a system from the behavior and properties of its components.

The outputs of combinational circuits are directly related to the input at any instant of time. If the circuit has the ability to hold values over a period of time (store information), we have sequential circuits. In a symbolic representation of the structure, we can write an expression that reflects the structure of the combinational network; the product of a set of outputs at time t is a function of the number of inputs at the same time t . Boolean equations as an expression no longer reflect the actual structure of the combinational circuit but become a model to predict its behavior.

A representation of a sequential switching circuit is basically the same as that of a combinational circuit, although one needs to add memory components, such as a delay element which produces a delayed output at time t due to the input at time $t-\tau$ where $\tau > 0$.

A lower level is concerned with explaining the behavior of a certain structure, whereas the next higher level takes this lower level as a given component. The higher level is concerned not about internal behavior but only how components (which are in lower level systems) are combined. To express behavior and structure, there are two basic representations of systems in the higher level. One is "Register Transfer (RT)" [7] and the other is "State System." The components of an RT representation are registers and functional transfers between registers. A register is a

device that holds a set of bits. The behavior of a system is given by the time course of values of these registers. The system undergoes discrete operations, whereby the values of various registers are combined according to some rule and then are stored in another register (thus "transferred"). The law of combination may be almost anything, from the simple unmodified transfer ($A \leftarrow B$) to the logical combination ($A \leftarrow B.AND.C$) or to the arithmetic ($A \leftarrow B+C$). A specification of the system behavior is a set of expressions (often called productions) which give the conditions under which such transfers can be made. The Register-Transfer expression has emerged from informal attempts to create a notation closer to the job to be performed. Recently, a formalized Register-Transfer expression system [33] has been proposed as a tool for system description.

The second representation is the State-System [82], which is the most general representation of a discrete system. In a State-System, the system is represented by n abstract states and the state transition rules among the abstract states. While the system is in an abstract state, which represents the system performing a function (n is finite or enumerable), the next state of the system is determined (and the output associated with the state transfer) by a transfer function that takes as arguments the current state and the current input.

A digital computer system (hardware as well as software) can be represented as a State-System, but the number of states required is far too large to make it useful to do so. However, the State-System becomes a useful representation in dealing with various subsystems of the total

system and with the abstracted higher level description of the computer system. Taking a sequential circuit that controls a line printer as an example, the number of states of the printer is small enough to be handled. Another example is a small module of a task taken as a subroutine of a large programming job could be described with a small number of states by the State-System so that the task to be performed becomes very clear.

The entire software column in Table 1.1 can be embedded in the frame of the programming level of the hardware column. The components of the programming level in the hardware column are a set of memories and a set of operations. The operations take various data structures as inputs and produce new data structures, which again reside in memories. Thus, the behavior of the system is the time pattern of data structures held in its memories. The unique feature of this programming level is the representation it provides for combining components, that is, for specifying what operations are to be executed on what data structures. This is the programming which consists of a sequence of instructions. Each instruction specifies that a given operation (or operations) be executed on a specified data structure. In addition to this, a sequence of instruction executions is controlled by a given data structure. Normally, this is done in the order in which the instructions are stored in memories, with jumps out of the sequence by branch instructions directed by the given data structure. This kind of data manipulation has come to play a critical role in decision making of Management Science.

A technique to describe the programming description is the utilization of the "Decision Table" [38] which is a tabular representation of:

1. Conditions: factors to consider in making a decision.

2. Actions: Steps to be taken when a certain combination of conditions exists,

3. Rules: Specific combinations of the conditions and the actions to be taken under these conditions.

Normally, the decision table format is in two matrix forms: conditions (in rows) versus rules (in columns) and actions (in rows) versus rules (in columns). When we assign appropriate independent variables as input for the conditions, appropriate dependent variables as output for the actions, and proper operations on these independent variables to produce dependent variables for the rules of the decision table, the description of the decision table becomes a set of logical equations (possibly multivalued logic). Most of the well known techniques of Boolean algebra may be applicable to these logical equations derived from the decision table. "Formal system" [15] is becoming important in the design, implementation, and study of programming language as well as programming itself. A formal system is an uninterpreted calculus or logistic system. It consists of an alphabet, a set of words called axioms, and a finite set of relations called rules (or productions) of inference. Examples of formal systems are: Peano arithmetic,[†] propositional and predicate calculus, and Post systems.

The computer configuration system described in the highest level of Table 1.1 consists of central processors, core memories, tapes, disks,

[†]Peano's axioms are concerned with the natural number. However, Bertrand A. Russell (Introduction to Mathematical Philosophy 1919, London-New York) completed Peano's work (Formulaire de Mathematiques 1895-1905) using Frege's logicism (Grundlagen der Arithmetik 1884) in the aspect of logicizing of mathematics.

input/output processors, communication lines, printers, tape controllers, remote terminals, etc. The system is viewed as processing on three levels: medium, information, and organized data. Each component has its own operating characteristics. All details of the programming level are suppressed, although many softwares play key roles in data transactions and functional relations between these components. Nevertheless, this level is more abstract than the logical level and the programming level of the computer system. As the complexity of the computer system increases, the level of abstraction also increases. Another indication of the emergence of the major component level (system configuration level) lies in the model used in most operation research types and simulation techniques in computer system studies, such as capacities, total flow rates, bottlenecks, queuing problems, and buffer sizes of information flows [3]. We have been mainly describing the computer system in the hardware viewpoint which contains non-hardware elements, or softwares, as the programming level. The level is very special, because it means that the importance of software is recognized in hardware operations. When the level is focused on and refined in a different viewpoint, we can see that another distinct hierarchy among softwares exists as shown in the software column Table 1.1. There is practically no consensus on the nature of this software system level; this is not surprising because of the state-of-the-art in programming. A main emphasis here is that there are a lot of analogies between the systems of hardware and software, and each level (mainly in the hardware column) corresponds to the technologies that are available for the analysis and synthesis of the computer systems. The combinational and sequential logics are special highly polished technologies.

1.3 Past and Present Efforts

Methods of error (or fault) detection in hardware have been considered by Yau and Tang [81], Sellers et al., [64], Roth et al., [58], and Su and Cho [67]. The first two methods are based on the Boolean difference for fault detection; the third one defines and applies the D-algorithm, which is an extension of Eldred's classical work [17]; and the the fourth one is an error-locating method using the D-algorithm. All four methods are deterministic and require exhaustive searching, and hence, they are costly and time consuming when applied to large computer systems.

Hardware failure in a computer system is referred to as a physical component failure, such as a transistor failure due to its malfunction or its life's extinction. In other words, a hardware failure is a temporary or permanent change in the component's characteristics that alters its function.

Software does not fail. What is often referred to as "software failure" is a matter of correctness. Correctness of a software system means correctness of its program description with respect to the software system's objective as specified by the semantic sentence. Regardless of the approach taken to favor correctness of a software system, it is always the responsibility of the designer to convince himself of the correctness of the system's description.

Two approaches to the correctness of a system have been suggested: (1) structured programming [13] and (2) proof of correctness [41]. To correct an error, the following debugging techniques are available: post-mortem dump, snapshot, trace, and traceback [5]. All the techniques

are basically operated in a trial and error mode before approaching the kernel of an error and, thus, are time-consuming.

These deterministic error detection methods are well established techniques for small scale system testing. However, the quantity of the input/output to and from a computer system is usually large; thus, the total number of test sets which are combinations of various inputs are exponentially large. Furthermore, each test sequence of the test sets, which is constructed to identify different logical paths in the system, is extremely long. Direct application of these deterministic error detection techniques to a large system like a computer system is practically prohibitive, therefore, partitioning or segmenting the large system into a number of smaller subsystems purely from the structural viewpoint [54] is necessary. Test sets and test sequences for each segmented subsystem become smaller and shorter, respectively, so that the error detection techniques are practically applicable. When applying the error detection to each segmented subsystem by performing independent, individual subsystem tests, a whole system viewpoint of testing may be lost.

It is desirable to have a method which makes a bridge from a whole system testing viewpoint to the individual subsystem tests. In other words, could we narrow the area of error in the system by knowing an error symptom? The next problem is how to choose a subsystem to be tested for an error. One solution is a table for looking up the correlation between error symptoms and possible subsystems concerned with the error. To construct the table would require a large amount of error data and correlation analyses between errors and subsystems, which have been gathered in

a trial and error manner over a long time period. It should be pointed out that no such table exists for a brand new system.

It is well known that the past performance history of a system is useful for error detection [5]. The past performances of subsystems in a system may provide a bridge from the technique of system segmentation to the available deterministic error detection methods. Therefore, an inexpensive but efficient method for evaluating past subsystem performances is desired. To achieve this goal, a statistical modeling and method to determine the error containing segment (subsystem) is proposed, instead of the trial and error procedure.

In this research, the meaning of the words "application to error detection" shall be limited to the past performance evaluation of subsystems in a system.

1.4 Description of the Research

This research investigates the problem of error detection in large computer systems. When the past performance of a system can be determined by some method and made available to the system designers/maintenance engineers, such information is useful for detecting errors in the system. Thus, the past performance determination techniques presented here are directly applicable to error detection.

The research consists of the following three parts: (1) the modeling of large computer systems, (2) a theoretical approach to error detection in the model, and (3) a practical approach to error detection in the model.

In modeling, a probabilistic sequential machine S for large computer systems is used. By using this model, a theoretical approach may be applied to a special class of systems whose past behaviors are traceable. This traceable property can be used for system diagnosis. In the practical approach, the error may be located in a subsystem s in an erroneous system S during its operation. The visiting probability distribution of activity on all subsystems is determined from the frequencies of the input load distribution and the activity of its output. The practical approach is more powerful than the theoretical one. These two approaches are applicable to both the hardware and the software of a computer system.

A brief description of the thesis by chapter is given below. Chapter II begins with the justification of modeling computer systems by probabilistic sequential machines (PSM). A generalized model of PSM is proposed of which the input to and the output from the machine are described by a stochastic vector and a matrix, rather than a single input symbol and a vector, respectively. This generalized PSM is then used for modeling the structures and behaviors of large computer systems. The most basic PSM which has only two states is studied in Chapter III. The input traceable machine, a special class of the basic PSM whose past inputs are traceable, is defined. When using the input traceable property the error in an input-traceable machine may be determined by finding a subsystem of the whole system which has been most active in the past operations. Decomposability of a large PSM into basic PSM's is presented in Chapter IV. It is shown that a (large) PSM which is decomposable into basic PSM's is shown to also be input-traceable if each decomposed basic PSM is input traceable. The error detection in an N -state

input-traceable PSM is discussed in Chapter V. Since the class of input-traceable PSM class is a subset of the set of PSM's, it can only model the subset of the set of computer systems. However, the class of input-traceable PSM's can be enlarged by approximating the infinite input sequence length by a finite one. This class of machines is called k-input-traceable PSM's, where k denotes the finite length used in the approximation. An even more practical technique of error detection based on a mathematical optimization scheme is presented in Chapter VI. This technique can be applied to error detection of any generalized PSM, input-traceable or not, and hence of any computer system. Finally, concluding remarks and suggested future research are included in Chapter VII.

CHAPTER II

COMPUTER SYSTEM MODELING

2.1 Justification for Modeling Computer Systems by Probabilistic Sequential Machines

A computer system is an example of a large system composed of many interacting subsystems or modules. Due to the size of the computer system, it is essential that module's rules are specified in such a way that module descriptions are independent of the pattern in which they are interconnected to form the whole system; each module's behavior must be clear and correctly understood regardless of the situations in which the module is used. All interactions between modules must be through explicit points of information flow and control of task execution by the designer of each module.

If two modules or subsystems of a system are independently designed, then the timing of events within one subsystem can only be constrained with respect to events in the other subsystem as a result of interaction between the two subsystems. So long as no interaction takes place, events in two subsystems may be processed concurrently and no definite time relationship will exist between them. A time relationship between independent actions of separate parts of a system makes comprehension of the system's behavior more difficult. Concurrent and asynchronous operations in multiprocessing and multiprogramming systems with interruption mechanisms are fundamental aspects of a computer system.

When we consider the situation of multiprogramming and multiprocessing of a computer, the possibility of asynchronous interruption and hardware interlock make execution of hardware nondeterministic; that is, there may be many successor states possible for a given state of the system. The hardware

facilities for process switching and interrupt processing are usually controlled by software for interprocess communications, which are implemented by the scheduling modules of the operating system. The environment for software execution consists of the central processing unit on which the software runs, together with any hardware components other than subsystems which are required by the software. The software and hardware must interact as a system in order to realize some desired function; certain hardware of the computer system is called the host system of the software. The host system may be the central processing unit and main/auxiliary memory hardware.

An operating system is a software system having many software modules and appropriate mass storage devices. The software system consists of an editor, an interpreter, a command processor, facility inventory controller, scheduler, core allocation, job dispatcher, and input/output co-operative controller.

The designers of software and hardware wish to achieve certain goals. The goals are expressed in terms of five properties desired of the completed software and hardware systems: function, performance, correctness, reliability/stability, and maintainability. Some available techniques to achieve these goals are described in Section 1.2, Description of Computer System. For a large system, it is very important to provide the last three goals: correctness, reliability/stability, and maintainability. Some suggestions for achieving these goals have been proposed and have been implemented in large systems. Examples are structural hierarchies, mechanisms for protection from alien activity, modularity of subsystems/components, portability of system/subsystems/components which are concerned with changes in the operational environment, adaptability of system/subsystems/components which is also concerned with changes in the logical structure of the system, and testing/diagnosis. Although the designers are working on these considerations and very careful

design processes, attaining the goals is almost impossible. The reasons for failure are the large size and complexity of a computer system. For example, problems are created by poor communication between designers and engineers, logical errors in interfacing subsystems, logical loopholes in parallel multi-activities of system performance and undefined boundaries of system definitions, etc.

Subsystem testing, system integration testing, diagnosing of system functions, and system monitoring are currently evoking a great deal of interest in the digital computer field, and a number of significant advances have been made. Noteworthy advances are the D-algorithm [58] in hardware and the work of Vienna's group who were able to demonstrate an error in an IBM PL/I Compiler by formal methods [30] in software.

When engineers and programmers commence testing, diagnosing, and monitoring of the system, they should be reminded that these actions can never show the absence of errors but only the presence of errors. Testing is demonstrating the presence of an error, diagnosing is identifying what kind of error it is and where it occurred, and monitoring is collecting errors during the system's normal operation. We shall subject ourselves to the monitoring and diagnosing of the system.

The system's normal operation is a steady-state behavior in daily operations. Northouse and Fu [48] indicate that dynamic scheduling of large computer systems leads to a steady state of the system. The performance of a system in a steady state is some reasonable function of the input (more precisely the past input as well as the present input), thus the change in the system's performance due to the present input is predictable. For example, if the input changes from jobs of one type to jobs of a slightly different type, then there may be a little unpredictable behavior until the system

settles down to another steady state which is a function of the new job.

A set of inputs belonging to a job type always requires an expected variety and degree of computer services, but a set of inputs in another job type also asks the other expected variety and degree of services. These varieties and degrees of required services for each job type can be described by probabilities of the system's function executions since we seldom, if ever, know the exact sequence of jobs or requested services, which are submitted to a computer system. Furthermore, we do not know the exact characteristics of the jobs or the requested services. Ordinarily, the most we know (or can estimate) is the probability distribution of such quantities as classifications of the job type and its resource demands on the system.

The system is in a steady state while processing inputs of one job type. This operation of the system can be regarded as a stochastic process, which is a family of random variables in time. A stochastic process can also be considered as a time series of the mixed variables $\{x_t\}$ consisting of job type inputs, asynchronous interruptions and interlocks of hardware resources caused by input/output processing, which changes the execution course of function sequences. Since $\{x_t\}$ is a sequence of discrete random variables, there must be a probability distribution assigned to the sequence. If an appropriate sampling time to measure the system's executing functions is chosen, the probability that the system shall perform a function b at the next sampling time, depends only on the executing function a at the present sampling time, or

$$P\{x_{t+1} = b | x_t = a, x_{t-1} = a_1, x_{t-2} = a_2, \dots\} = P\{x_{t+1} = b | x_t = a\}$$

where a_1, a_2, \dots are the other system functions.

It may be said that the evolution of the system, given the present time (that is $x_t = a$) is independent of the previous states assumed. This stochastic process is a Markov chain. The other steady state caused by inputs from the other job type can be described with another Markov chain. A set of Markov chains, which can be distinguished by each job type (or input symbols in the terms of probabilistic sequential machines), is a probabilistic sequential machine. Because this stochastic property is basic to an operating system, we should expect that it will also show up with some acceptable precision in a logical model using a probabilistic sequential machine.

Next, suppose that input types are mixed. When the mixing ratio is known, system functions performed for the given inputs can be predicted probabilistically by combining the probabilities of Markov chains associated with each job type in the inputs. Reversely, when the mixing ratio of input types is unknown, the unknown ratio could be identified by comparing collected statistics of performed system outputs with the probabilities of Markov chains. In these aspects, system monitoring is necessary.

In order to monitor a computer system, we need to know the system's structure and organization; most systems are sufficiently complex that any useful model aids in the understanding of the system in some precise aspects but not in detail. A complete, detailed description of the system is generally not useful, since it contains a large amount of unnecessary information and does not explicitly exhibit the relationships between basic system functions. A desirable model is an abstraction containing only the significant functions and relations to satisfy a particular purpose. Graphical representation of a computer system may provide a useful tool for abstracting the system.

2.2 Graphical Representation of a Computer System

One of the simplest representations of a computer system is a directed graph, which basically shows function sequence and information flow. Generally, some of the sequence and flow details have been suppressed, and some additional information which may help to understand the system has been added. A directed graph is a set of nodes and directed arcs (sometimes called edges). Each arc in the graph originates at a node and terminates at a node (possibly the same node). More than one arc may originate or terminate at a single node.

In modeling with a directed graph, the arcs represent the paths of possible function sequence performed by the system (or by the subsystem). The nodes represent individual functions of the system; input to a node (or a function) and output from the node (or generated from the function) are attached along with the node or a path associated with the function sequence. Branch points are represented by nodes with more than one arc originating at the node. Additional information may be associated with the node and arcs. For example, the probability that function sequence exists from a branch point along a given arc is often associated with the arc of a directed graph.

Ramamoorthy [53] developed an algorithm based on the connectivity matrix derived from a directed graph. By his generating functions of the directed graph, he pointed out a direct analog between software and discrete electrical circuits, which are characterized by entry and exit points in software and sources and sinks in hardware. The connectivity matrix can be explained by the following example. Figure 2.1 shows a directed graph (which is omitted information flow).

The connectivity matrix of the graph is shown in Figure 2.2. In particular, the arc from node i to node j describes the following functional relation. After performing the function i and with certain output generated by

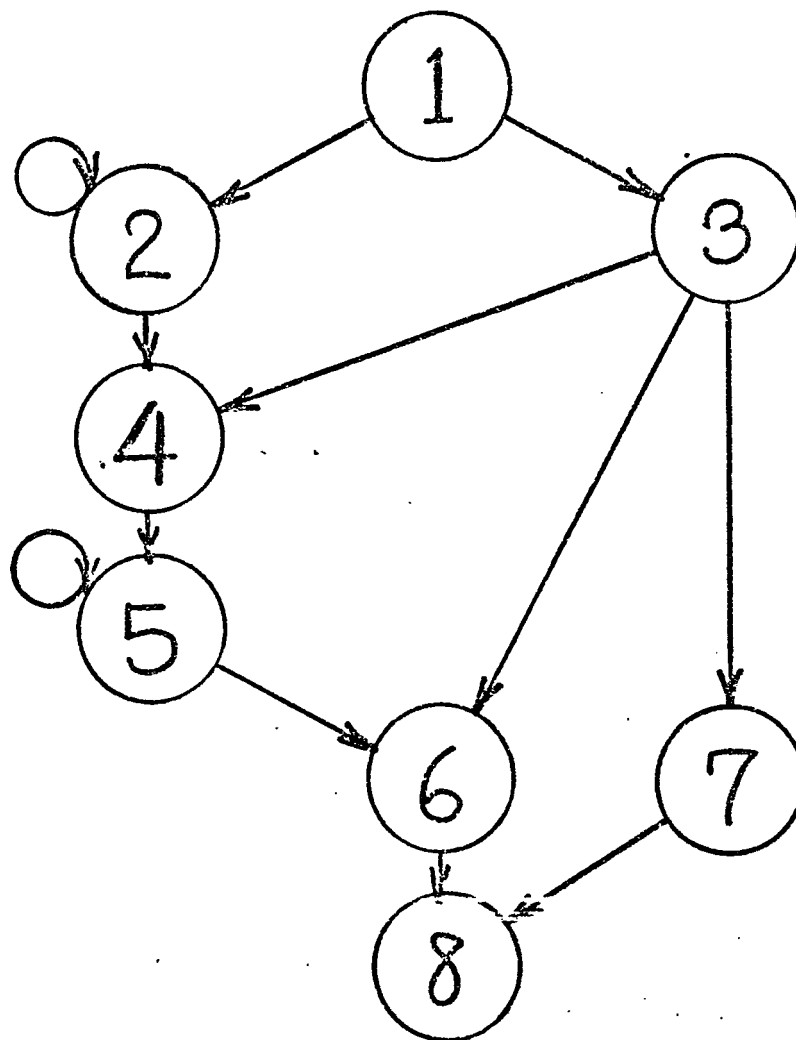


Figure 2.1 A Directed Graph

the function i , sequential control of the system function leaves node i and enters node j as the next function to execute. The input information to node j is the output from node i . Each row of the matrix in Figure 2.2 represents a possible transfer of function. For example, the third row shows that node 3 is directly connected to nodes 4, 6, and 7 by displaying the 1's in the third row. The 0's in the row show no direct connection from node 3 to the others. When a node is a branch point, indicating the directed connection to others (more than one node), which functional branch is selected next is, in the most cases, dependent on the input to the system and on the past history of the function's sequence. This property is known as reproducible system behavior;

	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	0	1	0	1	0	0	0	0
3	0	0	0	1	0	1	1	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	1	0	0
6	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0

Figure 2.2 The Connectivity Matrix of a Directed Graph

it is called a determinacy property. Dennis [12] presented an important result: if interactions between subsystems obey certain natural conditions, then determinacy of the subsystem guarantees determinacy of the whole system using Petri net [32], which is another directed graph. However, when a set of inputs is applied to the system and the order of the individual input in the set is unknown, the behavior of function sequence at a branch point at a particular moment cannot be predicted deterministically but can be predicted probabilistically, if different job type categories and the mixing ratio of them in the set are known. These probabilities of function sequence transfers at branch points can be found and be inserted in the entries of the connectivity matrix of a directed graph. Thus, the connectivity matrix becomes a probabilistic transfer table of the system's functional sequence. It is pointed out that asynchronous interruption in a computer system causes the execution of system components in a nondeterministic order during a short time period, but in an integrated long time period the system behavior is in a steady state. From the determinacy property of the system and the probability due to the unknown order of original inputs, we have a better understanding of the relationship between the nondeterministic behavior and the steady-state behavior of the system.

A set of inputs belonging to a job type category always provides same probabilistic transfer of function sequence at branch points, but a set of inputs in another job type category also provides the other probabilistic transfer table. In this situation, a computer system could be considered as a probabilistic sequential machine. As mentioned previously, outputs are produced from the nodes in the system-directed graph; therefore, most outputs of systems are functions of the states (represented by the nodes) of the probabilistic sequential machine. Specifically, the probabilistic sequential machine is a Moore type. Before building a model of the probabilistic sequential machine and drawing a graphical representation of a computer system, a real system analysis is necessary to focus attention on a particular system problem. In the analysis, test sets of input, which cover a wide range of a system's behavior, are essential for the model and the graphical representation.

In order to determine an economical and efficient test data set for diagnosis, a method is needed that will succinctly expose the complex interrelations between the flow of program control and the flow of data during that program's execution. Desirable test data is short and examines many diagnostic items simultaneously. Of special importance is the isolation of program segments in order to reach any one segment or data definition from any other segment or data definition. It is within such segments that we hope to identify the input to and the output from the computer system. A convenient method needs a simple but effective means of understanding, visualizing, and analyzing the test problem associated with the system diagnostics. The graphic representation can simplify the understanding of a large logical system's operation and can assist system analysts in handling unforeseen and unexpected problems.

Nodes in the graph represent functional elements. Directed arcs connecting two nodes represent lines of data transfer (or signal propagation)

and/or lines of function transition. In particular, the arc from node i to node j describes the functional relationship as the arc from node i enters node j like a passing of a baton. As is pointed out previously in Section 1.2, the computer system can be considered a multilevel structure, where each level can be analyzed in the same manner. Since the graphic representation is valid whether the node represents a single element, a logical building block or a complex functional module, analysis and techniques used in computer diagnosis are heavily dependent upon the level with which we are working. Therefore, to develop an efficient and economical diagnostic test set depends on the analyses and manipulations of a graphical level which describes the system's organization and performance. A higher level graphical representation could simplify the understanding of the operation of a large logical system.

A generalized approach to provide a structural graphic representation and diagnostic test sets is proposed by C. V. Kamamoorthy [54] as follows:

- (a) Structural representation of the system at appropriate level.
- (b) Partitioning or segmenting the system into a number of smaller subsystems purely from the structural description.
- (c) Strategic location of test point for purposes of subsystem segmentation, isolation, injection and/or monitoring of data during diagnostic tests.
- (d) Sequences in which test must be performed for error (fault) detection and/or location.
- (e) Determination of functional hard-core of the system.

The choice of a complex system level to consider depends primarily on the characteristics that we wish to study. Proper representation must mask out those details which are not pertinent to studying the problem at hand.

For diagnosis, we must look at the computer system from two distinct viewpoints, the structural viewpoint and the behavioral viewpoint (or function control and information flow viewpoint and the stimulus response viewpoint, respectively). The behavioral aspects like input/output relationships can be modeled by a sequential machine. When a system has a large number of responsive stimuli as input and has many variant responses as output, we become lost in a maze when considering the whole system in detail. For this reason, it is better to study the structure of the interconnections and the flow of information first and then derive valuable information before we use the behavioral description for devising the diagnostic test. If only the primary (externally controllable) inputs and observable (externally available) outputs are used for diagnosis of the computer system, the total number of test sequence will be very large and resolution of the error locating (fault) becomes low. Thus, it is desirable to break up a large system into small subsystems which reduce the length of test sequence as well as the average test time and which will also improve the resolution.

A procedure for diagnostic system treatment based on graphical representation is as follows:

- (a) A structure graph which represents a system, by flows of function control and information in the system, is needed.
- (b) Identification of externally controllable inputs and externally observable outputs of the system must be determined.
- (c) The interrelationship between the input and the output must be established as the behavioral viewpoint of the system.
- (d) When the numbers of the inputs and the outputs are very large, the large system must be broken into small subsystems.
- (e) Then, the input and output to/from a subsystem must be identified,

and the interrelationships between these subsystems should be clearly determined.

Since the computer system is represented in a multilevel structure recursively, Steps (d) and (e) may be repeated as many times as needed for test diagnosis.

2.3 Sequential Machine From a System Viewpoint

From a broad engineering viewpoint a sequential machine (and automata theory) can be considered as a branch of system theory that is concerned with the dynamic behavior of discrete parameter information systems. As such, it differs from switching theory in that its main objective is to model the macroscopic behavior of a system rather than to describe the microscopic details of the system's construction from such basic logic elements as AND gates, OR gates, and flip-flops.

System theory is based on the assumption that the external behavior of any physical device can be described by a suitable mathematical model, which identifies all of the critical features that influence the device's operation. The resulting mathematical model is called a system. Because many seemingly unrelated devices can be represented by essentially the same model, system theory provides a unified treatment of the mathematical techniques that can be used to investigate the dynamic behavior of these models.

The behavior of any system can be represented in terms of mathematical relations between three sets of variables, which describe the input, the output, and the state of the system. The input set represents those external quantities that can be applied to the system to produce a change in the system's behavior; the output set represents the possible, observable behavior of the system in response to these inputs. One of the basic characteristics

of any system is that its current output is a function not only of the present input but also of the past inputs and outputs. Because of this, we can think of a system as possessing a "memory", which stores information about the past behavior of the system. The state set, the third set of variables, is used to represent the amount of information stored by the system.

The response of a system to a given input can be represented by a set of equations that describe the functional relationships between a set of independent and a set of dependent variables. In many systems, these functions are described in terms of integral-differential equations, and of the variables taking on continuous values.

In a sequential machine (and automata theory), however, we are interested in a different class of systems. The systems are characterized by the fact that all of the variables can assume only discrete values. For example, one variable might be allowed to take only the value 0 or 1; whereas, another variable might take the value a, b , or c . Systems of this type are referred to as discrete-parameter systems.

As would be expected, the functions that describe the behavior of discrete parameter systems can no longer be represented by integral-differential equations. Fortunately, there is a branch of mathematics, referred to as abstract algebra, that provides a source of mathematical techniques that can be used to describe the functional relationships that characterize the operation of discrete-parameter systems.

The basic system model that we shall use can be thought of as a black box with a set of input and output terminals that can receive and discharge information, respectively. The black box is assumed to be constructed from storage elements and combinational logic elements. The actual details of internal structure are not available, and the only interest we have is in the resultant dynamic properties of the system that affect the way in which it

processes information.

Because of the storage elements, the present output depends on the history of the system. The following general representation is used to describe systems of this type. The input to the system is represented as a sequence of symbols $x_1, x_2, \dots, x_k, \dots$, where x_1 is the first symbol, x_2 is the second symbol, and x_k is the k th symbol. The value of a given symbol can be specified by identifying it as a particular value from a set X of all possible input symbols. For example, in a given system, X might consist of the set $\{0, 1, 2\}$. Thus, one possible input sequence might be $0, 1, 1, 0, 1, 2, 0, \dots$, where $x_1 = 0$, $x_2 = 1, \dots$, etc.

Inside the black box, there are storage elements that can remember past history of the input and response of the system. The contents of these storage elements at a given observation determine the state of the system. Because the system's input and output can take on only discrete values, the parameter representing the memory of the system can also be represented by a variable that can take on only discrete values. The state of the system at the i th observation is represented by the variable s_i , and it is assumed that s_i can take on any one of the values that belong to the set S of all possible states of the system. Because the state of the system represents the memory of the events that have previously occurred, it is possible to develop an expression that relates the value that the state variable will have at the next observation to the present value of the state variable and the value of the present input symbol. When an output is generated, its value will depend on the system's present input and state (or on only the present state of the system which will be explained later).

One representative class of discrete-parameter systems are digital networks, which are constructed from standard logic elements such as AND, OR,

NOT, or Exclusive OR circuits and storage elements such as flip-flops and delays.

Another example of discrete-parameter systems is an information transmission system, which consists of a sequence of information processing blocks; such as an information source generator, an encoder which accepts a sequence of input symbols and which generates an output sequence to transmit through a particular channel in which usually unavoidable noise is superposed and a decoder which reconstructs the original input sequence from the received output sequence of the transmission channel. Note: the decoder can make use of both present and past received symbols. Another example of discrete parameter systems is the neural network. McCulloch-Pitts "cells" or "neurons" are models for certain aspects of brain function. The models are interconnecting cells, which are a very simple, two-state, sequential machine. Once we understand perfectly each simple part and how it interacts with the others, we have a chance of understanding the network as a whole or as a system.

A final example of a discrete-parameter system is a digital computer, which provides other rich sources of sequential machine problems (and automata theory). For instance, the problems of determining the theoretical capabilities of a computer, the programming language translation, man-machine communication, etc. exist.

A deterministic sequential machine and a probabilistic sequential machine are described in detail in the following two sections.

2.4 Deterministic Sequential Machine

We are familiar the black box representation of the dynamic behavior of discrete systems and normally have no detailed description of the system's internal structures with which to work. The internal behavior is expressed

in terms of a set of possible states that the system might enter, while the number of elements in this state set provide a measure of the amount of information storage presented in the system.

The possible inputs to the system are assumed to be sequences of symbols in a finite set of input symbols, and the resulting output are sequences of symbols selected from a set of output symbols. Any black box that produces an output symbol whenever an input symbol is applied and that satisfies the above mentioned properties is called a deterministic sequential machine (DSM).

When we are studying DSM characteristics, the input set X , the output set Y , the state set S , and the relationships between these sets are of fundamental importance. Sets X and Y are external to the DSM and can be defined by direct observation. However, the internal structure of the machine is generally not available for direct observation. Thus, the set S is not easily found. The selection of a set of states for a given machine is not a unique process; it is not a serious limitation to describe the general input-output behavior of the machine rather than its actual construction. Methods to compare different state sets for a given DSM have been discussed by many researchers. A state set which has the minimum number of states is useful to describe the DSM for practical, as well as theoretical reasons. Such a description can be found in books by Hu [34], Booth [8] and Ginsburg [22].

Definition 2.1

A DSM is a five-tuple $M = (S, X, Y, \delta, \lambda)$ where

S is a nonempty set of states.

X is a finite set of input symbols.

Y is a finite set of output symbols.

δ is a mapping from $S \times X$ to S called the next state function.

λ is a mapping from $S \times X$ to Y called the output function.

A DSM in which the state set S contains only a finite number of element states is called a deterministic finite state machine. The machine thus defined is a Mealy machine. A modification of this definition, mapping λ from S to Y , is called a Moore machine.

Three techniques that have come into common usage to present the analytical properties of a DSM are the transition table, the transition directed graph, and the transition matrices. A DSM is defined when each technique properly formats the input, output, and state of the DSM.

The transition table representation of a DSM displays the properties of the next state and output mapping in tabular form. The columns of the table correspond to input symbols, and the rows correspond to states of the machine. The entry found at the intersection of the k th row and the j th column is $\delta(x_k, s_j) / \lambda(x_k, s_j)$. For a Moore machine, the output function in any row in the transition table is independent of the column, since $\lambda(s_j)$ has only one argument s_j . See Figure 2.3 for illustrations of transition tables.

A Mealy machine is equivalent in machine behavior to a Moore machine and vice versa.

$\begin{array}{c} X \\ S \end{array}$	0	1
s_1	s_3/a	s_2/b
s_2	s_1/b	s_3/c
s_3	s_3/c	s_1/a

(a) A Mealy Machine

where

$$S = \{s_1, s_2, s_3\}, X = \{0, 1\}$$

$$Y = \{a, b, c\}$$

$\begin{array}{c} X \\ S \end{array}$	0	1	$\lambda(s_i)$
s_1	s_1	s_3	a
s_2	s_3	s_1	b
s_3	s_2	s_1	a

(b) A Moore Machine

where

$$S = \{s_1, s_2, s_3\}, X = \{0, 1\}$$

$$Y = \{a, b\}$$

$$\begin{array}{ll}
 s_3 = \delta(0, s_1), s_2 = \delta(1, s_1) & s_1 = \delta(0, s_1), s_3 = \delta(1, s_1) \\
 s_1 = \delta(0, s_2), s_3 = \delta(1, s_2) & s_3 = \delta(0, s_2), s_1 = \delta(1, s_2) \\
 s_3 = \delta(0, s_3), s_1 = \delta(1, s_3) & s_2 = \delta(0, s_3), s_1 = \delta(1, s_3) \\
 \\
 a = \lambda(0, s_1), \quad b = \lambda(1, s_1) & a = \lambda(s_1) \\
 b = \lambda(0, s_2), \quad c = \lambda(1, s_2) & b = \lambda(s_2) \\
 c = \lambda(0, s_3), \quad a = \lambda(1, s_3) & a = \lambda(s_3)
 \end{array}$$

Figure 2.3 Illustrations of Transition Tables

A transition directed graph provides a graphical representation of the operation of a DSM. Each diagram consists of a set of vertices labeled to correspond to the states of the machine. For each ordered pair of states s_i and s_j (not necessarily distinct), a directed edge connects vertex s_i to s_j , if and only if there exists an input symbol $x_k \in X$ such that $\delta(x_k, s_i) = s_j$. When a directed edge connects s_i to s_j applying the input x_k to the machine, the edge is labeled as $x_k / \lambda(x_k, s_i)$. Thus, the vertices of the transition graph correspond to the present state of the machine; the label on the edge indicates the present input and the present output. The arrowhead on each edge indicates the next state of the machine. Figure 2.4 is the transition directed graph that corresponds to the transition table in Figure 2.3(a).

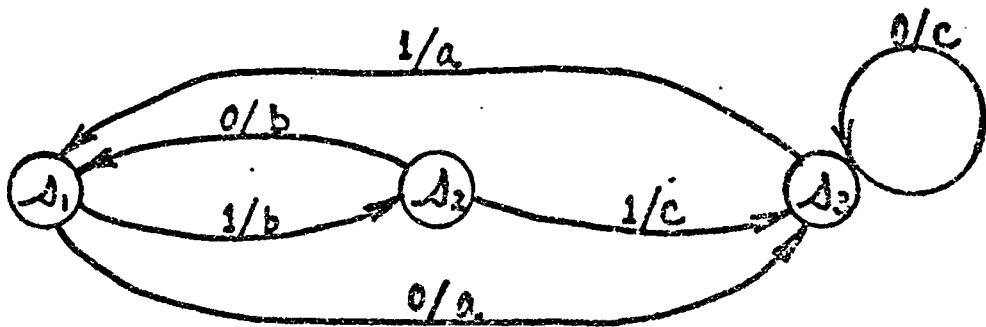


Figure 2.4 Typical Transition Directed Graph

If the machine is a Moore machine, the output mapping depends only upon the present state of the machine, and this property is used to simplify the transition directed graph. The output is usually indicated by including it in the labeling of the vertex. A transition directed graph provides easy understanding of the operation of a given DSM. In particular, abstract concepts associated with the DSM theory may become simple, visual interpretations of the DSM. However, as the number of states increases, it becomes difficult to present the transition directed graph of a machine in a compact manner. It is also difficult to use the information contained in one of these graphs to carry out computations on a digital computer. To overcome some of these problems, a transition matrix is useful.

A transition matrix, which has p rows and p columns, is an array of symbols, denoted input/present output. The rows correspond to the present state, and the columns correspond to the next state of the machine. The entry E_{ij} at the intersection of the i th row and the j th column indicates which input symbol will take the machine from state s_i to state s_j as well as which output symbol produced by the machine will correspond to this transition. Figure 2.5 is the transition matrix corresponding to the transition table in Figure 2.3(a).

$$M = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{pmatrix} --- & 1/b & 0/a \\ 0/b & --- & 1/c \\ 1/a & --- & 0/c \end{pmatrix} \end{matrix} \quad (2.1)$$

Figure 2.5 Typical Transition Matrix

Another class of machines exists in which the characteristics of the next state mapping and output mapping can be described only in a probabilistic rather

than a deterministic manner. Examples of each machines are those constructed from unreliable components or those in which internal noise sources are present. Machines of this type, which are called "probabilistic sequential machines", are described in the following section.

2.5 Probabilistic (Stochastic) Sequential Machine

We have been dealing with deterministic systems. Each sequence is well defined, and the behavior of any system is described exactly by the properties of its state transition directed graph. However, there are many systems in which these processes cannot be defined in a deterministic manner, and it is necessary to use statistical concepts to describe the system's behaviors. Techniques for analyzing the behaviors of continuous systems excited by random processes are well known. But when one attempts to analyze the behavior of the discrete parameter systems, such as found in a probabilistic sequential machine (PSM), it is conceivable that many of the standard techniques developed for the analysis of continuous systems are not directly applicable to discrete systems. Therefore, it is necessary to study some of the basic properties of the discrete random processes.

When we discussed the properties of a deterministic sequential machine, we assumed interest only in the response of the machine to determine sequences. Also, we assumed that the mappings $\delta(X \times S)$ and $\lambda(X \times S)$ uniquely defined the next state and present output of the machine. These assumptions are relaxed in that input sequences are either wholly or partially probabilistic and in that the mappings δ and λ are defined in a probabilistic manner due to a random property of the input sequences.

We start with an example of Markov chains in order to describe a PSM. Consider a physical system in motion. Let the various positions (states) of

the system be denoted by $0, 1, 2, 3, \dots$, and let $p_{ij}^{(r)}$ be the probability of the system's transition from the state i at time m to the state j at time $m+r$ (for any m); this is called an r -step transition probability. The matrix $p = (p_{ij}^{(0)})$ is called a stochastic matrix of the system. A sequence of various states of the system, $\{s_k\}$ for $k = 0, 1, 2, \dots$, is called a Markov chain if, for every finite collection of integers $\ell_1 < \ell_2 < \dots < \ell_r < \ell$, where ℓ is the length of the sequence. In other words, after ℓ moments of a unit of time, we have

$$P_r\{s_\ell | s_{\ell-r}, s_{\ell-r+1}, \dots, s_{\ell-1}\} = P_r\{s_\ell | s_{\ell-1}\}. \quad (2.2)$$

Let s_ℓ and $s_{\ell-1}$ be the i th and the j th states, respectively, then

$$P_r\{s_\ell | s_{\ell-1}\} = P_{ij}(\ell) \quad \text{for } i, j = 0, 1, 2, \dots \quad (2.3)$$

If $P_{ij}(\ell)$ is independent of ℓ for i, j , we say that the chain is homogeneous.

For a homogeneous Markov chain, let us denote

$$P_r\{s_\ell = j | s_{\ell-1} = i\} = p_{ij} \quad (2.4)$$

and quite generally,

$$P_r\{s_{\ell+r} = j | s_\ell = i\} = p_{ij}^{(r)} \quad \text{for } r = 1, 2, \dots \quad (2.5)$$

The probabilities (2.5) can be expressed in terms of (2.4), as follows:

$$\begin{aligned} p_{ij}^{(r)} &= \sum P_r\{s_{\ell+1} = i_1 | s_\ell = i\} P_r\{s_{\ell+2} = i_2 | s_{\ell+1} = i_1\} \\ &\quad \cdot \dots P_r\{s_{\ell+r} = j | s_{\ell+r-1} = i_{n-1}\} \\ &= \sum p_{ii_1} p_{i_1 i_2} \dots p_{i_{n-1} j} = p_{i_{n-1} j}, \end{aligned} \quad (2.6)$$

for the sum on i_1, i_2, \dots, i_{n-1} .

we have, from properties of probability,

$$p_{ij}^{(r)} \geq 0 \quad \text{and} \quad \sum_j p_{ij}^{(r)} = 1 \quad \text{for } r = 1, 2, \dots \quad (2.7)$$

Suppose the initial state of the system is the i th state, the probability of the system being in the j th state is p_{ij} after one unit of time where $j = 0, 1, 2, \dots, i, \dots$, in a matrix form,

$$\begin{matrix} \text{the } i\text{th} \\ (0, 0, \dots, 1, 0, \dots) \end{matrix} \begin{pmatrix} p_{00} & p_{01} & p_{02} & p_{03} & \dots \\ p_{10} & p_{11} & p_{12} & p_{13} & \dots \\ p_{20} & p_{21} & p_{22} & p_{23} & \dots \\ \vdots & \dots & \dots & \dots & \dots \end{pmatrix} = (p_{i0}, p_{i1}, \dots, p_{ii}, \dots). \quad (2.8)$$

When the number of states is finite, the system becomes a finite Markov chain. Feller [19] applied his theory of recurrent events to Markov chains and developed a simple and more elegant theory, which made no distinction between finite and infinite chains. We shall limit ourselves to a finite state system. Relaxing the DSM assumptions mentioned previously (namely the introduction of random inputs to discrete parameter systems), the next input symbol is uncertain. Thus, the appearance of a specific symbol $x_i \in X$ is described with a probability p_i , where X is a finite set of input symbols. Using the example in Figure 2.5, let p_0 and p_1 be a set α of the appearance probabilities of input symbols 0 and 1, respectively. The state transition becomes

$$M(\alpha) = \begin{pmatrix} 0 & p_1 & p_0 \\ p_0 & 0 & p_1 \\ p_1 & 0 & p_0 \end{pmatrix} = p_0 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + p_1 \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (2.9)$$

If we introduce the other set denoted by a β of probabilities p_0^i and p_1^i , the

transition matrix is

$$M(\beta) = \begin{pmatrix} 0 & p_1' & p_0' \\ p_0' & 0 & p_1' \\ p_1' & 0 & p_0' \end{pmatrix} = p_0' \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + p_1' \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

The probabilities of the output symbols are

$$M(a/\alpha) = \begin{pmatrix} 0 & 0 & p_0 \\ 0 & 0 & 0 \\ p_1 & 0 & 0 \end{pmatrix}, M(b/\alpha) = \begin{pmatrix} 0 & p_1 & 0 \\ p_0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, M(c/\alpha) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & p_1 \\ 0 & 0 & p_2 \end{pmatrix} \quad (2.10)$$

$$M(a/\beta) = \begin{pmatrix} 0 & 0 & p_0' \\ 0 & 0 & 0 \\ p_1' & 0 & 0 \end{pmatrix}, M(b/\beta) = \begin{pmatrix} 0 & p_1' & 0 \\ p_0' & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ and } M(c/\beta) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & p_1' \\ 0 & 0 & p_2' \end{pmatrix}.$$

Example:

Consider a DSM $(S, X, Y, \delta, \lambda)$ where

$$S = \{s_1, s_2\}, X = \{0, 1, 2, 3\}, Y = \{a, b\}$$

M and λ in matrix form augmented by the input symbol are

$$M(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, M(1) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, M(2) = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \text{ and } M(3) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

$$\lambda(0) = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}, \lambda(1) = \begin{pmatrix} 0 & b \\ a & 0 \end{pmatrix}, \lambda(2) = \begin{pmatrix} b & 0 \\ b & 0 \end{pmatrix} \text{ and } \lambda(3) = \begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix}.$$

Two sets of probabilities of random input are introduced

$$p_\alpha = (\frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}) \text{ and } p_\beta = (\frac{1}{6}, 0, \frac{1}{2}, \frac{1}{3}).$$

The constructed PSM $= (S, X, Y, \{M(x)\}, \Lambda)$, where $x \in X$ is as follows:

$$S = \{s_1, s_2\},$$

$$X = \{\alpha, \beta\},$$

$$Y = \{a, b\},$$

$$M(\alpha) = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + 0 \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} + \frac{1}{4} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

$$M(\beta) = \frac{1}{6} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + 0 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} + \frac{1}{3} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

then

$$M(\alpha) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{4} & \frac{3}{4} \end{pmatrix} \quad \text{and} \quad M(\beta) = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

$$\Lambda(a/\alpha) = \begin{pmatrix} \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{pmatrix}, \quad \Lambda(b/\alpha) = \begin{pmatrix} 0 & \frac{1}{4} \\ 0 & \frac{1}{4} \end{pmatrix},$$

$$\Lambda(a/\beta) = \begin{pmatrix} \frac{1}{6} & \frac{1}{3} \\ 0 & \frac{1}{6} \end{pmatrix} \quad \text{and} \quad \Lambda(b/\beta) = \begin{pmatrix} \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{3} \end{pmatrix}.$$

We shall define a probabilistic sequential machine formally as follows.

Definition 2.2

A Mealy type probabilistic sequential machine is a quadruple $PSM = \langle S, X, Y, \{M(y/x)\} \rangle$ where

S is a nonempty set of states.

X is a finite set of input symbols.

Y is a finite set of output symbols.

$\{M(y/x)\}$ is a finite set containing $|X||Y|$ square matrices of order $|S|$ such that $m_{ij}(y/x) \geq 0$ for all i and j , and

$$\sum_{y \in Y} \sum_{j=1}^{|S|} m_{ij}(y|x) = 1 \quad \text{where } M(y|x) = [m_{ij}(y|x)],$$

where the symbol $|S|$ denotes "the number of elements" in the set S .

Definition 2.3

A Moore type probabilistic sequential machine is a five-tuple $PSM = (S, X, Y, \{M(x)\}, \Lambda)$ where

S is a nonempty set of states.

X is a finite set of input symbols.

Y is a finite set of output symbols.

$\{M(x)\}$ is a finite set containing $|X|$ square stochastic matrices of order $|S|$.

Λ is a deterministic function from S into Y .

From the preceding discussion, we can say that the DSM is a special case of the PSM. In other words, the PSM is a generalized machine of the DSM, in the sense of machine behavior.

The PSM can be generalized by imposing random inputs and by introducing a transfer matrix, state versus item of output. This generalized machine is described in the next section.

2.6 Definition of a Generalized Probabilistic Sequential Machine

Definition 2.4

A generalized probabilistic sequential machine (GPSM) is a six-tuple $GPSM = (S, X, Y, \{M(x)\}, \phi, T)$, where

S is a nonempty set of states.

X is a finite set of input symbols.

Y is a finite set of output symbols.

$\{M(x)\}$ is a finite set containing $|X|$ square stochastic matrices of order $|S|$.

ϕ is a distribution of input symbols.

T is a transfer matrix of states versus output symbols.

A PSM imposed with random inputs is just another PSM. Without loss of generality, the two symbol-two state PSM expounds this property as follows:

$$M(0) = \begin{pmatrix} p_{11}^0 & p_{12}^0 \\ p_{21}^0 & p_{22}^0 \end{pmatrix} \quad \text{and} \quad M(1) = \begin{pmatrix} p_{11}^1 & p_{12}^1 \\ p_{21}^1 & p_{22}^1 \end{pmatrix}$$

where $\sum_j p_{ij}^k = 1$, $p_{ij}^k \geq 0$ for all i and k .

Suppose a specified random of inputs are given by $\pi = (\pi_0, \pi_1)$ where $\pi_0 + \pi_1 = 1$ and $\pi_0, \pi_1 \geq 0$. Thus

$$M(\pi) = \pi_0 \begin{pmatrix} p_{11}^0 & p_{12}^0 \\ p_{21}^0 & p_{22}^0 \end{pmatrix} + \pi_1 \begin{pmatrix} p_{11}^1 & p_{12}^1 \\ p_{21}^1 & p_{22}^1 \end{pmatrix}$$

where $\sum_k \sum_j \pi_k p_{ij}^k = 1$ and $\pi_k p_{ij}^k \geq 0$ for $k = 0, 1$ and $i, j = 1, 2$.

However, when the imposed randomness or a distribution of input probability is unknown in machine operation and when we need to know the distribution by some approximation, a model using this generalized machine is helpful. A large-logical system described in Chapters V and VI is modeled by GPSM.

2.7 A Model Used Generalized Probabilistic Sequential Machine

We now focus our attention on the central problem of system modeling. A system model expresses in some form the relationships which exist between

the basic functions of the system. The system model may be very simple, or it may be quite complex, depending on which level of the system we are interested in and how the model will be used.

It has been pointed out that a Moore machine of a PSM can represent a computer system. The stochastic nature of the PSM really reflects a basic property of computer system performance (Section 2.1). Although the exact sequence and characteristics of jobs are seldom known, the probability distribution of the job type and its resource demands can be predicted. This stochastic property is basic to the operation of a computer system. Before representing a computer system model, a description of a general contemporary computer system is needed. A computer system is managed by its operating system. Generally, an operating system has three main parts: job management or setting up the environment for the user's job execution, task handling or system resource management and data or input/output management. Operating systems can be divided into any number of parts associated with computer system levels as mentioned previously.

Job management or the total environment of a computer system is handled by the executive function. The executive arranges for the operator and for the user-programmer to communicate with the system either through operator commands or job control cards presented to run the program. The operator must be able to communicate with the operating system in order to tell the operating system what resources must be made available for his program.

Task management or allocation of system resources concerned with the program (once the program is executed) is directed by this portion of the operating system. Typically these system resources include main storage, central processing unit (CPU) time, input/output operations, and a system clock which arranges the programs in main storage in their proper place and takes

programs from secondary storage memory and brings them into an execution mode. Data or input/output (I/O) management is that portion of the operating system which controls the utilization of space on mass storage for direct access, controls the allocation of tape drive if it is a tape data set, or controls the reading in of cards and the output of printed matter. In short, all I/O operations must be scheduled and executed by the operating system, not the user's program. A typical computer system is illustrated in Figure 2.6. In the figure, the coarse scheduler, dispatcher, and facility inventory are performing the job management; the dynamic allocator is handling the task management; the symbiont complex, C/SP (Commun. Symbiont Processor), I/O, and communication handler are directing the data management performance.

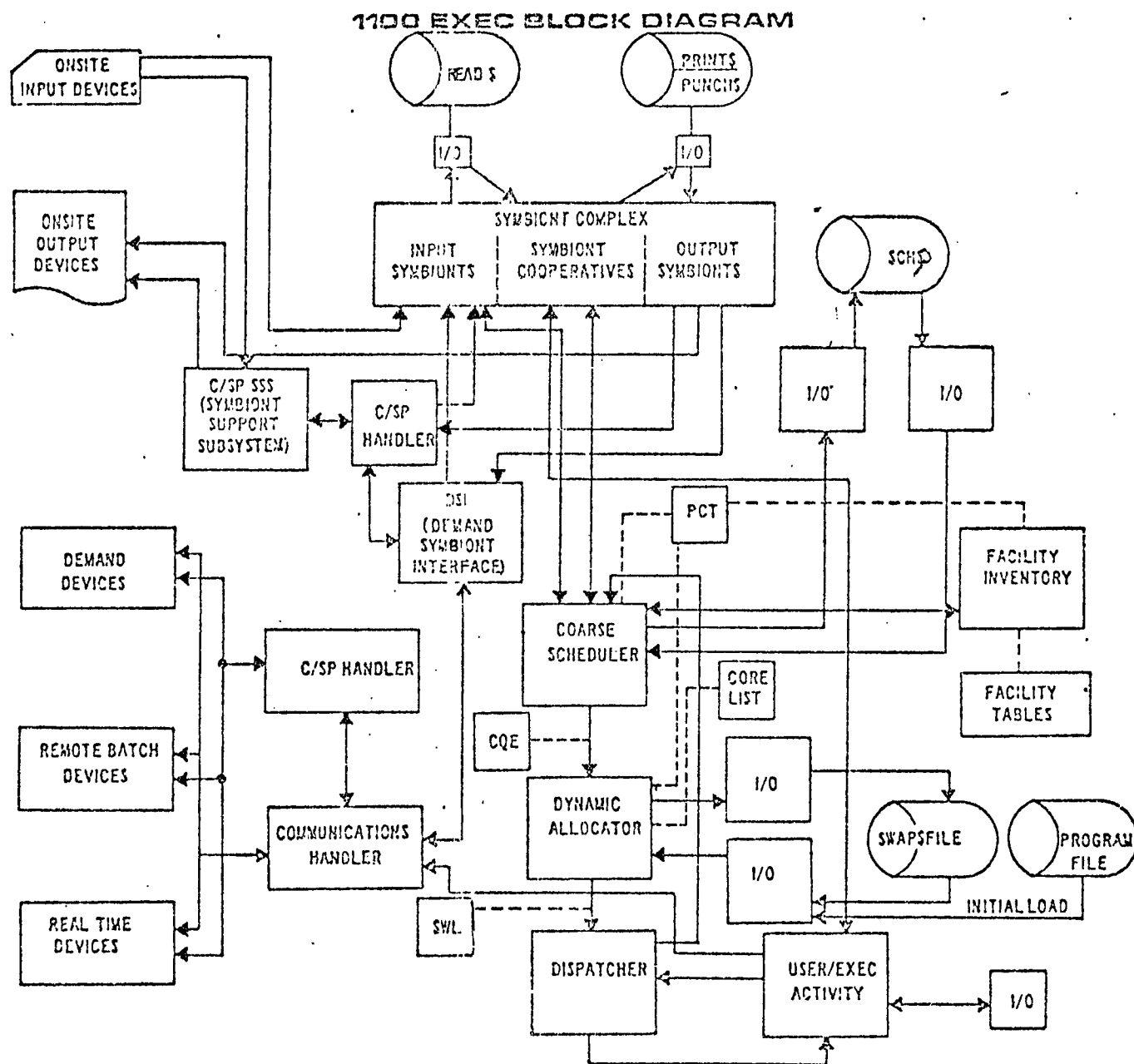
The following abbreviations are used in Figure 2.6:

PCT (Program Control Table),
 CQE (Core Request Queue Entry),
 SWL (Event Monitoring Switch List),
 SCHQ (Schedule Queue),
 READ\$ (Card Images READ from Card Reader), and
 PRINT\$/PUNCH\$ (Card Images to PRINT/PUNCH to line printer/card punch).

We classify computer job types into categories such as scientific computation, business computation, conversation mode, simulation, and data management according to their characteristics. Each category may be considered as an input symbol to a (Moore-type) PSM used to model the system.

The next step of the modeling is to determine the structure level of the system which shall be used. Then the connectivity matrix of functional elements in the selected structure level should be built from a directed graph representing the structure of the system. The probabilities of function trans-

Figure 2.6 A Computer System



fer (state transfer matrices) for each input symbol (for each job category) at branch points of the directed graph should be evaluated. Generally, the output items of the computer system are activities of output data channels, for example, swap file channel, program file channel, mass storage file channel, magnetic tape channel, line printer/card puncher channel, remote communication channel, etc.. Therefore, a transfer matrix from states (or the functional element) of the PSM to these output items is needed. This is a generalized probabilistic sequential machine (GPSM defined in Section 2.6), if we assume there is a input load distribution of the classified input category to the system.

For illustrating a computer system, a large logical system is

$$\text{GPSM} = (S, X, Y, \{M(x)\}, \phi, T)$$

where

$S = \{\text{Symbiont complex, I/O control, coarse scheduler, dynamic allocator, dispatcher, user/exec. activity}\},$

$X = \{\text{Scientific computation, business computation, conversation mode, simulation, data management}\},$

$Y = \{\text{Swap file channel, program file channel, mass storage file channel, magnetic tape channel, lineprint/cardpuncher channel, remote communication channel}\},$

$M(x)$, for $x \in X$ is the state transition probabilistic matrices associated with each input symbol,

ϕ is a load distribution of input category,

T is a transfer matrix of states versus output symbols (or output items).

There are two cases where the GPSM is useful:

- (1) When the input load distribution is known, the output items could be computed by the model. Testing/debugging and benchmarking of the system belong in this case because the input jobs are canned or known.
- (2) Input load distribution, which is unknown and which produces the system outputs could be computed with the model, when statistics of the computer system output performance (for example system failure) are available. This case is useful to determine internal past activities of the system, which might be applicable for performance analysis and failure detection of subsystems as well as the system.

A number of problems always arise whenever we attempt to model a system. The most significant problem is that of the validity of the model. A model of a system is an abstraction of the system in which many details of the system's structure have been omitted. The model is basically a simplified version of the system. In the process of deriving the model from the system, some significant relations may be dropped or some important constants may be estimated incorrectly. If this happens, the model is not valid, that is, the behavior of the model for a given input will not match the behavior of the real system within reasonable limits. An invalid model is relatively useless. The problem of model validity is probably the most difficult and certainly the most serious problem in modeling, especially for modeling of a large system.

The most difficult task in our modeling is the evaluation of the state transition probabilistic matrices and the transfer matrix of states versus output items. A rough evaluation of these matrices could be done by studying the internal logical structure of the operating system and estimated output

channel activities. A better way to evaluate these matrices is system monitoring using benchmark programs with an item of a job type category. Several monitoring methods in hardware and in software [1], [70] are available.

CHAPTER III

TWO-STATE INPUT-TRACEABLE
PROBABILISTIC SEQUENTIAL MACHINES3. Introduction

This chapter provides the mathematical foundations of the theoretical approach to error detection in a computer system modeled by a generalized probabilistic sequential machine (PSM). The foundations include a closed form representation of the product of any infinite or finite number of two-state stochastic matrices and the input traceability property of a two-state isolated PSM.

A review of the class of two-state, two-symbol, completely isolated PSM [79] and the class of two-state, multisymbol, completely isolated PSM [68] is presented in Section 3.2. Following it, a broader class of isolated machines called the absolutely isolated machine, which includes the completely isolated machine as a subclass, is introduced. It is demonstrated that the condition for isolation given by Yasui and Yajima [79] and Tan [68] is unnecessarily conservative. New conditions for determining the isolated property of a PSM are presented. It is shown that an isolated machine satisfying the new conditions may not satisfy the condition proposed by Yasui and Yajima and Tan. Section 3.5 introduces approximate probabilities and errors of approximation. It is shown that any PSM isolated by some k th approximation accepts a $(k+1)$ definite language. An algorithm is derived to synthesize an absolutely isolated probabilistic automata with any level of approximation.

The last section deals with the past input traceability property of an isolated machine, which can be applicable to error detection.

3.1 Completely Isolated Machines

First consider a two-state/two-symbol PSM. Let x_1 and x_2 be the two symbols and let

$$M(x_1) = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} \quad \text{and} \quad M(x_2) = \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix}$$

be the two-state transition (stochastic) matrices, where $0 \leq a, b, c$ and $d \leq 1$. Let the fundamental matrix $M_i(x_i)$ for $i=1,2$ be defined by

$$M_1(x_1) = \begin{pmatrix} b/(a+b) & a/(a+b) \\ b/(a+b) & a/(a+b) \end{pmatrix},$$

$$M_2(x_1) = \begin{pmatrix} a/(a+b) & -a/(a+b) \\ -b/(a+b) & b/(a+b) \end{pmatrix}$$

and

$$M_1(x_1) = X_1 Y_1 \quad \text{and} \quad M_2(x_1) = X_2 Y_2$$

where X_i and Y_i are the i^{th} characteristic column vector and the i^{th} characteristic row vector of $M(x_1)$ and $M(x_2)$, respectively.

Then

$$M(x_1) = 1 \cdot M_1(x_1) + \lambda M_2(x_1)$$

where $\lambda = 1-a-b$ is the eigenvalue of $M(x_1)$. The other eigenvalue of the matrix is 1. Similarly,

$$M(x_2) = 1 \cdot M_1(x_2) + \mu M_2(x_2)$$

where

$$M_1(x_2) = \begin{pmatrix} d/(c+d) & c/(c+d) \\ d/(c+d) & c/(c+d) \end{pmatrix}, \quad M_2(x_2) = \begin{pmatrix} c/(c+d) & -c/(c+d) \\ -d/(c+d) & d/(c+d) \end{pmatrix}$$

and $\mu = 1-c-d$.

A product of stochastic matrices $M(x_1)$ and $M(x_2)$ is

$$\begin{aligned} M(x_1)M(x_2) &= (M_1(x_1) + \lambda M_2(x_1))(M_1(x_2) + \mu M_2(x_2)) \\ &= M_1(x_1)M_1(x_2) + \mu M_1(x_1)M_2(x_2) + \lambda M_2(x_1)M_1(x_2) \\ &\quad + \lambda\mu M_2(x_1)M_2(x_2) \end{aligned}$$

Using the multiplication table of the two stochastic matrices $M(x_1)$ and $M(x_2)$ given by Yasui and Yajima [79], we obtain

$$M(x_1)M(x_2) = M_1(x_2) + \mu H + \lambda\mu M_2(x_1)$$

where

$$H = M_1(x_1)M_2(x_2) = \frac{bc-ad}{(a+b)(c+d)} \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$$

For simplicity, let $A = M(x_1)$ and $B = M(x_2)$. The multiplication table of the two stochastic matrices, $A = A_1 + \lambda_a A_2$ and $B = B_1 + \lambda_b B_2$ is as follows:

	A_1	A_2	B_1	B_2	H
A_1	A_1	0	B_1	H	H
A_2	0	A_2	0	A_2	0
B_1	A_1	$-H$	B_1	0	H
B_2	0	B_2	0	B_2	0
H	0	H	0	H	0

where A_i and B_i are the i^{th} fundamental matrices, for $i=1,2$ and λ_a and λ_b are

the λ and μ eigenvalues of A and B , respectively.

In general, $M(x)$, $x \in X^{*+}$ can be expanded as follows:

$$\begin{aligned} M(x) &= M(\sigma_1)M(\sigma_2)M(\sigma_3) \dots M(\sigma_m) \\ &= M_1(\sigma_m) + \lambda_m M_1(\sigma_{m-1})M_2(\sigma_m) + \lambda_m \lambda_{m-1} M_1(\sigma_{m-2})M_2(\sigma_{m-1}) \\ &\quad + \dots + \lambda_m \lambda_{m-1} \dots \lambda_1 M_2(\sigma_1) \end{aligned} \quad (3.1)$$

where σ_i is x_1 or x_2 , and λ_i is λ or μ depending on x_1 or x_2 for $i=1,2,\dots,m$ and $x = \sigma_1\sigma_2\sigma_3\dots\sigma_m$.

Definition of Isolated Machine by Yasui and Yajima:

The $(1,2)$ element of (3.1) is called the m^{th} approximation probability denoted as $p^{(m)}(x)$ after processing input x^* . Yasui and Yajima defined a set of two stochastic matrices as being completely isolated by the m^{th} approximation if and only if every pair of input strings of length $m+1$ (which is produced from the set of matrices) is isolated by the m^{th} approximation.

When the length m input string is truncated by the last $k+1$ symbols, the truncation error of the product (3.1) is bounded by a constant

$$\epsilon_{\xi}^{(k)} < \frac{\beta^{k+1}}{1-\beta} \max_{i,j} \{ ||H_{ij}||, ||M_2(x_i)|| \}^{(\xi)}$$

where $\beta = \max(|\lambda|, |\mu|)$, $H_{ij} = M_1(x_i)M_2(x_j)$. The norm $||A||^{(\xi)}$ is defined by the absolute value of the $(\xi,2)$ element of the matrix A , where $\xi=1,2$. If $||A||^{(1)} = ||A||^{(2)}$, then $||A||$ is interpreted as $||A||^{(\xi)}$.

The preceding properties of two-state/two-symbol PSM have been extended to the two-state multisymbol case by Tan.

[†]The symbol X^* denotes the set of all possible input strings that can be formed by the elements of X .

In this chapter, the properties of a two-state multisymbol PSM have been extended further. In particular, a much tighter upper bound than the previous equation for the two-state multisymbol PSM is presented. It is derived by considering the sign of the eigenvalue of each stochastic matrix rather than just the absolute value. It is shown that a PSM which is not isolated by Yasui and Yajima and by Tan's definitions may actually be isolated. Consider the following PSM

$$M(x_1) = \begin{pmatrix} 0.2 & 0.8 \\ 0.1 & 0.9 \end{pmatrix} \quad \text{and} \quad M(x_2) = \begin{pmatrix} 0.5 & 0.5 \\ 0.9 & 0.1 \end{pmatrix}$$

and evaluate Yasui and Yajima's criterion ($h \leq 1$) of the PSM (details of the evaluation are presented in the section 3.5). The value h of the PSM is $2.1 > 1$ so that it is not an isolated machine in their sense. However, the PSM is isolated in our sense. Our discussion begins in the next section.

3.2 The Product of 2x2 Stochastic Matrices

In the PSM $(S, X, Y, \{M(x_i)\}, \Lambda)$, the parameters are defined as follows:

S is a set of states, $\{s_1, s_2\}$.

X is a set of input symbols, $\{x_1, x_2, \dots, x_n\}$.

Y is a set of output symbols, $\{y_1, y_2\}$.

$\{M(x_i)\}$ is a set of state transition stochastic matrices.

Λ is a deterministic function from S into Y .

The PSM is obviously characterized by the transition matrices $M(x_i)$ ($i=1, 2, \dots, n$). For simplicity, we shall denote $M(x_i)$ by $M^{(i)}$. The behavior of the PSM due to an input string $x = \sigma_1 \sigma_2 \dots \sigma_m$, can be determined by the matrix product $M_1 M_2 \dots M_m$, $M_j \in \{M^{(i)}\}$ and $\sigma_j \in X$ for $j = 1, 2, \dots, m$. A closed form of matrix

expansion is presented by considering element by element multiplication. First we prove the following lemmas.

Lemma 3.1:

The eigenvalues of a 2×2 stochastic matrix

$$M(x_i) \triangleq M^{(i)} = \begin{pmatrix} 1-\alpha_i & \alpha_i \\ \beta_i & 1-\beta_i \end{pmatrix}$$

are 1 and $1-\alpha_i-\beta_i$, where $1 \geq \alpha_i, \beta_i \geq 0$.

Proof:

The eigenvalues of $M^{(i)}$ are obtained by solving for λ from the equation $|\lambda I - M^{(i)}| = 0$, i.e.,

$$\begin{vmatrix} \lambda - 1 + \alpha_i & -\alpha_i \\ -\beta_i & \lambda - 1 + \beta_i \end{vmatrix} = \lambda^2 - (2 - \alpha_i - \beta_i)\lambda + (1 - \alpha_i - \beta_i) = 0$$

Thus,

$$\lambda = \frac{2 - \alpha_i - \beta_i \pm (\alpha_i + \beta_i)}{2} = 1 \quad \text{or} \quad 1 - \alpha_i - \beta_i$$

Denote the second eigenvalue λ of $M^{(i)}$ by λ_i and consider a multiplication of the two matrices:

$$\begin{pmatrix} 1-a_{m-1} & a_{m-1} \\ b_{m-1} & 1-b_{m-1} \end{pmatrix} \begin{pmatrix} 1-a_m & a_m \\ b_m & 1-b_m \end{pmatrix} = \begin{pmatrix} 1-a'_{m-1} & a'_{m-1} \\ b'_{m-1} & 1-b'_{m-1} \end{pmatrix} \quad (3.2)$$

where

$$a_{m-1}^i = a_m + (1 - a_m - b_m) a_{m-1} = a_m + \lambda_m a_{m-1}$$

$$b_{m-1}^i = b_m + (1 - a_m - b_m) b_{m-1} = b_m + \lambda_m b_{m-1}.$$

The eigenvalue of the matrix product is

$$\lambda_{m-1}^i = 1 - a_{m-1}^i - b_{m-1}^i = \lambda_m \lambda_{m-1}.$$

Similarly, the product of three matrices may be presented as

$$\begin{aligned} & \begin{pmatrix} 1 - a_{m-2} & a_{m-2} \\ b_{m-2} & 1 - b_{m-2} \end{pmatrix} \begin{pmatrix} 1 - a_{m-1} & a_{m-1} \\ b_{m-1} & 1 - b_{m-1} \end{pmatrix} \begin{pmatrix} 1 - a_m & a_m \\ b_m & 1 - b_m \end{pmatrix} \\ &= \begin{pmatrix} 1 - a_{m-2} & a_{m-2} \\ b_{m-2} & 1 - b_{m-2} \end{pmatrix} \begin{pmatrix} 1 - a_{m-1}^i & a_{m-1}^i \\ b_{m-1}^i & 1 - b_{m-1}^i \end{pmatrix} = \begin{pmatrix} 1 - a_{m-2}^i & a_{m-2}^i \\ b_{m-2}^i & 1 - b_{m-2}^i \end{pmatrix} \end{aligned}$$

where

$$a_{m-2}^i = a_m + \lambda_m a_{m-1} + \lambda_m \lambda_{m-1} a_{m-2},$$

$$b_{m-2}^i = b_m + \lambda_m b_{m-1} + \lambda_m \lambda_{m-1} b_{m-2}.$$

The eigenvalue of the product is

$$\lambda_{m-2}^i = (1 - a_{m-2}^i - b_{m-2}^i) = \lambda_m \lambda_{m-1} \lambda_{m-2}.$$

Lemma 3.2:

Let

$$a_{m-i}^i = a_m + \lambda_m a_{m-1} + \lambda_m \lambda_{m-1} a_{m-2} + \dots + \prod_{j=0}^{i-1} \lambda_{m-j} a_{m-i}$$

$$b_{m-i}^i = b_m + \lambda_m b_{m-1} + \lambda_m \lambda_{m-1} b_{m-2} + \dots + \prod_{j=0}^{i-1} \lambda_{m-j} b_{m-i}$$

Then

$$1 - a_{m-i}^i - b_{m-i}^i = \lambda_m \lambda_{m-1} \lambda_{m-2} \dots \lambda_{m-i} = \lambda_{m-i}^i \quad (3.3)$$

Proof:

By substituting a_{m-i}^i and b_{m-i}^i into (3.3),

$$\begin{aligned} & 1 - a_m^i - b_m^i - \lambda_m (a_{m-1}^i + b_{m-1}^i + \lambda_{m-1} (a_{m-2}^i + b_{m-2}^i \\ & + \lambda_{m-2} (a_{m-3}^i + b_{m-3}^i + \dots + \lambda_{m-i+1} (a_{m-i}^i + b_{m-i}^i) \dots)) \\ & = \lambda_m (1 - a_{m-1}^i - b_{m-1}^i - \lambda_{m-1} (a_{m-2}^i + b_{m-2}^i + \lambda_{m-2} (a_{m-3}^i \\ & + b_{m-3}^i + \dots + \lambda_{m-i+1} (a_{m-i}^i + b_{m-i}^i) \dots)) \\ & = \lambda_m \lambda_{m-1} (1 - a_{m-2}^i - b_{m-2}^i + \lambda_{m-2} (a_{m-3}^i + b_{m-3}^i + \dots \\ & + \lambda_{m-i+1} (a_{m-i}^i + b_{m-i}^i) \dots)) \\ & = \lambda_m \lambda_{m-1} \lambda_{m-2} (1 - a_{m-3}^i - b_{m-3}^i + \dots + \lambda_{m-i+1} (a_{m-i}^i + b_{m-i}^i) \dots) \\ & = \lambda_m \lambda_{m-1} \lambda_{m-2} \lambda_{m-3} \dots \lambda_{m-i+1} (1 - a_{m-i}^i - b_{m-i}^i) \\ & = \lambda_m \lambda_{m-1} \lambda_{m-2} \lambda_{m-3} \dots \lambda_{m-i+1} \lambda_{m-i} \end{aligned}$$

Theorem 3.1:

The product of 2×2 stochastic matrices, $M_1 M_2 M_3 \dots M_m$, can be represented as follows:

$$M_1 M_2 M_3 \dots M_m = \begin{pmatrix} 1 - a_1^i & a_1^i \\ b_1^i & 1 - b_1^i \end{pmatrix}$$

where

$$M_i = \begin{pmatrix} 1 - a_i & a_i \\ b_i & 1 - b_i \end{pmatrix} \quad \text{for } i=1, 2, 3, \dots, m$$

and

$$a_1' = a_m + \lambda_m a_{m-1} + \lambda_m \lambda_{m-1} a_{m-2} + \dots + \prod_{j=0}^{m-2} \lambda_{m-j} a_1 \quad (3.4)$$

$$b_1' = b_m + \lambda_m b_{m-1} + \lambda_m \lambda_{m-1} b_{m-2} + \dots + \prod_{j=0}^{m-2} \lambda_{m-j} b_1 \quad (3.5)$$

Proof:

We shall prove this theorem using the mathematical induction method.

Let k be the upper limit of Π in the above equations. For $k=1$, we have (3.2)

which has been proven to be true.

For $k = i-1$, we have

$$a_{m-i+1}' = a_m + \lambda_m a_{m-1} + \dots + \prod_{j=0}^{i-1} \lambda_{m-j} a_{m-i+1} \quad (3.6)$$

$$b_{m-i+1}' = b_m + \lambda_m b_{m-1} + \dots + \prod_{j=0}^{i-1} \lambda_{m-j} b_{m-i+1} \quad (3.7)$$

which we assume to be true. Now we want to prove that $k=i$ is also true.

For $k=i$,

$$\begin{pmatrix} 1-a_{m-i}' & a_{m-i}' \\ b_{m-i}' & 1-b_{m-i}' \end{pmatrix} \begin{pmatrix} 1-a_{m-i+1}' & a_{m-i+1}' \\ b_{m-i+1}' & 1-b_{m-i+1}' \end{pmatrix} = \begin{pmatrix} 1-a_{m-i}' & a_{m-i}' \\ b_{m-i}' & 1-b_{m-i}' \end{pmatrix}.$$

The (1,2) element of the left-hand side of the equation is computed first,

$$\begin{aligned} & (1-a_{m-i}')a_{m-i+1}' + a_{m-i}'(1-b_{m-i+1}') \\ &= a_{m-i+1}' + a_{m-i}'(1-a_{m-i+1}' - b_{m-i+1}') \\ &= a_{m-i+1}' + \lambda_{m-i+1}' a_{m-i}'. \end{aligned}$$

By substituting the equations (3.3), (3.6), and (3.7) into the above equation, we obtain

$$a'_{m-i} = a'_m + \lambda'_m a'_{m-1} + \dots + \prod_{j=0}^{i-1} \lambda'_{m-j} a'_{m-i+1} + \prod_{j=0}^i \lambda'_{m-j} a'_{m-i}.$$

Similarly, we can compute b'_{m-i} which is

$$b'_{m-i} = b'_m + \lambda'_m b'_{m-1} + \dots + \prod_{j=0}^{i-1} \lambda'_{m-j} b'_{m-i+1} + \prod_{j=0}^i \lambda'_{m-j} b'_{m-i}.$$

This completes the proof. ■

Corollary 3.2:

The eigenvalues of the product of m 2×2 stochastic matrices, $M_1 M_2 \dots M_m$, are 1 and $\lambda'_m \lambda'_{m-1} \dots \lambda'_2 \lambda'_1$, where $\lambda'_i = 1 - a'_i - b'_i$ is an eigenvalue of the matrix M_i in the product when $i = 1, 2, \dots, m$.

Proof:

By Theorem 3.1:

$$M_1 M_2 \dots M_m = \begin{pmatrix} 1 - a'_1 & a'_1 \\ b'_1 & 1 - b'_1 \end{pmatrix}$$

From Lemma 3.1, the eigenvalues of the product matrix are

$$\lambda = 1 \text{ and } 1 - a'_1 - b'_1.$$

Lemma 3.2 proved that

$$1 - a'_1 - b'_1 = \lambda'_m \lambda'_{m-1} \lambda'_{m-2} \dots \lambda'_2 \lambda'_1$$

The proof is thus completed. ■

It should be pointed out that this closed form expansion of 2×2 matrix multiplication is similar to the fundamental matrix expansion given by Yasui and Yajima, but this formula is much simpler when the individual elements a'_1 , b'_1 , $1 - a'_1$, and $1 - b'_1$ of the above matrix are under consideration.

3.3 Two-State Isolated PSM

Using the properties developed in the previous section, we present an isolated machine, which includes the ones defined by Yasui, Yajima, and Tan. In this section, we treat the PSM for which $\max_i \{|\lambda_i|\} < 1$ for $i = 1, 2, \dots, n$, holds true. We can assume that one state of a PSM is the initial state and the other state of the PSM is the last state (we assume s_1 is the initial state and s_2 is the last state). In general, the initial and last states are either s_1 or s_2 . Thus, after an input string x is accepted by the PSM with the initial state distribution $(1,0)$, the probability of the last state is the $(1,2)$ element of the multiplied matrices $M_1 M_2 \dots M_m$, where the length of the input string is m .

Before we introduce the concept of an isolated machine, a few lemmas are needed.

Lemma 3.3:

Let S be the series,

$$S = a_{m-1} + \lambda_{m-1} a_{m-2} + \lambda_{m-1} \lambda_{m-2} a_{m-3} + \dots + \prod_{\ell=1}^k \lambda_{m-\ell} a_{m-k-1},$$

where $\lambda_j \in \{(1-\alpha_i, -\beta_i)\}$ for $i = 1, 2, \dots, n$ and $j = m, m-1, m-2, \dots$, and $a_{m-\ell} \in \{\alpha_i\}$ for $\ell = 1, 2, \dots$. Let $a = \max_i \{\alpha_i\}$, $c = \min_i \{\alpha_i\}$, $\lambda_a = \max_i \{1-\alpha_i, -\beta_i\}$ and $\lambda_c = \min_i \{1-\alpha_i, -\beta_i\}$ for $i = 1, 2, \dots, n$. The maximum and the minimum of S denoted by S_{\max} and S_{\min} , respectively, are found as follows:

Case 1: If $\lambda_a > \lambda_c > 0$, then

$$S_{\max} = \frac{a}{1-\lambda_a} \quad \text{and} \quad S_{\min} = \frac{c}{1-\lambda_c}.$$

Case 2: If $\lambda_a > 0 > \lambda_c$ and $|\lambda_a| \geq |\lambda_c|$, then

$$S_{\max} = \frac{a}{1-\lambda_a} \quad \text{and} \quad S_{\min} = c + a \frac{\lambda_c}{1-\lambda_a}.$$

Case 3: If $\lambda_a \geq 0 \geq \lambda_c$ and $|\lambda_a| \leq |\lambda_c|$, then

$$S_{\max} = a \frac{1+\lambda_a}{1-\lambda_c^2} \quad \text{and} \quad S_{\min} = c + a \frac{\lambda_c(1+\lambda_a)}{1-\lambda_c^2}.$$

Case 4: If $0 > \lambda_a > \lambda_c$, then

$$S_{\max} = \frac{a}{1-\lambda_c^2} + c \frac{\lambda_a}{1-\lambda_a^2} \quad \text{and} \quad S_{\min} = c + a \frac{\lambda_c}{1-\lambda_c^2}.$$

Proof:

Case 1: Since $\lambda_a > \lambda_c > 0$,

$$S \leq a(1 + \lambda_a + \lambda_a^2 + \lambda_a^3 + \dots) = \frac{a}{1-\lambda_a} = S_{\max}$$

$$S \geq c(1 + \lambda_c + \lambda_c^2 + \lambda_c^3 + \dots) = \frac{c}{1-\lambda_c} = S_{\min}$$

Case 2: Since $\lambda_a \geq 0$ and $|\lambda_a| \geq |\lambda_c|$,

$$S \leq a(1 + \lambda_a + \lambda_a^2 + \lambda_a^3 + \dots) = \frac{a}{1-\lambda_a} = S_{\max}.$$

To find S_{\min} is to find the largest negative value for the partial sum

$$\begin{aligned} & a(\lambda_m + \lambda_m \lambda_{m-1} + \lambda_m \lambda_{m-1} \lambda_{m-2} + \dots + \prod_{\ell=0}^k \lambda_{m-\ell} + \dots) \\ & = a\lambda_m (1 + \lambda_{m-1} + \lambda_{m-1} \lambda_{m-2} + \dots + \prod_{\ell=1}^k \lambda_{m-\ell} + \dots) \end{aligned}$$

To make this partial sum negative, from the condition $0 \geq \lambda_c$, λ_m should be λ_c .

The sum of the terms in the parentheses is maximum when all λ_{m-1} , λ_{m-2} , ... are equal to λ_a . Thus

$$S \geq c + a\lambda_c (1 + \lambda_a + \lambda_a^2 + \lambda_a^3 + \dots) = c + a \frac{\lambda_c}{1 - \lambda_a} = S_{\min}$$

Case 3: To determine S_{\max} , terms with an odd number of λ 's must contain a λ_a in order to be positive and terms with an even number of λ 's must be all λ_c ; thus

$$\begin{aligned} S &\leq a(1 + \lambda_a + \lambda_c^2 + \lambda_a \lambda_c^2 + \lambda_c^4 + \lambda_a \lambda_c^4 + \dots) \\ &= a\{1 + \lambda_c^2 + \lambda_c^4 + \dots + \lambda_a(1 + \lambda_c^2 + \lambda_c^4 + \dots)\} \\ &= a \frac{1 + \lambda_a}{1 - \lambda_c^2} = S_{\max} \end{aligned}$$

To find S_{\min} , we should make each term, except the first constant term, have the largest possible odd number of λ_c 's in order to make it have the largest possible negative value.

$$\begin{aligned} S &\geq c + a\lambda_c + a\lambda_a \lambda_c + a\lambda_c^3 + a\lambda_a \lambda_c^3 + a\lambda_c^5 + \dots \\ &= c + a\lambda_c (1 + \lambda_c^2 + \lambda_c^4 + \dots) + a\lambda_a \lambda_c (1 + \lambda_c^2 + \lambda_c^4 + \dots) \\ &= c + a \frac{\lambda_c (1 + \lambda_a)}{1 - \lambda_c^2} = S_{\min} \end{aligned}$$

Case 4: To determine S_{\max} , terms containing an even number of λ 's should be all λ_c 's and terms containing an odd number of λ 's should be all λ_a 's to make them the largest positive numbers and smallest negative numbers, respectively.

Hence

$$\begin{aligned} S &\leq a + c\lambda_a + a\lambda_c^2 + c\lambda_a^3 + a\lambda_c^4 + c\lambda_a^5 + \dots \\ &= a + a\lambda_c^2 + a\lambda_c^4 + \dots + c\lambda_a (1 + \lambda_a + \lambda_a^2 + \lambda_a^3 + \dots) \\ &= \frac{a}{1 - \lambda_c^2} + c \frac{\lambda_a}{1 - \lambda_a} = S_{\max} \end{aligned}$$

To find the S_{\min} , odd λ -terms, which will be negative, should be made as large as possible and even λ -terms, which will be positive, should be made as small as possible. Thus

$$\begin{aligned} S &\geq c + a\lambda_c + c\lambda_a^2 + a\lambda_c^3 + c\lambda_a^4 + a\lambda_c^5 + \dots \\ &= c(1 + \lambda_a^2 + \lambda_a^4 + \lambda_a^6 + \dots) + a(\lambda_c + \lambda_c^3 + \lambda_c^5 + \dots) \\ &= \frac{c}{1 - \lambda_a^2} + a \frac{\lambda_c}{1 - \lambda_c^2} \geq c + a \frac{\lambda_c}{1 - \lambda_c^2} = S_{\min} . \end{aligned}$$

From the above lemma, we have

Lemma 3.4:

The four upper bounds, S_{\max} 's of S in Lemma 3.3, are always positive, and the four lower bounds, S_{\min} 's of S in the lemma, are positive or negative depending on the quantities of a, c, λ_a , and λ_c . When S_{\min} is negative, the lower bound of S_{\min} is $S_{\min} = c + a\lambda_c \frac{1}{1 - \lambda_{\max}^2}$ where $\lambda_{\max} = \max_i \{|1 - \alpha_i - \beta_i|\}$ for $i = 1, 2, \dots, n$.

Proof:

For the S_{\max} case, it is obvious that S_{\max} 's in Cases 1 and 2 are positive.

In Case 3, since $\lambda_a \geq 0 \geq \lambda_c$ and $|\lambda_a| \leq |\lambda_c|$,

$$S_{\max} = a \frac{1 + \lambda_a}{1 - \lambda_c^2} \geq a \frac{1 + \lambda_a}{1 - \lambda_a^2} = \frac{a}{1 - \lambda_a^2}$$

In Case 4, since $0 \geq \lambda_a \geq \lambda_c$ and $a \geq c$,

$$S_{\max} = \frac{a}{1 - \lambda_c^2} + c \frac{\lambda_a}{1 - \lambda_a^2} \geq \frac{a}{1 - \lambda_a^2} + c \frac{\lambda_a}{1 - \lambda_a^2} \geq \frac{a}{1 - \lambda_a^2} .$$

Since $|\lambda_a| < 1$ and $a > 0$, all S_{\max} 's are bounded by $\frac{a}{1-\lambda_a}$ so that they are positive. For the S_{\min} case, since $|\lambda_c| < 1$ and $c > 0$ in Case 1, then S_{\min} is positive. However, λ_c is negative (or nonpositive) and $a \geq c \geq 0$ in all the other cases; therefore, S_{\min} may be negative or positive, depending on the magnitude of the second term compared with the first one of S_{\min} . When S_{\min} is positive, it is the same as Case 1. When S_{\min} is negative, the following cases hold.

Case 2: Since $\lambda_{\max} = \lambda_a$,

$$S_{\min} = c + a \frac{\lambda_c}{1-\lambda_{\max}}.$$

Case 3: Since $\lambda_a > 0 > \lambda_c$, $|\lambda_a| \leq |\lambda_c|$, and $\lambda_{\max} = |\lambda_c|$,

$$\begin{aligned} S_{\min} &= c + a \lambda_c \frac{1+\lambda_a}{1-\lambda_c^2} \geq c + a \lambda_c \frac{1-\lambda_c}{1-\lambda_c^2} = c + a \frac{\lambda_c}{1+\lambda_c} \\ &= c + a \frac{\lambda_c}{1-\lambda_{\max}}. \end{aligned}$$

Case 4: Since $0 > \lambda_a > \lambda_c$ and $\lambda_{\max} = |\lambda_c|$,

$$S_{\min} = c + a \frac{\lambda_c}{1-\lambda_c^2} \geq c + a \frac{\lambda_c}{1+\lambda_c} = c + a \frac{\lambda_c}{1-\lambda_{\max}}.$$

Lemma 3.5:

Define $M(x_i) = \begin{pmatrix} 1-\alpha_i & \alpha_i \\ \beta_i & 1-\beta_i \end{pmatrix}$. The absolute maximum radius γ of the transferable range with respect to the (1,2) element α_i of the right-most matrix $M(x_i)$ of the ℓ matrix multiplication ($\ell = m, m-1, \dots$) is bounded by

$$\gamma \leq |\lambda_i| S_{\max} = \gamma_{\max}.$$

Proof:

From Theorem 3.1, the (1,2) element of a product of infinite matrices, $\dots M_{m-2} M_{m-1} M(x_i)$, is

$$\gamma_i = \alpha_i + \lambda_i (a_{m-1} + \lambda_{m-1} a_{m-2} + \lambda_{m-1} \lambda_{m-2} a_{m-3} + \dots) \quad (3.8)$$

where

$$\lambda_j \in \{1 - \alpha_k - \beta_k\} \quad \text{for } k = 1, 2, \dots, n \quad \text{and } j = m-1, m-2, m-3, \dots$$

From Lemma 3.3,

$$\begin{aligned} \gamma_i &\leq \alpha_i + |\lambda_i| (a_{m-1} + \lambda_{m-1} a_{m-2} + \lambda_{m-1} \lambda_{m-2} a_{m-3} + \dots) \\ &\leq \alpha_i + |\lambda_i| S_{\max} . \end{aligned}$$

From the definition of γ ,

$$\gamma = \gamma_i - \alpha_i \leq |\lambda_i| S_{\max} = \gamma_{\max} .$$

Note that the product of stochastic matrices is also stochastic; thus the (1,2) element cannot exceed the bounds zero and one. Similarly, the lower bound of γ also exists.

Lemma 3.6:

The absolute radius of the transferable range with respect to the (1,2) element of the right-most matrix $M(x_i)$ of the product of infinite matrices is bounded by

$$\gamma \geq |\lambda_i| S_{\min} = \gamma_{\min}$$

where S_{\min} is defined in Lemma 3.3.

Proof:

Using (3.8) and Lemma 3.3,

$$\gamma_i \geq \alpha_i + \lambda_i S_{\min}$$

Considering the sign of λ_i , the radius, $\gamma = \gamma_i - \alpha_i$, therefore is

$$\gamma \geq |\lambda_i| |S_{\min}| = \gamma_{\min}$$



Definition 3.1:

Considering the sign of λ_i in Lemma 3.5, the directed transferable range from α_i is in $[\alpha_i, \alpha_i + \lambda_i S_{\max}]$ if $\lambda_i > 0$ and is in $[\alpha_i + \lambda_i S_{\max}, \alpha_i]$ if $\lambda_i < 0$. Similarly in Lemma 3.6, the directed transferable range from α_i is in $[\alpha_i, \alpha_i + \lambda_i S_{\min}]$ if λ_i and S_{\min} have the same sign and is in $[\alpha_i + \lambda_i S_{\min}, \alpha_i]$ if the signs of λ_i and S_{\min} are different.

The concept of the absolute radius introduced by Yasui and Yajima and Tan is too conservative. We therefore abandon the concept of the radius of the transferable range of α_i and instead define the maximum transferable range R from α_i as follows:

Definition 3.2:

The transferable range;

$$R_{\max}(\alpha_i) = \min\{1 - \alpha_i, \lambda_i S_{\max}\} \quad \text{if } \lambda_i > 0,$$

$$R_{\max}(\alpha_i) = \min\{\alpha_i, \lambda_i S_{\max}\} \quad \text{if } \lambda_i < 0,$$

$$R_{\min}(\alpha_i) = \min\{\lambda_i S_{\min}, 0\} \quad \text{if } \lambda_i > 0,$$

$$R_{\min}(\alpha_i) = \max\{\lambda_i S_{\min}, 0\} \quad \text{if } \lambda_i < 0,$$

therefore

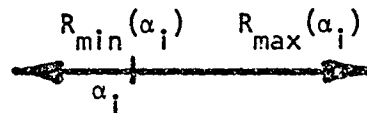
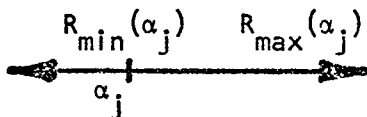
$$R(\alpha_i) = R_{\max}(\alpha_i) - R_{\min}(\alpha_i) \quad \text{if } \lambda_i > 0,$$

and $R(\alpha_i) = -R_{\max}(\alpha_i) + R_{\min}(\alpha_i)$ if $\lambda_i < 0$.

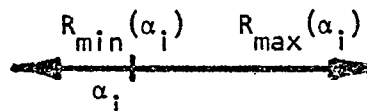
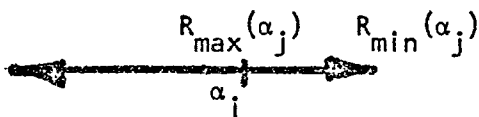
Let α_i be the (1,2) element of the two-state stochastic matrix associated with the input symbol x_i for $i = 1, 2, \dots, n$. When α_i is plotted on the line $[0, 1]$, the line $[0, \alpha_i]$ represents the state transient from s_1 to s_2 of $M(x_i)$. The complement line $[\alpha_i, 1]$, which is the length $1 - \alpha_i$, represents the state transient from s_1 to s_1 of $M(x_i)$. Suppose α_i and α_j are a pair of adjacent (1,2) elements on the line $[0, 1]$ and $\alpha_i > \alpha_j$. Since $\lambda_k S_{\max}$ depends upon the sign of λ_k , the transferable range $R_{\max}(\alpha_k)$ of the (1,2) element of a matrix product for any combination of the matrices described by the order of the input symbols is only in one direction, to the right or to the left from the point α_k , depending on the sign of λ_k . Similarly, the transferable range $R_{\min}(\alpha_k)$ is to the opposite direction from the range of $R_{\max}(\alpha_k)$. Note: When the direction of the transferable range $R_{\min}(\alpha_k)$ is the same as the one of the range $R_{\max}(\alpha_k)$, $R_{\min}(\alpha_k)$ is zero from Definition 3.2.

Let λ_i and λ_j be the eigenvalues of the two-state stochastic matrices $M^{(i)}$ and $M^{(j)}$, respectively. The following four cases involving the adjacent points α_i and α_j on the line $[0, 1]$ are considered separately.

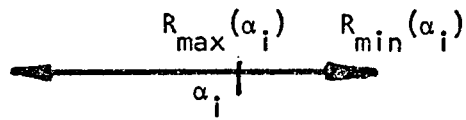
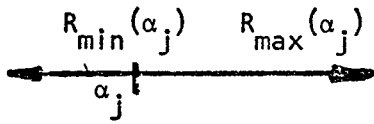
Case 1: $\lambda_i, \lambda_j > 0$,



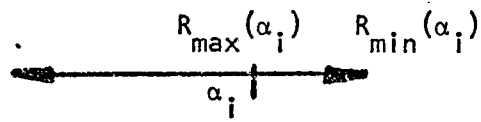
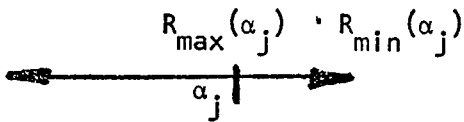
Case 2: $\lambda_i > 0, \lambda_j < 0$



Case 3: $\lambda_i < 0, \lambda_j > 0$



Case 4: $\lambda_i, \lambda_j < 0$



Definition 3.3:

The (1,2) elements α_i and α_j which are adjacent on the line $[0,1]$ are isolated if there is no overlap of transferable ranges from α_i and α_j .

The following theorem is concerned with the isolation of transferable ranges.

Theorem 3.3:

The (1,2) elements α_i and α_j which are adjacent are isolated if the following condition is satisfied. Assuming $\alpha_i > \alpha_j$,

Case 1: If $\lambda_i, \lambda_j > 0$,

$$\alpha_i - \alpha_j > R_{\max}(\alpha_j) - R_{\min}(\alpha_i), \quad (3.9a)$$

Case 2: If $\lambda_i > 0, \lambda_j < 0$,

$$\alpha_i - \alpha_j > R_{\min}(\alpha_j) - R_{\min}(\alpha_i), \quad (3.9b)$$

Case 3: If $\lambda_i < 0, \lambda_j > 0$,

$$\alpha_i - \alpha_j > R_{\max}(\alpha_j) - R_{\max}(\alpha_i), \quad (3.9c)$$

Case 4: If $\lambda_i, \lambda_j < 0$,

$$\alpha_i - \alpha_j \geq R_{\min}(\alpha_j) - R_{\max}(\alpha_i), \quad (3.9d)$$

Proof:

It is evident from Lemmas 3.5 and 3.6 and Definitions 3.2 and 3.3. ■

Before we introduce a new concept of the "absolutely" isolated machine, one additional lemma is needed.

Lemma 3.7:

Define $\lambda_{\max} = \max_i \{ |1 - \alpha_i - \beta_i| \}$ for $i = 1, 2, \dots, n$.

The upper bound of the four S_{\max} 's in Lemma 3.3 denoted by \bar{S}_{\max} is $a/(1 - \lambda_{\max})$.

Proof:

In Cases 1 and 2, it is obvious. In Case 3, since $\lambda_a \geq 0 \geq \lambda_c$, $|\lambda_a| \leq |\lambda_c|$, and $\lambda_{\max} = |\lambda_c|$,

$$S_{\max} = a \frac{1 + \lambda_a}{1 - \lambda_c^2} < a \frac{1 - \lambda_c}{1 - \lambda_c^2} = \frac{a}{1 + \lambda_c} = \frac{a}{1 - |\lambda_c|},$$

In Case 4, since $0 \geq \lambda_a \geq \lambda_c$, and $\lambda_{\max} = |\lambda_c|$,

$$\begin{aligned} S_{\max} &= \frac{a}{1 - \lambda_c^2} + c \frac{\lambda_a}{1 - \lambda_a^2} \leq \frac{a}{1 - \lambda_c^2} - a \frac{\lambda_c}{1 - \lambda_c^2} = \frac{a}{1 + \lambda_c} \\ &= \frac{a}{1 - |\lambda_c|} \end{aligned}$$

To summarize

$$S_{\max} \leq \bar{S}_{\max} = \frac{a}{1 - \lambda_{\max}} \quad \blacksquare \quad (3.10)$$

By substituting \bar{S}_{\max} into S_{\max} of Definition 3.2, a new $R_{\max}(\alpha_i)$ can be defined with Definition 3.2 in a similar manner and can be denoted as $\bar{R}_{\max}(\alpha_i)$.

3.4 An Absolutely Isolated Machine

The definition of an absolutely isolated machine is quite similar to the completely isolated machine introduced by Yasui, Yajima, and Tan. However, the class of absolutely isolated machines contains the class of completely isolated machines. In other words, the latter is a subset of the former.

Definition 3.4:

A machine is absolutely isolated if all adjacent pairs α_i and α_j are isolated.

From the preceding statement, we have the following theorem.

Theorem 3.4:

A PSM is absolutely isolated if all adjacent pairs (α_j, α_i) on the line $[0,1]$ of the $(1,2)$ element of the two-state stochastic matrices $M^{(k)}$ for $k = 1, 2, \dots, n$ satisfy the following conditions:

Case 1: If $\lambda_i, \lambda_j > 0$,

$$\alpha_i - \alpha_j > R_{\max}(\alpha_j) - R_{\min}(\alpha_i), \quad (3.11a)$$

Case 2: If $\lambda_i > 0, \lambda_j < 0$,

$$\alpha_i - \alpha_j > R_{\min}(\alpha_j) - R_{\min}(\alpha_i), \quad (3.11b)$$

Case 3: If $\lambda_i < 0, \lambda_j > 0$,

$$\alpha_i - \alpha_j > R_{\max}(\alpha_j) - R_{\max}(\alpha_i), \quad (3.11c)$$

Case 4: If $\lambda_i, \lambda_j \leq 0$,

$$\alpha_i - \alpha_j - R_{\min}(\alpha_j) - R_{\max}(\alpha_i) \quad (3.12d)$$

where $\alpha_i > \alpha_j$.

Proof:

The proof again is evident from Lemmas 3.3 and 3.6, Theorem 3.3, Definitions 3.2 and 3.4 and the fact that the transferable ranges of the pairs $[0, c]$ and $[a, 1]$ with negative and positive λ 's respectively, are again $[0, c]$ and $[a, 1]$. This is correct because the (1,2) element of the stochastic matrix product never exceeds the extreme points zero and one. ■

Definition 3.5:

When we are interested only in the last $k+1$ symbols, which correspond to the last $k+1$ matrices from the rightmost of an infinite matrix product, the series $S_{\inf} = a_m + \lambda_m a_{m-1} + \lambda_m \lambda_{m-1} a_{m-2} + \lambda_m \lambda_{m-1} \lambda_{m-2} a_{m-3} + \dots$ is truncated at the term $\lambda_m \lambda_{m-1} \dots \lambda_{m-k+1}$. It is called the k^{th} approximation of the infinite series

S_{\inf} . More precisely,

$$S_k = a_m + \lambda_m a_{m-1} + \lambda_m \lambda_{m-1} a_{m-2} + \dots + \lambda_m \lambda_{m-1} \dots \lambda_{m-k+1} a_{m-k}.$$

3.5 The k^{th} Approximation of the Absolutely Isolated Machine

Consider the k^{th} approximation of the infinite series S_{\inf} and we have the following lemma.

Lemma 3.3a:

The S_{\max} and S_{\min} of the term S (in Lemma 3.3) of the S_k are as follows:
where $S = a_{m-1} + \lambda_{m-1} a_{m-2} + \dots + \prod_{\ell=1}^{k-1} \lambda_{m-\ell} a_{m-k}$,

Case 1: If $\lambda_a > \lambda_c > 0$, then

$$S_{\max} = a \frac{1-\lambda_a^k}{1-\lambda_a} \quad \text{and} \quad S_{\min} = c \frac{1-\lambda_c^k}{1-\lambda_c} \quad (3.12)$$

Case 2: If $\lambda_a > 0 > \lambda_c$ and $|\lambda_a| \geq |\lambda_c|$, then

$$S_{\max} = a \frac{1-\lambda_a^k}{1-\lambda_a} \quad \text{and} \quad S_{\min} = c + a \frac{\lambda_c (1-\lambda_a^{k-1})}{1-\lambda_a} \quad (3.14)$$

Case 3: If $\lambda_a > 0 > \lambda_c$ and $|\lambda_a| < |\lambda_c|$, then

$$S_{\max} = a(1+\lambda_a) \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} + a\lambda_c^{k-1} \quad k \geq 3, \quad k = \text{odd} \quad (3.15a)$$

$$S_{\max} = a(1+\lambda_a) \frac{1-\lambda_c^k}{1-\lambda_c^2} \quad k \geq 2, \quad k = \text{even}$$

$$S_{\min} = c + a\lambda_c(1+\lambda_a) \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} \quad k \geq 3, \quad k = \text{odd} \quad (3.15b)$$

$$S_{\min} = c + a\lambda_c(1+\lambda_a) \frac{1-\lambda_c^{k-2}}{1-\lambda_c^2} + a\lambda_c^{k-1} \quad k \geq 4, \quad k = \text{even}$$

Case 4: If $0 > \lambda_a > \lambda_c$, then

$$S_{\max} = a \frac{1-\lambda_c^{k+1}}{1-\lambda_c^2} + c\lambda_a \frac{(1-\lambda_a^{k-1})}{1-\lambda_a^2} \quad k \geq 3, \quad k = \text{odd} \quad (3.16a)$$

$$S_{\max} = a \frac{1-\lambda_c^k}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_a^k}{1-\lambda_a^2} \quad k \geq 2, \quad k = \text{even}$$

and

$$S_{\min} = c \frac{1-\lambda_a^{k+1}}{1-\lambda_a^2} + a\lambda_c \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} \quad k \geq 3, \quad k = \text{odd} \quad (3.16b)$$

$$S_{\min} = c \frac{1-\lambda_a^k}{1-\lambda_a^2} + a\lambda_c \frac{1-\lambda_c^k}{1-\lambda_c^2} \quad k \geq 2, \quad k = \text{even}$$

Proof:

The proof of this lemma is similar to that of Lemma 3.3 for the infinite series case; we may thus omit it. ■

In parallel to Lemmas 3.4 and 3.7 for S_{\min} and \bar{S}_{\max} respectively, we have the following lemma.

Lemma 3.8:

The upper bound, denoted as \bar{S}_{\max} , of the four S_{\max} 's in Lemma 3.3a is $a(1-\lambda_{\max}^k)/1-\lambda_{\max}$, and when S_{\min} is negative, there exists the lower bound of S_{\min} , which is denoted as $S_{\min} = c + a\lambda_c(1-\lambda_{\max}^{k-1})/1-\lambda_{\max}$, where $\lambda_{\max} = \max_i \{|1-\alpha_i-\beta_i|\}$ for $i = 1, 2, \dots, n$ and k indicates the first k (finite) terms in S .

Proof:

For \bar{S}_{\max} :

Cases 1 and 2: $\lambda_{\max} = \lambda_a$. Thus

$$\bar{S}_{\max} = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}$$

Case 3: There are two subcases to be considered. Since $|\lambda_c| < 1$, $\lambda_c^{k-3} > \lambda_c^{k-1}$ for $k = 4, 5, \dots$ and from $\lambda_a \geq 0 \geq \lambda_c$ and $|\lambda_a| \leq |\lambda_c|$,

(a) k is odd

$$\begin{aligned} a(1+\lambda_a) \frac{1-\lambda_c^{k-3}}{1-\lambda_c^2} + a\lambda_c^{k-1} &\leq a(1-\lambda_c) \frac{1-\lambda_c^{k-3}}{1-\lambda_c^2} + a\lambda_c^{k-1} \\ &= a \frac{(1-\lambda_c)}{(1-\lambda_c)(1+\lambda_c)} \{1-\lambda_c^{k-3} + \lambda_c^{k-1}(1+\lambda_c)\} \\ &\leq a \frac{1}{1-|\lambda_c|} \{1-\lambda_c^{k-1} + \lambda_c^{k-1}(1-|\lambda_c|)\} \end{aligned}$$

$$= a \frac{1}{1-|\lambda_c|} (1-\lambda_c^{k-1} |\lambda_c|) = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}$$

(b) k is even

Similarly,

$$a(1+\lambda_a) \frac{1-\lambda_c^{k-2}}{1-\lambda_c^2} \leq a \frac{1-\lambda_c^{k-2}}{1-|\lambda_c|} \leq a \frac{1-\lambda_c^k}{1-|\lambda_c|} = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}$$

Case 4: From $0 \geq \lambda_a > \lambda_c$ ($|\lambda_a| \leq |\lambda_c| < 1$) and $\lambda_c^{k-3} > \lambda_c^{k-1}$ for $k = 4, 5, \dots$,

(a) k is odd,

$$\begin{aligned} & a \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_a^{k-3}}{1-\lambda_a^2} \leq a \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_c^{k-3}}{1-\lambda_a^2} \\ & \leq a \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} - c\lambda_c \frac{1-\lambda_c^{k-3}}{1-\lambda_a^2} \leq a \frac{1}{1-\lambda_c^2} \{1-\lambda_c^{k-1} - \lambda_c(1-\lambda_c^{k-3})\} \\ & \leq a \frac{1}{1-\lambda_c^2} \{1-\lambda_c^{k-1} - \lambda_c(1-\lambda_c^{k-1})\} = a \frac{1-\lambda_c}{1-\lambda_c^2} (1-\lambda_c^{k-1}) \\ & = a \frac{1-\lambda_c^{k-1}}{1-|\lambda_c|} \leq a \frac{1-|\lambda_c|^k}{1-|\lambda_c|} = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}, \end{aligned}$$

(b) k is even,

$$\begin{aligned} & a \frac{1-\lambda_c^{k-2}}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_a^{k-2}}{1-\lambda_a^2} \leq a \frac{1-\lambda_c^{k-2}}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_c^{k-2}}{1-\lambda_a^2} \\ & \leq a \frac{1-\lambda_c^{k-2}}{1-\lambda_c^2} - c\lambda_c \frac{1-\lambda_c^{k-2}}{1-\lambda_a^2} \leq a \frac{1-\lambda_c}{1-\lambda_c^2} (1-\lambda_c^{k-2}) \\ & = a \frac{1-\lambda_c^{k-2}}{1-|\lambda_c|} \leq a \frac{1-\lambda_c^k}{1-|\lambda_c|} = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}. \end{aligned}$$

For S_{\min} , S_{\min} may be negative in Cases 2, 3, and 4.

Case 2: Since $\lambda_{\max} = |\lambda_a|$ and from (3.14)

$$S_{\min} = c + a\lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}}.$$

Case 3: Since $\lambda_a > 0 > \lambda_c$ and $\lambda_{\max} = |\lambda_c|$, and from (3.15b),

(a) k is odd,

$$\begin{aligned} S_{\min} &= c + a\lambda_c (1 + \lambda_a) \frac{1 - \lambda_c^{k-1}}{1 - \lambda_c^2} \geq c + a\lambda_c (1 - \lambda_c) \frac{1 - \lambda_c^{k-1}}{1 - \lambda_c^2} \\ &= c + a\lambda_c \frac{1 - \lambda_c^{k-1}}{1 + \lambda_c} = c + a\lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}}, \end{aligned}$$

(b) k is even,

$$\begin{aligned} S_{\min} &= c + a\lambda_c (1 + \lambda_a) \frac{1 - \lambda_c^{k-2}}{1 - \lambda_c^2} + a\lambda_c^{k-1} \\ &\geq c + a\lambda_c (1 - \lambda_c) \frac{1 - \lambda_c^{k-2}}{1 - \lambda_c^2} + a\lambda_c^{k-1} \\ &= c + a\lambda_c \left\{ \frac{1 - \lambda_c^{k-2} + \lambda_c^{k-2}(1 + \lambda_c)}{1 + \lambda_c} \right\} = c + a\lambda_c \frac{1 + \lambda_c^{k-1}}{1 + \lambda_c} \\ &= c + a\lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}}. \end{aligned}$$

Case 4: Since $0 > \lambda_a > \lambda_c$ and $\lambda_{\max} = |\lambda_c|$ and from (3.16b),

(a) k is odd,

$$S_{\min} \geq c + a\lambda_c \frac{1 - \lambda_c^{k-1}}{1 - \lambda_c^2} \geq c + a\lambda_c \frac{1 - \lambda_c^{k-1}}{1 + \lambda_c} = c + a\lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}},$$

(b) k is even,

$$S_{\min} \geq c + a\lambda_c \frac{1 - \lambda_c^k}{1 - \lambda_c^2} \geq c + a\lambda_c \frac{1 - \lambda_c^{k-1}}{1 + \lambda_c} = c + a\lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}}.$$

Considering an input string with $k+1$ symbols. The matrix product of $M_{m-k} \dots M_{m-1} M_m$ represents the behavior of a PSM due to the input. Suppose that the last symbol is x_i , $M_m = M^{(i)}$ and the rest $M_{m-1}, M_{m-2}, \dots, M_{m-k} \in \{M^{(j)}\}$ for $j = 1, 2, \dots, n$, then the transferable range of α_i with the $k+1$ input length can be drawn on the line $[0, 1]$ in a similar manner as Theorem 3.3.

Definition 3.6:

If there are no overlapped ranges between two transferable ranges of α_i and α_j after taking $k+1$ input symbols, the pair (α_j, α_i) is called the k^{th} isolated pair where α_j and α_i are the $(1, 2)$ elements of $M^{(j)}$ and $M^{(i)}$, respectively.

The next lemma describes the k^{th} isolated adjacent pair (α_j, α_i) .

Lemma 3.9:

An adjacent pair (α_j, α_i) on the line $[0, 1]$ is the k^{th} isolated pair if k is determined by the following equation. From Theorem 3.3 and Definition 3.2, we need to consider two cases or S_{\min} is nonnegative and S_{\min} is negative. By assuming $\alpha_j < \alpha_i$,

When $S_{\min} \geq 0$,

Case 1: $\lambda_i, \lambda_j \geq 0$,

$$[k] = \lceil \ln \left\{ 1 - \frac{\alpha_i - \alpha_j}{\lambda_j a} (1 - \lambda_{\max}) \right\} / \ln \lambda_{\max} \rceil. \quad (3.17a)$$

Case 2: $\lambda_i > 0, \lambda_j < 0,$

k is arbitrary

(3.17b)

Case 3: $\lambda_i < 0, \lambda_j > 0,$

$$[k] = \ln\{1 - \frac{\alpha_i^{-\alpha_j}}{(|\lambda_i| + |\lambda_j|)^a} (1 - \lambda_{\max})\} / \ln \lambda_{\max} . \quad (3.17c)$$

Case 4: $\lambda_i < 0, \lambda_j < 0,$

$$[k] = \ln\{1 - \frac{\alpha_i^{-\alpha_j}}{|\lambda_i|^a} (1 - \lambda_{\max})\} / \ln \lambda_{\max} . \quad (3.17d)$$

When $S_{\min} < 0,$

Case 1: $\lambda_i, \lambda_j > 0,$

$$[k-1] = \ln\{1 - \frac{\alpha_i^{-\alpha_j} + c\lambda_i}{a(\lambda_j - \lambda_i \lambda_c)} (1 - \lambda_{\max})\} / \ln \lambda_{\max} \quad (3.18a)$$

Case 2: $\lambda_i > 0, \lambda_j < 0$

$$[k-1] = \ln\{1 - (\frac{\alpha_i^{-\alpha_j}}{\lambda_j - \lambda_i} - c) \frac{1 - \lambda_{\max}}{a\lambda_c}\} / \ln \lambda_{\max} \quad (3.18b)$$

Case 3: $\lambda_i < 0, \lambda_j > 0,$

$$[k] = \ln\{1 - \frac{\alpha_i^{-\alpha_j}}{a(\lambda_j - \lambda_i)} (1 - \lambda_{\max})\} / \ln \lambda_{\max} . \quad (3.18c)$$

Case 4: $\lambda_i < 0, \lambda_j < 0,$

$$[k-1] = \ln\{1 - \frac{\alpha_i^{-\alpha_j} - c\lambda_i}{a(\lambda_c \lambda_j - \lambda_i)} (1 - \lambda_{\max})\} / \ln \lambda_{\max} \quad (3.18d)$$

where $[k]$ is Gauss notation, and

$$a = \max_{\ell} \{\alpha_{\ell}\}, c = \min_{\ell} \{\alpha_{\ell}\}, \lambda_{\max} = \max_{\ell} \{|1 - \alpha_{\ell} - \beta_{\ell}|\}$$

$$\lambda_i = 1 - \alpha_i - \beta_i, \lambda_j = 1 - \alpha_j - \beta_j \text{ and "ln"}$$

is the natural logarithm function which is only defined for a nonnegative argument.

Proof:

From Theorem 3.3 and Definition 3.2, and Lemma 3.8, we have $S_{\min} \geq 0$;

$R_{\min}(\alpha_i)$ and $R_{\min}(\alpha_j)$ are zero by Definition 3.2,

$$\text{Case 1: } \alpha_i - \alpha_j \geq \lambda_j \quad \bar{S}_{\max} \quad (3.19a)$$

$$\text{Case 2: } \alpha_i - \alpha_j \geq 0 \quad (\text{No Restriction}) \quad (3.19b)$$

$$\text{Case 3: } \alpha_i - \alpha_j \geq (|\lambda_i| + |\lambda_j|) \quad \bar{S}_{\max} \quad (3.19c)$$

$$\text{Case 4: } \alpha_i - \alpha_j \geq |\lambda_i| \quad \bar{S}_{\max} \quad (3.19d)$$

$S_{\min} < 0$; $R_{\min}(\alpha_i)$ and $R_{\min}(\alpha_j)$ are negative by Definition 3.2,

$$\text{Case 1: } \alpha_i - \alpha_j \geq \lambda_j \quad \bar{S}_{\max}^{-\lambda_i} S_{\min} = \lambda_j a \frac{1 - \lambda_{\max}^k}{1 - \lambda_{\max}} - \lambda_i (c + a \lambda_c \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}}),$$

$$\text{Since } |\lambda_{\max}| < 1 \text{ and } 1 - \lambda_{\max}^k < 1 - \lambda_{\max}^{k-1},$$

$$\alpha_i - \alpha_j \geq a(\lambda_j - \lambda_i \lambda_c) \frac{1 - \lambda_{\max}^{k-1}}{1 - \lambda_{\max}} - \lambda_i c. \quad (3.20a)$$

$$\text{Case 2: } \alpha_i - \alpha_j \geq \lambda_j \quad \bar{S}_{\min}^{-\lambda_i} S_{\min} = (\lambda_j - \lambda_i) S_{\min}. \quad (3.20b)$$

$$\text{Case 3: } \alpha_i - \alpha_j \geq \lambda_j \quad \bar{S}_{\max}^{-\lambda_i} \bar{S}_{\max} = (\lambda_j - \lambda_i) \bar{S}_{\max}. \quad (3.20c)$$

Case 4: $\alpha_i - \alpha_j > \lambda_j S_{\min} - \lambda_i \bar{S}_{\max} = \lambda_j (c + a\lambda_c \frac{1-\lambda_{\max}^{k-1}}{1-\lambda_{\max}}) - \lambda_i a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}$

Since $1-\lambda_{\max}^k \leq 1-\lambda_{\max}^{k-1}$,

$$\alpha_i - \alpha_j \geq a(\lambda_c \lambda_j - \lambda_i) \frac{1-\lambda_{\max}^{k-1}}{1-\lambda_{\max}} + c\lambda_j \quad (3.20d)$$

where $\bar{S}_{\max} = a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}$ and $S_{\min} = c + a\lambda_c \frac{1-\lambda_{\max}^{k-1}}{1-\lambda_{\max}}$.

The k 's are found by solving the above equations (3.19a), (3.19c), (3.19d), (3.20a), (3.20b), (3.20c) and (3.20d) with an equals sign substituted for the greater than or equal sign.

When comparisons between the $a/1-\lambda_{\max}$ and $a(1-\lambda_{\max}^k)/1-\lambda_{\max}$ and the $c+a\lambda_c/1-\lambda_{\max}$ and $c+a\lambda_c(1-\lambda_{\max}^{k-1})/1-\lambda_{\max}$ are made, the differences between these comparisons suggest that if the pair (α_j, α_i) is isolated by an infinite length of input, then the pair is also isolated by a finite length of input (or by the k^{th} input length, namely by the k^{th} approximation of the matrix product $M_{m-k} \dots M_{m-1} M_m$ for $k = 1, 2, 3, \dots$). This is described in the following lemma.

Lemma 3.10:

If an adjacent pair (α_j, α_i) on the line $[0, 1]$ is isolated in the transferable ranges of α_j and α_i with an infinite length of input x^* , then the pair (α_j, α_i) is also isolated in the transferable ranges with the $k+1$ length of input (by the k^{th} approximation).

Proof:

It is evident from $a(1-\lambda_{\max}^k)/1-\lambda_{\max} < a/1-\lambda_{\max}$ for \bar{S}_{\max} , and for S_{\min} , $c+a\lambda_c(1-\lambda_{\max}^{k-1})/1-\lambda_{\max} > c+a\lambda_c/1-\lambda_{\max}$ when $S_{\min} < 0$, where $0 < \lambda_{\max} < 1$.

We are now in a position to introduce the k^{th} (absolutely) isolated machine.

Theorem 3.5:

A PSM is the k^{th} (absolutely) isolated machine if all transferable ranges of adjacent pairs of the $(1,2)$ element of $M^{(i)}$ for $i = 1, 2, \dots, n$ are at least the k^{th} isolated adjacent pair.

Proof:

From Lemmas 3.3a, 3.9, and 3.10 the k_{ℓ}^{th} isolation for each adjacent pair of the $(1,2)$ elements for $\ell = 0, 1, 2, \dots, n$ can be determined.

Take k such as the minimum $\{k_{\ell}\}$ for $\ell = 1, 2, \dots, n$, so that all adjacent pairs are absolutely isolated at least by the k^{th} approximation. □

Example 3.1:

Consider the following example, which is an isolated machine in both Yasui and Yajima's sense (completely isolated machine) and the sense of Theorem 3.4 (absolutely isolated machine).

$$\text{PSM} = (S, X, Y, \{M(x)\}, \Lambda(s))$$

where $S = \{s_1, s_2\}$, $X = \{x_1, x_2\}$, $Y = \{a, b\}$,

$$M(x_1) = \begin{pmatrix} 0.8 & 0.2 \\ 0.7 & 0.3 \end{pmatrix} \quad \text{and} \quad M(x_2) = \begin{pmatrix} 0.6 & 0.4 \\ 0.5 & 0.5 \end{pmatrix}$$

and

$$\Lambda(s_1) = a \quad \text{and} \quad \Lambda(s_2) = b.$$

(a) Completely Isolated Machine:

The criterion for the completely isolated machine, using Yasui and Yajima's result,

$$h = \frac{2\lambda_{\max}}{1-\lambda_{\max}} \frac{\max\{||H||, ||M_2(x_1)||, ||M_2(x_2)||\}}{||H||}.$$

The norm and the matrices H , $M_2(x_1)$ and $M_2(x_2)$ are defined in Section 3.1. If $h\lambda_{\frac{k}{a-c}}^k$ then a PSM is isolated by the k^{th} approximation (or the $k+1$ input length) where $\lambda_a = \max\{1-\alpha_i-\beta_i\}$, $\lambda_c = \min\{1-\alpha_i-\beta_i\}$ and $\lambda_{\max} = \max\{1-\alpha_i-\beta_i\}$ for $i=1,2$. The preceding PSM has the following quantities:

$$\lambda_a = \lambda_c = 0.1, ||H|| = 0.18/0.81, ||M_2(x_1)|| = 2/9, ||M_2(x_2)|| = 4/9,$$

$$\text{and } h = 2 \times 0.1 \times 0.81 \times 4 / 0.9 \times 0.18 \times 9 = 4/9 < 1,$$

then $(4/9)0.1^k < 0.1^k$ for $k = 0, 1, 2, \dots$; it is an completely isolated machine.

(b) Absolutely Isolated Machine:

Since $\lambda_a = \lambda_c = 0.1 > 0$, the PSM belongs in Case 1 of Lemma 3.3 or of Lemma 3.3a and S_{\min} is positive. By Definition 3.2, Theorem 3.3 and from (3.9a) or (3.19a), we have

$$\alpha_i - \alpha_j > \lambda_j \bar{S}_{\max} = \lambda_j \frac{a}{1-\lambda_{\max}} \quad (\text{or } = \lambda_j a \frac{1-\lambda_{\max}^k}{1-\lambda_{\max}}),$$

where $\alpha_i = 0.4$, $\alpha_j = 0.2$, $\lambda_j = 0.1$, $a = 0.4$ and $\lambda_{\max} = 0.1$.

Therefore, the following equation is held true,

$$0.4 - 0.2 > 0.1 \times 0.4 \frac{1-0.1^k}{0.9}$$

$$0.2 > \frac{0.04}{0.9} (1-0.1^k) \quad \text{for } k = 1, 2, 3, \dots$$

For the other pairs (the edge pairs),

$$\alpha_i = 0.2, \alpha_j = 0., \lambda_a = 0.1 \text{ and } \lambda_{\max} = 0.1$$

$$0.2 - 0.1 > 0.1 \times 0.4 \frac{1 - 0.1^k}{0.9} \quad \text{for } k = 0, 1, 2, \dots$$

$$\alpha_i = 1.0, \alpha_j = 0.4, \lambda_a = 0.1 \text{ and } \lambda_{\max} = 0.1$$

$$1.0 - 0.4 > 0.1 \times 0.4 \frac{1 - 0.1^k}{0.9} \quad \text{for } k = 0, 1, 2, \dots$$

so it is an absolutely isolated machine by the k^{th} approximation for $k = 1, 2, 3, \dots$

Example 3.2:

Consider another example of a PSM which is not a completely isolated machine but which is indeed an absolutely isolated machine. We assume the same sets of states, input, output, and output functions from the previous example but different state transition matrices:

$$M(x_1) = \begin{pmatrix} 0.2 & 0.8 \\ 0.1 & 0.9 \end{pmatrix} \quad \text{and} \quad M(x_2) = \begin{pmatrix} 0.5 & 0.5 \\ 0.9 & 0.1 \end{pmatrix}$$

(a) Completely Isolated Machine:

The criterion for the completely isolated machine,

$$a = 0.8, c = 0.5, \lambda_a = 0.1, \lambda_c = -0.4 \text{ and } \lambda_{\max} = 0.4,$$

$$||H|| = \frac{0.1 \times 0.5 - 0.8 \times 0.9}{0.9 \times 1.4} = \frac{0.68}{1.26},$$

$$||M_2(x_1)|| = \frac{8}{9}, \quad ||M_2(x_2)|| = \frac{5}{14},$$

$$\text{Then } h = \frac{2 \times 0.4}{0.6} \frac{8}{9} \frac{1.26}{0.68} = 2.1 > 1.$$

Therefore it is not a completely isolated machine.

(b) Absolutely Isolated Machine:

Since $\lambda_a = 0.1 > 0$ and $\lambda_c = -0.4 < 0$ and $|\lambda_a| < |\lambda_c|$, the PSM is in Case 3 of Lemma 3.3 or Lemma 3.3a, but $S_{\min} (= c + a\lambda_c \frac{1+\lambda_a}{1-\lambda_c^2} = 0.5 - 0.8 \times 0.4 \frac{1+0.1}{1-0.16} = 0.5 - 0.381)$ is positive. By Theorem 3.3 or Lemma 3.9, $\alpha_i = 0.8, \alpha_j = 0.5, \lambda_i = 0.1$ and $\lambda_j = -0.4$; the pair $(0.5, 0.8)$ is in Case 2 in the theorem or the lemma. From Definition 3.2, $R_{\min}(\alpha_i) = R_{\min}(\alpha_j) = 0$; thus, $\alpha_i - \alpha_j = 0.8 - 0.5 > 0$ in (3.9b) or, (3.19b) is true.

The transferable ranges of the other pairs $[0, 0.5]$ and $[0.8, 1.0]$ are the same ranges $[0, 0.5]$ and $[0.8, 1.0]$, respectively. Because the stochastic matrix product is again the stochastic matrix, the $(1, 2)$ element of the product never exceeds the extreme points zero and one.

This is the absolutely isolated machine.

Truncation Error (with the $k-1$ approximation)

The truncation by the last k symbols in an input string denoted as γ_i and expressed by (3.8) over $x \in X^*$ for $i = 1, 2, \dots, n$, induces an error $\epsilon^{(k)}$ defined as follows.

Definition 3.7:

The infinite series γ_i of (3.8) can be divided into two parts, the first k terms (corresponding to the last k input symbols of a matrix product) and the remainder such as

$$\gamma_i = \gamma_i^{(k)} + \epsilon^{(k)} \quad (3.21)$$

$$\begin{aligned} \text{where } \gamma_i^{(k)} = & \alpha_i + \lambda_i (a_{m-1} + \lambda_{m-1} a_{m-2} + \lambda_{m-1} \lambda_{m-2} a_{m-3} + \dots \\ & + \prod_{j=1}^{k-2} \lambda_{m-j} a_{m-k+1}) \end{aligned} \quad (3.22)$$

and

$$\begin{aligned} \epsilon^{(k)} = & \lambda_i \prod_{j=1}^{k-1} \lambda_{m-j} (a_{m-k} + \lambda_{m-k} a_{m-k-1} \\ & + \lambda_{m-k} \lambda_{m-k-1} a_{m-k-2} + \lambda_{m-k} \lambda_{m-k-1} \lambda_{m-k-2} a_{m-k-3} \\ & + \dots) \end{aligned} \quad (3.23)$$

for $i = 1, 2, \dots, n$.

Theorem 3.6:

By cutting the first k terms of γ_i , the truncation error $|\epsilon^{(k)}|$ is bounded on both sides; the upper and lower bounds are

$$\underline{\gamma}^{(k)} = |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| s_{\min} \quad \text{and} \quad (3.24)$$

$$\overline{\gamma}^{(k)} = |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| \overline{s}_{\max}, \quad (3.25)$$

therefore

$$\underline{\gamma}^{(k)} \leq |\epsilon^{(k)}| \leq \overline{\gamma}^{(k)} \quad (3.26)$$

where

$a = \max_i \{\alpha_i\}$, $c = \min_i \{\alpha_i\}$, and s_{\min} and \overline{s}_{\max} are defined in Lemmas 3.3 and 3.7, respectively, for $i = 1, 2, \dots, n$.

Proof:

From Lemmas 3.3, 3.4, 3.5, 3.6, and 3.7 and Theorem 3.3,

$$\begin{aligned} |\epsilon^{(k)}| & \leq |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| a (1 + \lambda_{m-k} + \lambda_{m-k} \lambda_{m-k-1} + \dots) \\ & \leq |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| s_{\max} \leq |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| \overline{s}_{\max}, \end{aligned}$$


and similarly

$$|\varepsilon^{(k)}| \geq |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| |c(1 + \lambda_{m-k} + \lambda_{m-k} \lambda_{m-k-1} + \dots)|$$

$$\geq |\lambda_i| \prod_{j=1}^{k-1} |\lambda_{m-j}| |S_{\min}|.$$

Since $1 + \lambda_{m-k} + \lambda_{m-k} \lambda_{m-k-1} + \dots$ where

$$\lambda_{m-\ell} \in \{1 - \alpha_i, -\beta_i\} \quad \text{for } i = 1, 2, \dots, n \text{ and } \ell = k, k+1, \dots,$$

thus the bounded sum is described in Lemma 3.3. 

The error bounds of (3.23) are directed by the signs of $(\lambda_i \prod_{j=1}^{k-1} \lambda_{m-j})$ and S_{\min} , therefore, the upperbound error exists in only one direction.

Corollary 3.7:

When S_{\min} is nonnegative,

$$0 \leq \underline{\gamma}^{(k)} \leq \varepsilon^{(k)} \leq \bar{\gamma}^{(k)} \quad \text{if } \text{sign}(\lambda_i \prod_{j=1}^{k-1} \lambda_{m-j}) > 0, \quad (3.27a)$$

$$-\bar{\gamma}^{(k)} \leq \varepsilon^{(k)} \leq -\underline{\gamma}^{(k)} \leq 0 \quad \text{if } \text{sign}(\lambda_i \prod_{j=1}^{k-1} \lambda_{m-j}) < 0, \quad (3.27b)$$

when S_{\min} is negative,

$$\underline{\gamma}^{(k)} \leq 0 \leq \varepsilon^{(k)} \leq \bar{\gamma}^{(k)} \quad \text{if } \text{sign}(\lambda_i \prod_{j=1}^{k-1} \lambda_{m-j}) > 0, \quad (3.27c)$$

$$-\bar{\gamma}^{(k)} \leq \varepsilon^{(k)} \leq 0 \leq -\underline{\gamma}^{(k)} \quad \text{if } \text{sign}(\lambda_i \prod_{j=1}^{k-1} \lambda_{m-j}) < 0 \quad (3.27d)$$

Proof:

From Definition 3.7 and Theorem 3.6, it is obvious. 

For future reference, all (3.27) equations can be denoted as

$$|\varepsilon^{(k)}| \leq (\underline{\gamma}^{(k)}, \overline{\gamma}^{(k)})$$

3.6 Two-State Input-Traceable Machine

When a two-state PSM is absolutely isolated, all past input symbols can be determined precisely; therefore, the past history of the behavior of the PSM may be traceable from the known initial state distribution (1,0) and the present state distribution.

From the property of the absolutely isolated machine, we have the following corollary.

Corollary 3.8:

Given an absolutely isolated PSM and if the value of the (1,2) element of the matrix product is known after an input string has been accepted, then we can determine all symbols supplied in the input string.

Proof:

Let V be the value of the (1,2) element of the matrix product,

$$V = a_m + \lambda_m (a_{m-1} + \lambda_{m-1} (a_{m-2} + \lambda_{m-2} (a_{m-3} + \dots) \dots))$$

where $a_j \in \{\alpha_i\}$ and $\lambda_j \in \{1 - \alpha_i - \beta_i\}$ for

$$i = 1, 2, \dots, n \text{ and } j = m, m-1, \dots$$

Since the machine is absolutely isolated, V must belong to an isolated transferable range of α_i . Suppose V belong in the range of α_ℓ so that the last symbol read is α_ℓ .

Then compute

$$\frac{V - \alpha_\ell}{\lambda_\ell} = a_{m-1} + \lambda_{m-1} (a_{m-2} + \lambda_{m-2} (a_{m-3} + \dots) \dots)$$

where $\lambda_\ell = 1 - \alpha_\ell - \beta_\ell$.

Again $V - \alpha_\ell / \lambda_\ell$ must belong to a transferable range of α_i due to the property of absolute isolation. Therefore, the other $\alpha_{\ell-1}$ is found by computing $(V - \alpha_\ell / \lambda_\ell) - \alpha_{\ell-1} / \lambda_{\ell-1}$ and so on.

An equivalent corollary of Corollary 3.8 is given below.

Corollary 3.9:

If a PSM is the k^{th} isolated machine, the last $k+1$ symbols of an input string to the PSM are uniquely determined from the (1,2) element of the matrix product, $M_{m-k} \dots M_{m-1} M_m$.

Proof:

The proof is similar to that of Corollary 3.8 and may thus be omitted. The theorem given by Yasui, Yajima and Tan which describes $(k+1)$ definite events (or languages) is stated here in the sense of the absolutely isolated probabilistic automaton (PA) similar to theirs. A PA is defined as follows.

Definition 3.8:

A probabilistic automaton (PA) is a system composed of four elements, $PA = (S, M, \pi_i, F)$, over the alphabets x , where $S = \{s_1, s_2, \dots, s_n\}$ is a finite nonempty set of states, M is a mapping of $x \in X$ into the set of $n \times n$ stochastic matrices, π_i is the initial state distribution, and F is a nonempty subset of S (the subset is the set of designated final states).

An event (language) accepted by a PA is defined as follows.

Definition 3.9:

Let η be a real number, $0 \leq \eta \leq 1$, and $\epsilon > 0$ be an arbitrary small number. Then a set of input strings for a PA is defined by

$$L(PA, \eta, \vec{\epsilon}) = \{x | x \in X^*, |\Pi_i M(x) \Pi_f - \eta| \leq (\underline{\epsilon}, \bar{\epsilon})\} \quad (3.28)$$

where Π_f is an n -dimensional column vector whose j^{th} component equals 1 if $s_j \in F$ and 0 otherwise. The event (language) denoted by $L(PA, \eta, \vec{\epsilon})$ is the one accepted by a PA with cut point η and error $\vec{\epsilon} = (\underline{\epsilon}, \bar{\epsilon})$. $\underline{\epsilon}$ and $\bar{\epsilon}$ were defined in Corollary 3.7 with the k^{th} approximation.

By defining

$$S = \{s_1, s_2\}, X = \{x_1, x_2, \dots, x_n\}, M = \{M(x_j)\}$$

for $j = 1, 2, \dots, n$, $F = \{s_2\}$, $\Pi_i = (1, 0)$, $\Pi_f = (0, 1)$

and $\Lambda = \{\Pi_i M(x) \Pi_f\}$ for $x \in X^*$ in a PA, the properties of a two-state PSM derived in the previous sections may be applicable to a two-state PA.

Theorem 3.10:

If a PA has a set of stochastic matrices which is absolutely isolated by the k^{th} approximation, then

$$L(PA, \eta, \vec{\epsilon}) = Z \cup \{X^*x\} \quad (3.29)$$

where Z is a finite set of input strings of lengths less than $k+1$, x is any input string of length $k+1$ for $k \leq \ell$, X^* in X^*x is any pre-fixed input string,

$\eta = \gamma_i^{(k)}$ and $\epsilon = (\underline{\gamma}^{(k)}, \bar{\gamma}^{(k)})$. Thus, the PA accepts $(k+1)$ definite events:

(languages) for $k=0, 1, 2, \dots, \ell$, where η and ϵ are found with the k^{th} approximation.

Proof:

The proof is obvious from Definition 3.7, Theorem 3.6 and Corollary 3.7 since the truncation error for any input string $\epsilon\{X^*x\}$ is bounded and the state after processing any input string ϵZ is isolated. □

The absolutely isolated PA has finer resolution of error bound than the

completely isolated PA does.

Independence of the Initial State Distribution

We should point out another important property of the absolutely isolated machine. The initial state distribution has always been assumed as $(1,0)$, or $s_1 = 1$ and $s_2 = 0$. When this assumption is relaxed, $s_1 = 1-s$ and $s_2 = s$ where $0 < s \leq 1$, the absolutely isolated machine still has the same property with the initial state distribution $(1,0)$, if the machine is processing an infinite length of input string.

Theorem 3.11:

If a two state PSM is absolutely isolated with the starting initial state distribution of $(1,0)$, then the behavior of a PSM with a starting initial state distribution of $(1-s,s)$ where $0 < s \leq 1$ is the same as the machine behavior with the initial distribution $(1,0)$, when the PSM is processing an input string with an infinite length.

Proof:

After processing m input symbols, the product of the stochastic matrices of state transition associated with individual input symbols in the m length becomes the following matrix from Theorem 3.1:

$$\begin{pmatrix} 1-a_1' & a_1' \\ b_1' & 1-b_1' \end{pmatrix}$$

where a_1' and b_1' are expressed in (3.4) and (3.5), respectively.

Compute the present state distribution from a given initial state distribution $(1-s,s)$


$$(1-s,s) \begin{pmatrix} 1-a_1' & a_1' \\ b_1' & 1-b_1' \end{pmatrix} = (1-a_1'-s\lambda_1', a_1'+s\lambda_1') \quad (3.30)$$

where $\lambda_1^i = 1 - a_1^i - b_1^i$.

From Lemma 3.2 $\lambda_1^i = \lambda_m \lambda_{m-1} \lambda_{m-2} \dots \lambda_2 \lambda_1$,

λ_1^i becomes zero as $m \rightarrow \infty$, since

$$|\lambda_i| < 1 \quad \text{for } i = 1, 2, 3, \dots$$


Therefore, the present state distribution becomes $(1 - a_1^i, a_1^i)$, which is the same as the starting state distribution of $(1, 0)$. 

From Theorem 3.11, we have the following corollary.

Corollary 3.12:

An absolutely isolated two-state PSM becomes an initial state independent machine as a processing length of input symbols approaches infinity.

Proof:

It is evident from the proof of Theorem 3.11. 

When we are interested in the k^{th} approximation of the finite series S with the initial state distribution $(1-s, s)$ where $0 < s \leq 1$ instead of $(1, 0)$, Lemma 3.3 should be modified by adding the new term, $s \lambda_m \lambda_{m-1} \dots \lambda_{m-k+1}$, at the end of the series as seen in (3.30). We have the following lemma.

Lemma 3.11:

$$\text{Given } S_{k+1} = a_m + \lambda a_{m-1} + \lambda \lambda_{m-1} a_{m-2} + \dots + \prod_{\ell=0}^k \lambda_{m-\ell} a_{m-k-1} + s \prod_{\ell=0}^k \lambda_{m-\ell},$$

then the upper bounds denoted as S_{\max}^i of the term S of S_{k+1} are the following:

Case 1: If $\lambda_a > \lambda_c > 0$, then

$$S_{\max}^i = a \frac{1 - \lambda_a^k}{1 - \lambda_a} + s \lambda_a^{k-1}$$

Case 2: If $\lambda_a > 0 \geq \lambda_c$ and $|\lambda_a| \geq |\lambda_c|$, then

$$S'_{\max} = a \frac{1-\lambda_a^k}{1-\lambda_a} + s\lambda_a^{k-1}$$

Case 3: If $\lambda_a > 0 > \lambda_c$ and $|\lambda_a| \leq |\lambda_c|$, then

$$S'_{\max} = a(1+\lambda_a) \frac{1-\lambda_c^{k-1}}{1-\lambda_c^2} + (a+s)\lambda_c^{k-1} \quad k \geq 3, \quad k = \text{odd}$$

$$S'_{\max} = a(1+\lambda_a) \frac{1-\lambda_c^k}{1-\lambda_c^2} + s\lambda_a\lambda_c^{k-2} \quad k \geq 2, \quad k = \text{even}$$

Case 4: If $0 > \lambda_a > \lambda_c$, then

$$S'_{\max} = a \frac{1-\lambda_c^{k+1}}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_a^{k-1}}{1-\lambda_a^2} + s\lambda_c^{k-1} \quad k \geq 3, \quad k = \text{odd}$$

$$S'_{\max} = a \frac{1-\lambda_c^k}{1-\lambda_c^2} + c\lambda_a \frac{1-\lambda_a^k}{1-\lambda_a^2} + s\lambda_a^{k-1} \quad k \geq 2, \quad k = \text{even}$$

Proof:

It is similar to the proof of Lemma 3.3a; therefore, the proof is omitted. □

Definition 3.10:

A new upperbound denoted as \bar{S}'_{\max} of the four S'_{\max} 's in Lemma 3.11 is defined as

$$\bar{S}'_{\max} = \max\{\bar{S}_{\max}, S'_{\max}\}$$

where $\bar{S}_{\max} = 1-\lambda_{\max}^k / 1-\lambda_{\max}$ defined in Lemma 3.8.

By substituting S'_{\max} and \bar{S}'_{\max} into S_{\max} and \bar{S}_{\max} , respectively, into Lemma 3.9, Lemma 3.10, Theorem 3.6, Corollary 3.7, Theorem 3.8, Corollary 3.9 and Theorem 3.10, the results are valid. The new proofs are very much the same as the proofs of the previously mentioned lemmas, theorems and corollaries.

A property of the absolutely isolated machine is that all past inputs are traceable; therefore, the history of state transition and all output produced can be determined. The past history may help to find an error which occurred in past. Although a past error may change the character of the machine, or probabilities in the state transition matrix, an input-traceable machine is called a diagnoseable machine as long as the absolutely isolated property exists.

CHAPTER IV

DECOMPOSITION OF A PROBABILISTIC SEQUENTIAL MACHINE

4. Introduction

We studied the two-state PSM in the previous chapter. Two-state machines are too restrictive for practical applications because of the small number of states. We may need an n -state machine for practical applications, where $n > 2$; however, an n -state machine can be composed of interconnecting component machines, which have less states than n . This problem is known as decomposition of an n -state PSM. Therefore, in this chapter, we present a PSM decomposition technique which enables us to decompose an n -state PSM into interconnected, two-state PSM machines.

Bacon [4] and Paz [51] introduced the "lumpability" and the "separability" properties of a PSM whereby an n -state PSM could be decomposed into two PSM's, which have states less than n , if the n -state PSM has these two properties. The decomposition of a PSM deals with the problems of how a sequential machine can be realized from sets of smaller component machines, how these component machines are to be interconnected, and how information flows in and among these machines when they operate. In this chapter, we show a different decomposition scheme of an n -state PSM from those of Bacon and Paz.

At first, we discuss an interconnection property of component machines (PSM's), which can compose a large system function. Next, a composition example for interconnecting two-state PSM's is discussed as a prefacing step to the decomposition of a three-state PSM by two-state component PSM's. A decomposition process for the three-state PSM composed of three two-state PSM's is presented in Section 4.4. The process can be extended to an n -state PSM decomposition by $n(n-1)/2$ two-state component PSM's. A computation procedure

of state transitions of the n-state PSM by the matrix product of two-state stochastic matrices (by two-state component PSM's) is described in the last section.

4.1 An Interconnecting Property of N Two-State Machines

Consider an interconnection of n two-state machines, which are numbered 1,2,...,n, the states on each machine are denoted as $M_1 = (s_{11}, s_{12})$, $M_2 = (s_{21}, s_{22})$, ..., $M_n = (s_{n1}, s_{n2})$. The first index of s_{ni} refers to the machine number, and the second indicates the state number. When the interconnection is viewed as a single machine, a typical transition probability which presents a possibility of the transition between two states in the single machine would be

$$P(s_1^i, s_2^i, \dots, s_n^i / s_1, s_2, \dots, s_n, x) \quad (4.1)$$

where s_i and s_i^i are the state vectors before and after the transition of the ith machine respectively and x is any input. This may be rewritten as

$$\begin{aligned} & p(s_1^i / s_2^i, s_3^i, \dots, s_n^i, s_1, s_2, \dots, s_n, x) \\ & \cdot p(s_2^i / s_3^i, \dots, s_n^i, s_1, s_2, \dots, s_n, x) \\ & \cdot p(s_3^i / s_4^i, \dots, s_n^i, s_1, s_2, \dots, s_n, x) \\ & \dots \dots \dots p(s_n^i / s_1, s_2, \dots, s_n, x) . \end{aligned}$$

From the knowledge of interconnection of component machines, which describes this transition probability, the probability of a particular next state of any component machine can be determined only with the present state of the system and the present input. Also, it is assumed that once this input to any component machine in the interconnection is specified, the transition of the machine is independent of the transition of the other interconnecting components.

In other words, the transition probabilities of the component machines specify the transition probabilities of the interconnection through the assumption that given its input, each component operates independently of the others. Thus

$$p(s_i^1/s_{i+1}^1, s_{i+2}^1, \dots, s_n^1, s_1, s_2, s_3, \dots, s_n, x) \\ = p(s_i^1/s_1, s_2, s_3, \dots, s_n, x) \quad \text{for } i = 1, 2, \dots, n-1.$$

Then (4.1) may be rewritten as

$$p(s_1^1/s_1, s_2, \dots, s_n, x) p(s_2^1/s_1, s_2, \dots, s_n, x) \\ \cdot p(s_3^1/s_1, s_2, \dots, s_n, x) \dots p(s_n^1/s_1, s_2, \dots, s_n, x).$$

For a given n -state machine, this factorization may be possible by an interconnection of component machines; which is isomorphic to the machine. Namely, the input, output, and state sets of the two machines are respectively isomorphic, and the transition probabilities for states corresponding with the isomorphism are equal. Also, when each component machine has fewer states than n , the factorization is called a "decomposition" of the n -state machine.

We shall consider a decomposition which is loop-free; there are no closed paths of information flow. All delay elements and logical elements in a machine can be so arranged that the information flow in the machine propagates only in one direction from input terminal to output terminal.

A basic property of a loop-free decomposition is that some of the components operate independently of all the others. The component machines receive only the input to the original machine. Regardless of the states of all the other component machines, the transition probability to any next state for this machine depends only upon the present state of this component machine and the input to the original machine (or the interconnection).

Consider an example of decomposition by Bacon [4]. Let M be a four-state two-input machine as specified below.

$$M(x_1) = \begin{pmatrix} .2 & .2 & .3 & .3 \\ .0 & .4 & .0 & .6 \\ .4 & .1 & .4 & .1 \\ .25 & .25 & .25 & .25 \end{pmatrix} \quad M(x_2) = \begin{pmatrix} .3 & .0 & .7 & .0 \\ .0 & .3 & .0 & .7 \\ .08 & .12 & .32 & .48 \\ .06 & .14 & .24 & .56 \end{pmatrix}$$

It is easy to find a block partition of states $\Pi = (\Pi_1, \Pi_2) = (\overline{1,2}, \overline{3,4})$ and another partition $\tau = (\tau_1, \tau_2) = (\overline{1,3}, \overline{2,4})$. Also, it is simple to check out the set, $\{\Pi_1 \Lambda \tau_1, \Pi_1 \Lambda \tau_2, \Pi_2 \Lambda \tau_1, \Pi_2 \Lambda \tau_2\} = \{1,2,3,4\}$ where Λ is the intersection operation. Thus by lumping Π_1 and Π_2 , we have

$$M_{\Pi}(x_1) = \begin{pmatrix} .4 & .6 \\ .5 & .5 \end{pmatrix}, \quad M_{\Pi}(x_2) = \begin{pmatrix} .3 & .7 \\ .2 & .8 \end{pmatrix}.$$

Summing the columns of $M(x_i)$, $i = 1, 2$ on τ_1 and τ_2 , we obtain the matrices

$$M^{\tau}(x_1) = \begin{pmatrix} .5 & .5 \\ .0 & 1.0 \\ .8 & .2 \\ .5 & .5 \end{pmatrix}, \quad M^{\tau}(x_2) = \begin{pmatrix} 1.0 & .0 \\ .0 & 1.0 \\ .4 & .6 \\ .3 & .7 \end{pmatrix}.$$

In this example, each block of Π intersects each block of τ in one and only one state and the transition matrices of M_{τ} are uniquely specified as

$$M_{\tau}(\Pi_1, x_1) = \begin{pmatrix} .5 & .5 \\ .0 & 1.0 \end{pmatrix}, \quad M_{\tau}(\Pi_1, x_2) = \begin{pmatrix} 1.0 & .0 \\ .0 & 1.0 \end{pmatrix},$$

$$M_{\tau}(\Pi_2, x_1) = \begin{pmatrix} .8 & .2 \\ .5 & .5 \end{pmatrix}, \quad M_{\tau}(\Pi_2, x_2) = \begin{pmatrix} .4 & .6 \\ .3 & .7 \end{pmatrix}.$$

Therefore, the two machines are interconnected as shown below.

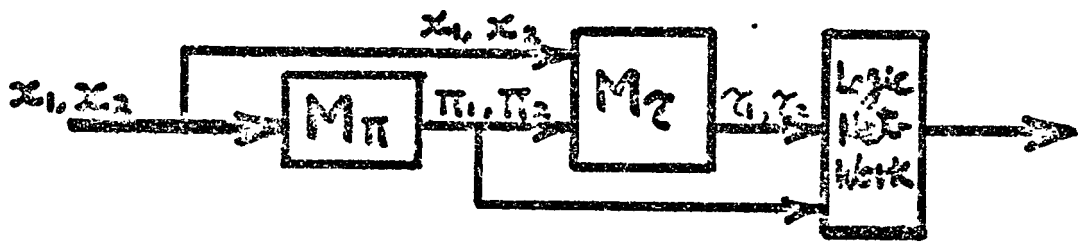


Figure 4.1
A Decomposition of a PSM

The original state transition is computed by multiplying the two states of the two machines.

There are many ways to interconnect component machines. The following figure explains several interconnections.

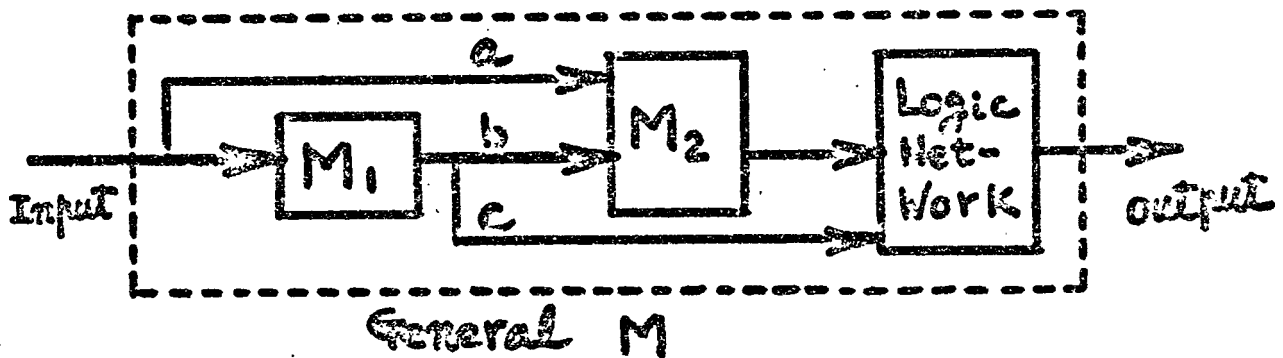


Figure 4.2
Interconnection of Component Machines M_1 and M_2 for Machine M

The types of decomposition for the various connections are as follows.

- (a) Loop free serial-parallel decomposition, if connections a, b , and c exist.
- (b) Parallel decomposition, if connections a and c exist.

- (c) Quasi-parallel decomposition, if connections b and c exist.
- (d) Quasi-serial decomposition, if connections a and b exist.
- (e) Serial decomposition, if only connection b exists.

4.2 Composition of Two-State PSM's

Theorems for decomposition by Bacon and Paz are based on "lumpability" and "separability" properties of a given PSM. Paz [51] also showed that an n-state PSM can be decomposed by two smaller state PSM's; one has two states and the other has n-1 states.

Decomposition represented here is different from those of Bacon and Paz. A decomposition of an n-state PSM by $n(n-1)/2$ two-state PSM's is studied in this research. To explain the decomposition of the n-state machine, we start with a composition (synthesis) by component machines.

Consider a composition of a four-symbol/four-state machine by two, two-symbol/two-state machines. Let the two, two-symbol/two-state machines be

$$M_1(x_1) = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \quad M_1(x_2) = \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} \quad (4.2-a)$$

$$M_2(x_1) = \begin{pmatrix} 1-e & e \\ f & 1-f \end{pmatrix}, \quad M_2(x_2) = \begin{pmatrix} 1-g & g \\ h & 1-h \end{pmatrix} \quad (4.2-b)$$

Consider a pair of two symbols for M_1 and M_2 . The four symbols, (x_1^1) , (x_1^2) , (x_2^1) and (x_2^2) , are defined as input to the four-state machine. Similarly, the states of the four-state machine are defined in vector form as (s_{11}, s_{21}) , (s_{11}, s_{22}) , (s_{12}, s_{21}) and (s_{12}, s_{22}) , which shall be abbreviated in $(1,1)$, $(1,2)$, $(2,1)$ and $(2,2)$, respectively. The first index of s_{ij} refers to the machine number, and the second indicates the state number.

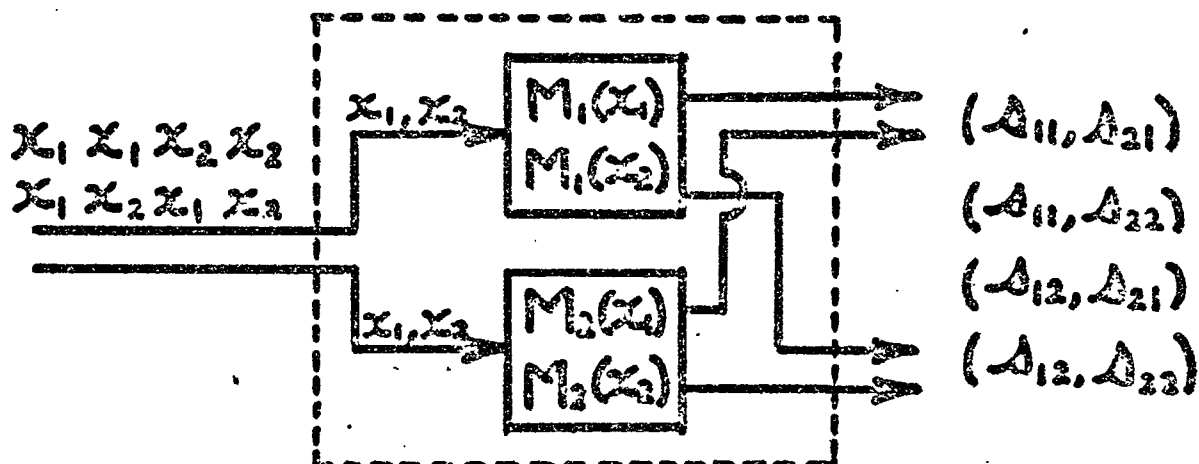


Figure 4.3

Four-Symbol/Four-State Machine Composed of Two, Two-Symbol/Two-State Machines

The state transition matrix of the four-state machine for the first symbol is

$$M_{12}^{x_1} = \begin{matrix} \text{STATE ID} \\ (1,1) \\ (1,2) \\ (2,1) \\ (2,2) \end{matrix} \begin{pmatrix} (1,1) & (1,2) & (2,1) & (2,2) \\ (1-a)(1-e) & (1-a)e & a(1-e) & ae \\ (1-a)f & (1-a)(1-f) & af & a(1-f) \\ b(1-e) & be & (1-b)(1-e) & (1-b)e \\ bf & b(1-f) & (1-b)f & (1-b)(1-f) \end{pmatrix} \quad (4.3)$$

Each element in the matrix is constructed as the product of two probabilities, $p_{ij}^1(x_m)$, $p_{kl}^2(x_m)$, where the subscript indicates the transition probability between two states, i to j and k to l , respectively. The superscript is the machine number. Equation (4.3) is known as the Kronecker product of $M_1(x_1)$ and $M_2(x_1)$.

$$M_1(x_1) \otimes M_2(x_1) = \begin{pmatrix} (1-a)M_2(x_1) & a M_2(x_1) \\ b M_2(x_1) & (1-b)M_2(x_1) \end{pmatrix} \quad (4.4-a)$$

The state transition matrices for the other three symbols can be derived simi-

larly.

$$M_{12} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = M_1(x_1) \otimes M_2(x_2) \quad (4.4-b)$$

$$M_{12} \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} = M_1(x_2) \otimes M_2(x_1) \quad (4.4-c)$$

and

$$M_{12} \begin{pmatrix} x_2 \\ x_2 \end{pmatrix} = M_1(x_2) \otimes M_2(x_2) \quad (4.4-d)$$

The discussion is a parallel decomposition. An example of this decomposition is shown in the following example.

Example 4.1:

$a = \frac{1}{10}$, $b = \frac{1}{3}$, $c = \frac{1}{4}$, $d = \frac{1}{5}$, $e = \frac{1}{6}$, $f = \frac{1}{8}$, $g = \frac{1}{2}$ and $h = \frac{1}{12}$, thus

$$M_{12}^{x_1} = \begin{pmatrix} \frac{9}{10} \frac{5}{6} & \frac{9}{10} \frac{1}{6} & \frac{1}{10} \frac{5}{6} & \frac{1}{10} \frac{1}{6} \\ \frac{9}{10} \frac{1}{8} & \frac{9}{10} \frac{7}{8} & \frac{1}{10} \frac{1}{8} & \frac{1}{10} \frac{7}{8} \\ \frac{1}{3} \frac{5}{6} & \frac{1}{3} \frac{1}{6} & \frac{2}{3} \frac{5}{6} & \frac{2}{3} \frac{1}{6} \\ \frac{1}{3} \frac{1}{8} & \frac{1}{3} \frac{7}{8} & \frac{2}{3} \frac{1}{8} & \frac{2}{3} \frac{7}{8} \end{pmatrix} \quad (4.5)$$

The numerical quantities can be substituted in (4.4-b), (4.4-c), and (4.4-d) for the other three symbols. The preceding composition example is the reverse of decomposition, how to find component machines for a given n -state machine where $n > 2$, is a decomposition problem, which is solved in this chapter.

4.3 A Decomposition by Two-State PSM's

Any n -state PSM can be represented by $n(n-1)$ parameters which are non-negative, but which are less than or equal to one. Each state of an n -state PSM is named s_1, s_2, \dots, s_n . Consider the first row in a state transition matrix of the n -state PSM,

$$M = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \quad (4.6)$$

a_{1i} is the transition probability from s_1 to s_i , $i = 1, 2, \dots, n$. Suppose we are interested in the transition from s_1 to s_1 , the rest may be regarded as a pseudo-state. Assign $a_{11} = 1 - a_1$, where a_1 is the probability of state transition from s_1 to the rest, $0 \leq a_1 \leq 1$. By focusing the state transition from s_1 to s_2 within the probability a_1 , the remaining states, s_3, s_4, \dots, s_n become another pseudo-state so that $a_{12} = a_1(1 - a_2)$, where a_2 is the probability of state transition to the remaining states, $0 \leq a_2 \leq 1$, and so on.

This process can be repeated for the other rows; thus, M of (4.6) can be rewritten as

$$M = \begin{pmatrix} 1 - a_1 & a_1(1 - a_2) & a_1 a_2(1 - a_3) & \dots & a_1 a_2 \dots a_n \\ b_1 b_2 \dots b_n & 1 - b_1 & b_1(1 - b_2) & b_1 b_2(1 - b_3) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_1(1 - c_2) & c_1 c_2(1 - c_3) & \dots & c_1 c_2 \dots c_n & 1 - c_1 \end{pmatrix} \quad (4.7)$$

Each row has $n-1$ parameters, and the total number of the parameters is $n(\text{rows}) \times (n-1)$ (per each row). The domain of each parameter is $[0, 1]$. We state it in

the next theorem.

Theorem 4.1:

All n -state PSM's exist in unit volume in the $n(n-1)$ dimensional Euclidean space.

Proof:


By mathematical induction, when $n=2$,

$$\begin{pmatrix} 1-a_1 & a_1 \\ b_1 & 1-b_1 \end{pmatrix}$$

these are $2 \times 1 = 2$ parameters, a_1 and b_1 .

Suppose $n=m$,

$$M = \begin{pmatrix} 1-a_1 & a_1(1-a_2) & \dots & a_1 a_2 \dots a_{m-1} \\ b_1 b_2 \dots b_{m-1} & 1-b_1 & \dots & b_1 b_2 \dots b_{m-2} (1-b_{m-1}) \\ \vdots & & & \\ c_1(1-c_2) & \dots & c_1 c_2 \dots c_{m-1} & 1-c_1 \end{pmatrix} \quad (4.9)$$

There are m rows, and each row has $m-1$ parameters so that the total number is $m(m-1)$. Consider $n = m+1$. By increasing one column at the right of the last column, circulating each term below the diagonal terms to right in order to fill the new column, splitting the term $z = \prod_{i=1}^{m-1} z_i$ into $z(1-z_m)$ and zz_m , where z_i and z_m are a_i, b_i, \dots or c_i , and a_m, b_m, \dots or c_m , respectively, inserting the term zz_m into the left vacant entry of each diagonal term and adding one row below the last row, we obtain $m+1$ rows, and each row now has m parameters. The total number is $(m+1)m$. 

When we consider the interconnection of two-state PSM's, each PSM exists

in a unit volume in two dimensions. It is convenient to use a binary number system representation for each PSM, such as an m -binary digit represents m two-state PSM's and digits 0 and 1 correspond to the first and second states of each PSM, respectively.

Numbers which are represented by m -binary digits are 0 through to $2^m - 1$; the number of states constructed by m interconnected PSM's is 2^m . For example, when $m=3$, the number of states is eight. Consider the binary number 010 or (010) state in three interconnected PSM's; the first, the second, and the third PSM's are in state s_1 , state s_2 , and state s_1 , respectively.

Note: The space defined by m -binary digits is in a unit volume of 2^m dimensional space. We shall distinguish between the m -binary dimension (denoted as m B-vector) from the unit volume in 2^m dimensional space (denoted as B-unit volume). Before we discuss decomposition, the following definitions are needed.

Definition 4.1:

A node of a B-unit volume is a point represented by a coordinate (x_1, x_2, \dots, x_m) , where $x_i \in \{0, 1\}$ for $i = 1, 2, \dots, m$.

Definition 4.2:

The distance between a pair of two nodes in a B-unit volume is defined by the difference of the number of "1" bits in their two coordinates.

For example, the distance between (000) and (001) is one; the distance between (010) and (110) is also one, while (000) and (111) are separated by the distance three. This distance is known as the Hamming distance.

Definition 4.3:

A pair of nodes which have the distance one are called adjacent nodes.

Definition 4.4:

An edge is a line between adjacent nodes on a B-unit volume.

Definition 4.5:

The direction of an edge is the coordinate direction where the distance one takes place.

Definition 4.6:

If the direction of two edges is orthogonal, the two edges are called orthogonal edges.

Definition 4.7:

A set of edges is a block (or a subset) of states s , and a partition of states is a collection of subsets of s such that each state in s belongs to one and only one such subset.

Definition 4.8:


If an edge in a block is orthogonal to an edge in another block, these blocks are called orthogonal blocks. If all blocks in states s are orthogonal to each other then the partitions are called independent partitions.

Using the above definitions, we show that eight states which are constructed with three two-state PSM's are blocked into three orthogonal blocks. Since the three two-state PSM's are composed of three unit B-vector spaces (001), (010) and (100), and are enclosed by eight nodes, there are three different directions of edges. The total number of independent blocks of three two-state PSM's is given in the next theorem. We shall present only the decomposition of a three-state PSM; however, the decomposition method can be extended to n -state PSM's, $n > 3$.

Theorem 4.2:

The total number of independent partitions of the three states of a three-state PSM composed of three two-state PSM's is $4 \times 2 \times 3 = 24$.

Proof:

Since eight states (whose binary representation is 0 through 2^3-1) consisting of three two-state PSM's are enclosed by a unit volume in three unit B-vector spaces, there are three different directions of edges, and there are four edges in each direction (see Figure 4.5). The first block of two states is chosen in four ways in any one of four different edges in a direction. There is a surface plan of the unit volume which is parallel to the chosen edge. The second one of two states is selected in two ways in one on the edges of the parallel plan in the remaining two directions. The third block of two states assumes the third direction edge which connects the parallel plan to the other plan containing the first edge (block). The fourth pair of two states, which has not been shared in any direction, may belong in any one of three directions, so that the total number of independent partitions is $4 \times 2 \times 3 = 24$. 

Before showing a PSM decomposition, it is necessary to present the decomposition of a three-state stochastic matrix. We introduce pseudoprobability for our convenient.

Definition 4.9:

Pseudoprobability x exists in the extended domain $-\infty < x < 0$ or $1 < x < \infty$. A probability x belonging in the extended domain or in $0 \leq x \leq 1$ is called extended probability. Similarly, we have an extended stochastic matrix as long as the sum of the extended probabilities in each row is one.

Definition 4.9 provides a closed domain, $-\infty < x < \infty$, for an inverse matrix

operation on an extended stochastic matrix.

Lemma 4.1:

The domain of the entries in the inverse matrix of an extended stochastic matrix is the same domain of the entries in an extended stochastic matrix, if the eigenvalue of the original extended stochastic matrix is not zero.

Proof:

Let $M = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}$ be an extended stochastic matrix, where $-\infty < a, b < \infty$, $1-a-b \neq 0$.

$$\begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}^{-1} = \begin{pmatrix} 1-b/1-a-b & -a/1-a-b \\ -b/1-a-b & 1-a/1-a-b \end{pmatrix} = \begin{pmatrix} 1-a' & a' \\ b' & 1-b' \end{pmatrix}$$

a' and b' are the extended probabilities, and the sum of each row is one. 

Using the extended probability, the decomposability of a three-state matrix is stated in the next theorem.


Theorem 4.3:

If all three rows of a three-state matrix are linearly independent, then a partition of states composed of three two-state extended stochastic matrices, which is isomorphic to the original three-state matrix, must be independent.

Proof:

Suppose the contrary is true; i.e., the partition is not independent. Then there is at least one pair of two blocks which are not orthogonal to each other from Definition 4.8, and only one bit in the binary number representation of edges in the two blocks is different. This means that a two-state matrix represented by two unchanged bits becomes a common factoring matrix to the

transition matrices of these two blocks.

Since the third block in the partition is the complement of the first two states, or $1-s_1-s_2$, where s_1 and s_2 are the first and second blocks, respectively, if any two blocks have a common factoring matrix, then the partition matrix (all three blocks) has the common factoring matrix. From the existence of the common factoring matrix, the given three-state matrix must have a common factoring matrix; in order for this matrix to be isomorphic to three two-state extended matrices. Namely, all three rows are not linearly independent. This is a contradiction. 

The decomposition of a three-state stochastic matrix by two-state stochastic matrices is best explained by the following examples.

Example:

Three two-state stochastic matrices are

$$M_1(x_1) = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \quad M_2(x_1) = \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} \quad \text{and} \quad M_3(x_1) = \begin{pmatrix} 1-e & e \\ d & 1-d \end{pmatrix} \quad (4.10)$$

The matrix composed of three two-state matrices, $M_1(x_1)M_2(x_1)M_3(x_1)$, in three binary digit representation, is found as follows. A partition is assumed as $\{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$, where the first block, second block, and third block are composed of the states 0 and 4; 1 and 3; 2,5,6 and 7; respectively. The transition probability $p_{ij}(x_1)$ from the i^{th} state to the j^{th} state in the composed matrix is found as follows:

$$p_{ij}(x_1) = p_{i_1j_1}^1(x_1) \cdot p_{i_2j_2}^2(x_1) \cdot p_{i_3j_3}^3(x_1),$$

where the superscript is the machine number, the subscript indicates the state transition between, and $i = i_1i_2i_3$, $j = j_1j_2j_3$.

For example, $i=0$ and $j=1$; the state transition $000 \rightarrow 001$, namely $i_1=0$, $j_1=0$ in M_1 ; $i_2=0$, $j_2=0$ in M_2 ; $i_3=0$, $j_3=1$ in M_3 ; thus $p_{00}^1(x_1) = (1-a)$, $p_{00}^2(x_1) = (1-c)$, $p_{01}^3(x_1) = e$ and $p_{01}(x_1) = (1-a)(1-c)e$ which is located at the first row and the third column in (4.11).

	$\overline{0,4}$		$\overline{1,3}$	
$\overline{0,4}$	$(1-a)(1-c)(1-e)$	$a(1-c)(1-e)$	$(1-a)(1-c)e$	$(1-a)ce$
	$b(1-c)(1-e)$	$(1-b)(1-c)(1-e)$	$b(1-c)e$	bce
$\overline{1,3}$	$(1-a)(1-c)f$	$a(1-c)f$	$(1-a)(1-c)(1-f)$	$(1-a)c(1-f)$
	$(1-a)df$	adf	$(1-a)d(1-f)$	$(1-a)(1-d)(1-f)$

	$(1-a)d(1-e)$	$ad(1-e)$	$(1-a)de$	$(1-a)(1-d)e$
$\overline{2,5}$	$bd(1-e)$	$(1-b)d(1-e)$	bde	$b(1-d)e$
$\overline{6,7}$	$b(1-c)f$	$(1-b)(1-c)f$	$b(1-c)(1-f)$	$bc(1-f)$
	bdf	$(1-b)df$	$bd(1-f)$	$b(1-d)(1-f)$
	$\overline{2,5,6,7}$			
$\overline{04}$	$(1-a)c(1-e)$	$ac(1-e)$	$a(1-c)e$	ace
	$bc(1-e)$	$(1-b)c(1-e)$	$(1-b)(1-c)e$	$(1-b)ce$
$\overline{1,3}$	$(1-a)cf$	acf	$a(1-c)(1-f)$	$ac(1-f)$
	$(1-a)(1-d)f$	$a(1-d)f$	$ad(1-f)$	$a(1-d)(1-f)$

$\overline{2,5}$	$(1-a)(1-d)(1-e)$	$a(1-d)(1-e)$	ade	$a(1-d)e$
$\overline{6,7}$	$b(1-d)(1-e)$	$(1-b)(1-d)(1-e)$	$(1-b)de$	$(1-b)(1-d)e$
	bcf	$(1-b)cf$	$(1-b)(1-c)(1-f)$	$(1-b)c(1-f)$
	$b(1-d)f$	$(1-b)(1-d)f$	$(1-b)d(1-f)$	$(1-b)(1-d)(1-f)$

An independent partition $\{b_1, b_2, b_3\} = \{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$ is used in (4.11) and is depicted in Figure 4.5.

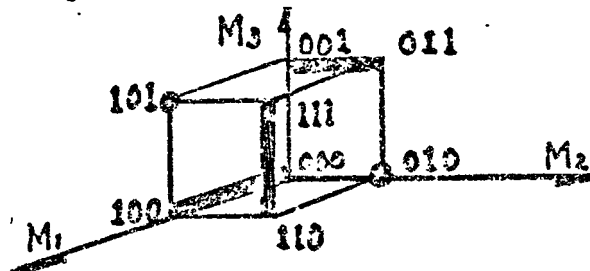


Figure 4.5 A Partition $\{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$

It is easy to verify that the partition is independent. Let a given three-state stochastic matrix be

$$M(x_1) = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{pmatrix} 1-\alpha & \alpha(1-r) & \alpha r \\ \beta s & 1-\beta & \beta(1-s) \\ \gamma(1-t) & \gamma t & 1-\gamma \end{pmatrix} \end{matrix} \quad (4.12)$$

Let the first block, $\overline{0,4}$, the second block $\overline{1,3}$ and the third block $\overline{2,5,6,7}$ be s_1 , s_2 and s_3 , respectively. Unknown parameters a, b, c, d, e and f in (4.10) are found by equating the first two columns in (4.11) with the first two columns in (4.12), respectively.

State transitions are found as

$$\begin{aligned} s_1 \rightarrow s_1: (1-c)(1-e) &= 1-\alpha, & s_1 \rightarrow s_2: (1-a)e + be &= 2\alpha(1-r) \\ s_2 \rightarrow s_1: (1-c)f + df &= 2\beta s, & s_2 \rightarrow s_2: (1-a)(1-f) &= 1-\beta \\ s_3 \rightarrow s_1: 2d(1-e) + (1-c+d)f &= 4\gamma(1-t), & s_3 \rightarrow s_2: 2b(1-f) + (1-a+b)e &= 4\gamma t. \end{aligned} \quad (4.13)$$

For simplicity, we shall use the following conventions:

Pivot states denoted by p_1 and p_2 are the diagonal term transition in (4.13); $s_1 \rightarrow s_1$, $s_2 \rightarrow s_2$.

Pivot matrix denoted by $\begin{pmatrix} 1-p_{11} & p_{12} \\ p_{21} & 1-p_{22} \end{pmatrix}$ is a 2×2 matrix of which the diagonal terms are the pivot states.

Note: Pivot matrix is a principal minor or a matrix with a rearranged order of rows/columns of the principal minor. In (4.13), $p_1 = s_1$ and $p_2 = s_2$ and $p_{11} = \alpha$, $p_{12} = 2\alpha(1-r)$, $p_{21} = 2\beta s$ and $p_{22} = \beta$.

Each column element of the third row in (4.13) which is called the off-pivot state is containing the off-diagonal term of the same column of the pivot

matrix and can be rewritten by substituting the quantity of the term in the element. Therefore, the remaining parts of the column elements of the off-pivot state denoted by q_1 and q_2 are the probability difference of the state transitions from the off-pivot state (1) to the pivot state and (2) to the off-diagonal state of the pivot matrix in the same column. In (4.13), the probability difference q_1 in the first column is the probability of the state transition $s_3 \rightarrow s_1$ (s_1 is the pivot state) minus the probability of the state transition $s_3 \rightarrow s_2$ (s_2 is the off-diagonal state) or $q_1 = 4\gamma(1-t) - 2\beta s$. Similarly q_2 in the second column is the probability of the state transition $s_3 \rightarrow s_2$ (s_2 is the pivot state) minus the probability of the state transition $s_3 \rightarrow s_1$ (s_1 is the off-diagonal state) or $q_2 = 4\gamma t - 2\alpha(1-r)$. p_{ij} and q_i for $i, j = 1, 2$ shall be called p-q parameters.

From (4.13), we have six unknowns and six equations.

Note: The multiplication factors 4 and 2 in the above q_1 and q_2 equations come from the sums of two rows in the second block $\overline{1,3}$ and four rows in the third block $\overline{2,5,6,7}$, respectively, by assuming equal distribution among states in each block. The equations (4.13) can be rewritten as follows:

$$\begin{aligned}
 c(1-e) + e &= \alpha = p_{11}, & (1-a+b)e &= 2\alpha(1-r) = p_{12}, \\
 (1-c+d)f &= 2\beta s = p_{21}, & a(1-f) + f &= \beta = p_{22}, \\
 2d(1-e) &= 4\gamma(1-t) - 2\beta s = q_1, & 2b(1-f) &= 4\gamma t - 2\alpha(1-r) = q_2.
 \end{aligned} \tag{4.14}$$

The unknown quantities, a, b, c and d are found to be functions of e and f from (4.14),

$$a = \frac{p_{22} - f}{1 - f}, \tag{4.15}$$

$$b = \frac{q_2}{2(1-f)} , \quad (4.16)$$

$$c = \frac{p_{11}-e}{1-e} , \quad (4.17)$$

$$d = \frac{q_1}{2(1-e)} . \quad (4.18)$$

By substituting a, b and c, d into the remaining equations in (4.14), respectively,

$$\{1 - \frac{p_{11}-e}{1-e} + \frac{q_1}{2(1-e)}\}f = p_{21} \quad (4.19)$$

and

$$\{1 - \frac{p_{22}-f}{1-f} + \frac{q_2}{2(1-f)}\}e = p_{12} \quad (4.20)$$

By solving (4.19) and (4.20) for e and f,

$$\begin{aligned} \{2(1-e) - 2(p_{11}-e) + q_1\}f &= 2p_{21}(1-e) \\ f &= \frac{2p_{21}(1-e)}{2(1-p_{11})+q_1} \end{aligned} \quad (4.21)$$

and

$$\begin{aligned} \{2(1-f) - 2(p_{22}-f) + q_2\}e &= 2p_{12}(1-f) \\ f &= \frac{1}{2p_{12}} [2p_{12} - \{2(1-p_{22}) + q_2\}e] . \end{aligned} \quad (4.22)$$

By eliminating f from (4.21) and (4.22),

$$\begin{aligned} 4p_{12}p_{21}(1-e) &= [2p_{12} - \{2(1-p_{22}) + q_2\}e] \{2(1-p_{11}) + q_1\} \\ [\{2(1-p_{11}) + q_1\} \{2(1-p_{22}) + q_2\} - 4p_{12}p_{21}]e &= 2p_{12} \{2(1-p_{11}) + q_1\} - 4p_{12}p_{21} \\ e &= \frac{2p_{12} \{2(1-p_{11}) + q_1\} - 4p_{12}p_{21}}{\{2(1-p_{11}) + q_1\} \{2(1-p_{22}) + q_2\} - 4p_{12}p_{21}} \end{aligned} \quad (4.23)$$

By back-substitution of the quantity e in (4.22), (4.18) and (4.17) and of the quantity f in (4.16) and (4.15), numerical quantities for a, b, c and d are determined.

Numerical Example 4.2

$$M(x_1) = \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.2 & 0.7 \end{pmatrix}, \quad (4.24)$$

find $\alpha, \beta, \gamma, r, s$ and t in (4.12) as follows:

$$\begin{aligned} \alpha &= 7/10, & r &= 1/7, \\ \beta &= 4/10, & s &= 1/2, \\ \gamma &= 3/10, & t &= 2/3. \end{aligned}$$

From (4.14),

$$\begin{aligned} p_{11} &= 7/10, & p_{12} &= 6/5, \\ p_{21} &= 2/5, & p_{22} &= 4/10, \\ q_1 &= 0, & q_2 &= -2/5. \end{aligned} \quad (2.25)$$

From (4.23), (4.22), (4.15), (4.16), (4.17) and (4.18),

$$\begin{aligned} e &= 1/3, & f &= 8/9, & a &= -44/10, & b &= -9/5, \\ c &= 11/20 & \text{and} & & d &= 0. \end{aligned} \quad (2.26)$$

$M_1(x_1)$ is an extended stochastic matrix because of $a, b < 0$;
 $M_2(x_1)$ and $M_3(x_1)$ are stochastic matrices.

In order for all decomposed two-state matrices to be strictly stochastic, we shall find conditions which are assured stochastics.

There are three ways to map the third block $b_3 = \overline{2,5,6,7}$ onto a state s_i for $i = 1, 2, 3$ along three axes.

1. On the mapping of the third block b_3 on s_2 and $b_1 \rightarrow s_1$, $b_2 \rightarrow s_3$, we have the pivot matrix $\begin{pmatrix} 1-\alpha & \alpha r \\ \gamma(1-t) & 1-\gamma \end{pmatrix}$. The q-parameters are

$$q_1 = 4\beta s - 2\gamma(1-t) \quad (4.27)$$

$$q_2 = 4\beta(1-s) - 2\alpha r.$$

2. On the mapping of the third block b_3 on s_1 and $b_1 \rightarrow s_3$, $b_2 \rightarrow s_2$, we have the pivot matrix $\begin{pmatrix} 1-\beta & \beta(1-s) \\ \gamma t & 1-\gamma \end{pmatrix}$. The q-parameters are

$$q_1 = 4\alpha(1-r) - 2\gamma t \quad (4.28)$$

$$q_2 = 4\alpha r - 2\beta(1-s)$$

There is at least one single set of positive q-parameters.

Lemma 4.2:

Let $\{b_1, b_2, b_3\}$ be an independent partition of eight states composed of three two-state PSM's. If the block, consisting of four states in the partition, is always selected as the off-pivot, then there exists at least one single set of the positive q-parameters along the three-way mapping (three directions) of the third block.

Proof:

Let a given three-state stochastic matrix be

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Without loss of generality, $\{b_1, b_2, b_3\}$ is assumed as $\{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$.

1. By selecting s_3 as the off-pivot state, the q-parameters are computed,

$$q_1 = 4 a_{31} - 2 a_{21}$$

$$q_2 = 4 a_{32} - 2 a_{12} .$$

Let q_1 and q_2 be negative so that

$$2 a_{31} < a_{21} \quad \text{and} \quad 2 a_{32} < a_{12} .$$

2. By choosing s_2 as the off-pivot state, the q-parameters are

$$q_1 = 4 a_{21} - 2 a_{31}$$

$$q_2 = 4 a_{23} - 2 a_{13} .$$

since $a_{31} < \frac{1}{2} a_{21}$ from the first mapping,

$$q_1 = 4 a_{21} - 2 a_{31} > 4 a_{21} - 2 \frac{1}{2} a_{21} = 3 a_{21} > 0$$

so q_1 is positive.

Suppose q_2 is negative, $2 a_{23} < a_{13} .$

3. In the last mapping of the third block, the q-parameters are

$$q_1 = 4 a_{12} - 2 a_{32}$$

$$q_2 = 4 a_{13} - 2 a_{23} .$$

q_1 and q_2 are positive since $a_{12} > 2a_{32}$ and $a_{13} > 2a_{23} .$

For the sake of simplicity, mapping notations are defined as follows:

The first row indicates block assignment to the states, and the second row

presents p-q parameters assignment to the blocks. The form is as follows:

$$\left(\begin{array}{cc|c} y & z & \\ p & q & \end{array} \right)$$

For example

$$\left(\begin{array}{cc|c} b_1 & b_2 & \\ p_1 & p_2 & \end{array} \right)$$

means blocks b_1 and b_2 are assigned into states s_1 and s_2 , respectively, and the numerical constants, p_{11}, p_{21}, q_1 and p_{12}, p_{22}, q_2 are assigned to columns one and two, respectively. Finally "|" in the third column shows that state s_3 is the off-pivot state. This mapping assignment is related to compute the unknown variables a, b, c, d, e and f , Equations (4.14) can be represented in the following shortened form.

$$\left(\begin{array}{cc|c} c & e & \\ f & a & \\ d & b & \end{array} \right)$$

indicates a, b and e are always associated with block b_2 and p_1 -parameters, and c, d and f are always associated with block b_1 and p_2 -parameters. When the block assignment is reversed like $\left(\begin{array}{cc} b_2 & b_1 \\ p_1 & p_2 \end{array} \right)$, the expressions of a, b, c, d, e and f are changed like

$$\left(\begin{array}{cc|c} a & f & \\ e & c & \\ b & d & \end{array} \right)$$

which means $p_{22} \rightarrow p_{11}$, $q_2 \rightarrow q_1$, $p_{11} \rightarrow p_{22}$, $q_1 \rightarrow q_2$, $p_{12} \rightarrow p_{21}$ and $p_{21} \rightarrow p_{12}$.

Also, a, c always come from the diagonal terms of the pivot matrix; b, d always come from the off-pivot state (in the above example, s_3), and e, f always come from the off-diagonal terms of the pivot matrix.

Since $q_2 = -\frac{2}{5}$ of $M(x_1)$ in the numerical example 4.2, we must find another mapping in order to get three two-state stochastic matrices.

Consider a mapping

$$\left(\begin{array}{c|c} b_1 & b_2 \\ \hline p_1 & p_2 \end{array} \right) \rightarrow \left(\begin{array}{c|c} c & e \\ \hline d & b \\ \hline f & a \end{array} \right)$$

the six equations to solve are as follows.

$$\begin{aligned} (1-c)(1-e) &= 1-\alpha & (1-a)e+be &= 2\alpha r \\ 2d(1-e)+(1-c)f+df &= 4\beta s & (1-a)e+2b(1-f)+be &= 4\beta(1-s) \\ (1-c)f+df &= 2\gamma(1-t) & (1-a)(1-f) &= 1-\gamma \end{aligned} \quad (4.29)$$

Then $p_{11} = \alpha$, $q_1 = 4\beta s - 2\gamma(1-t)$, $p_{21} = 2\gamma(1-t)$, $p_{12} = 2\alpha r$, $q_2 = 4\beta(1-s) - 2\alpha r$ and $p_{22} = \gamma$.

To find a, b, c, d, e and f ,

$$a = \frac{p_{22} - f}{1-f} \quad (4.30)$$

$$b = \frac{q_2}{2(1-f)} \quad (4.31)$$

$$c = \frac{p_{11} - e}{1-e} \quad (4.32)$$

$$d = \frac{q_1}{2(1-e)} \quad (4.33)$$

$$f = \frac{1}{2p_{12}} [2p_{12} - \{2(1-p_{22}) + q_2\}e] \quad (4.34)$$

$$e = \frac{2p_{12}[\{2(1-p_{11})+q_1\}-2p_{21}]}{\{2(1-p_{11})+q_1\}\{2(1-p_{22})+q_2\}-4p_{12}p_{21}} \quad (4.35)$$

$$p_{11} = \frac{7}{10}, \quad q_1 = 4 \frac{4}{10} \frac{1}{2} - 2 \frac{3}{10} \frac{1}{3} = \frac{4}{5} - \frac{1}{5} = \frac{3}{5}, \quad p_{21} = 2 \frac{3}{10} \frac{1}{3} = \frac{1}{5},$$

$$p_{12} = 2 \frac{7}{10} \frac{1}{7} = \frac{1}{5}, \quad q_2 = 4 \frac{4}{10} \frac{1}{2} - 2 \frac{7}{10} \frac{1}{7} = \frac{4}{5} - \frac{1}{5} = \frac{3}{5},$$

$$p_{22} = \frac{3}{10}.$$

Thus q_1 and q_2 are positive.

Decomposed matrices are found as:

$$a = \frac{1}{50}, \quad b = \frac{21}{50}, \quad c = \frac{39}{60}, \quad d = \frac{7}{20}, \quad e = \frac{1}{7}, \quad \text{and} \quad f = \frac{2}{7}.$$

By substituting these values in (4.11) and taking the sum of all elements in a block divided by the number of rows in it, (see Definition 4.14), it is easy to check the substituted matrix having the same quantities as (4.24).

Note: Equations, (4.30) through (4.35) are the same as (4.15) through (4.18) and (4.22) and (4.23). The next step is to show that the two diagonal terms of the pivot matrix are positive when equating (4.11) with (4.12). The equating result is shown in (4.13) or (4.14). The two terms which assure a and $c \geq 0$ of (4.15) and (4.17), $p_{11}-e \geq 0$ and $p_{22}-f \geq 0$, are included in the next theorem. The necessary and sufficient conditions of a decomposable three-state matrix are also stated in the following theorem.

Theorem 4.4:

Suppose $\{b_1, b_2, b_3\}$ is an independent partition, and the block consisting of four states is always assigned as the off-pivot state.

A given three-state stochastic matrix is decomposable by three two-state stochastic matrices if and only if

$$1. \quad 2 \geq q_i \geq 0 \quad \text{for } i = 1, 2, \quad (4.36)$$

$$2. \quad \epsilon = \{2(1-p_{11})+q_1\}\{2(1-p_{22})+q_2\}-4p_{12}p_{21} > 2p_{i\bar{i}}\{2+q_i-2(p_{1i}+p_{2i})\} \geq 0 \quad (4.37)$$

and

$$3. \quad \epsilon \geq \frac{2p_{i\bar{i}}}{p_{ii}} \{2+q_i-2(p_{1i}+p_{2i})\} \quad \text{for } i = 1, 2. \quad (4.38)$$

$$\begin{aligned} \text{where } \bar{i} &= 2 & \text{if } i=1 \\ &= 1 & \text{if } i=2. \end{aligned}$$

Proof:

Without loss of generality, $\{b_1, b_2, b_3\}$ is assumed as $\{0, 4; 1, 3; 2, 5, 6, 7\}$.

Proof of Sufficiency:

Taking the pivot matrix $\begin{pmatrix} 1-p_{11} & p_{12} \\ p_{21} & 1-p_{22} \end{pmatrix}$, the row elements of the off-pivot state, and b_1 and b_2 as the pivot states p_1 and p_2 , respectively, (4.15) through (4.18), and (4.22) and (4.23) are obtained by equating the block matrix with the given three-state matrix.

Note: $1 \geq p_{11}, p_{22} \geq 0$, $2 \geq p_{12}, p_{21} \geq 0$ and $4 \geq q_1, q_2 \geq -2$. From hypothesis 1, (4.16) and (4.18), $1 \geq b$, $d \geq 0$, if $1 > e$, $f \geq 0$. From (4.22), (4.23) and hypothesis 2,

$$e = \frac{2p_{12}}{\epsilon} \{2+q_1-2(p_{11}+p_{21})\} < 1, \quad (4.39)$$

$$f = 1 - \frac{2p_{12}}{2p_{12}} \frac{\{2(1-p_{22})+q_2\}}{\epsilon} \{2+q_1-2(p_{11}+p_{21})\}$$

$$\begin{aligned} &= \frac{1}{\epsilon} [\{2(1-p_{11})+q_1\}\{2(1-p_{22})+q_2\}-4p_{12}p_{21} \\ &\quad - \{2(1-p_{22})+q_2\}\{2+q_1-2(p_{11}+p_{21})\}] \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\epsilon} [\{2(1-p_{22})+q_2\}\{2(1-p_{11})+q_1-2-q_1+2(p_{11}+p_{21})\}-4p_{12}p_{21}] \\
&= \frac{2p_{21}}{\epsilon} \{2(1-p_{22})+q_2-2p_{12}\} = \frac{2p_{21}}{\epsilon} \{2+q_2-2(p_{12}+p_{22})\} < 1. \quad (4.40)
\end{aligned}$$

Since p_{ii} , $1-p_{ii}$, $q_i \geq 0$ (by Lemma 4.2) for $i = 1, 2$, from (4.40) and the similar equation of e to the above equation of f ,

$$e = 1 - \frac{\{2(1-p_{11})+q_1\}\{2(1-p_{22})+q_2-2p_{12}\}}{\epsilon} = \frac{2p_{12}\{2(1-p_{11})+q_1-2p_{21}\}}{\epsilon} \quad (4.41)$$

so $1 > e$, $f \geq 0$

Also from (4.15), (4.17) and hypothesis 3, namely (4.38),

$$p_{11} \geq e \quad \text{and} \quad p_{22} \geq f$$

$$\text{and } 1 \geq p_{ii} \geq 0 \quad \text{so that} \quad 1 \geq a, c \geq 0. \quad (4.42)$$

Proof of Necessity:

Since the three-state matrix is decomposable, there are $1 \geq a, b, c, d \geq 0$ and $1 > e, f \geq 0$.

From Lemma 4.2, there is at least a mapping of blocks to states such that $q_i \geq 0$ for $i = 1, 2$. From (4.16) and (4.18)

$$2(1-e) \geq q_1 \quad \text{and} \quad 2(1-f) \geq q_2.$$

Also

$$1-e \geq p_{11} - e \geq 0 \quad \text{and} \quad 1-f \geq p_{22} - f \geq 0$$

Inserting (4.22) and (4.23) into the above equation

$$p_{11} - e \geq \frac{2p_{12}\{2(1-p_{11})+q_1\}-4p_{22}p_{21}}{\epsilon} = \frac{2p_{12}\{2+q_1-2(p_{11}+p_{21})\}}{\epsilon} \quad (4.43)$$

and

$$p_{22} - f \geq 0$$

$$p_{22} \geq \frac{2p_{21}}{\epsilon} \{2+q_2 - 2(p_{12}+p_{22})\} \quad (4.44)$$

we have

$$\epsilon \geq \frac{2p_{ii}}{p_{ii}} \{2+q_i - 2(p_{1i}+p_{2i})\} \quad \text{for } i = 1, 2 \quad (4.45)$$

From (4.42), (4.43) and (4.44)

$$1 > e = \frac{2p_{12}}{\epsilon} \{2+q_1 - 2(p_{11}+p_{21})\} \geq 0 \quad (4.46)$$

$$1 > f = \frac{2p_{21}}{\epsilon} \{2+q_1 - 2(p_{12}+p_{22})\} \geq 0, \quad (4.47)$$

$$\epsilon > 2p_{ii} \{2+q_i - 2(p_{1i}+p_{2i})\} \geq 0 \quad (4.48)$$

From (4.15) and (4.17),

$$1 \geq p_{ii}.$$

Finally, by (4.16) and (4.18)

$$2(1-f) \geq q_2 \quad \text{and} \quad 2(1-e) \geq q_1, \quad (4.49)$$

since $1 > e, f \geq 0$,

$$2 \geq q_i \quad \text{for } i = 1, 2.$$

The decomposition of a three-state PSM by three two-state PSM's may be stated in procedure form.

Step (A). Evaluate α, β, γ and r, s, t of a three-state matrix from (4.12).

Step (B). Find an independent partition of eight states composed of three B-vectors or eight nodes in B-unit volume.

Step (C). Determine a mapping of the partition on states of the given three-state PSM such that all q-parameters of the mapping are positive by using (4.14), (4.27) or (4.28). Lemma 4.2 shows that there exists at least one single set of such q-parameters.

Step (D). Apply Theorem 4.4 on the partition and the mapping. If the conditions of the theorem are not satisfied, then find another mapping such that all q-parameters are positive and go to step (D). If there is no more such mapping, then the given three-state PSM is not decomposable.

Step (E). Otherwise, the three-state PSM is decomposable (only when the conditions are satisfied). Compute the state transition probabilities of the three (component) two-state PSM's.

The decomposition procedure is demonstrated in the following example.

Example 4.3:

A given three-state transition matrix is (4.24)

Step (A) from (4.12), $\alpha = 7/10$, $\beta = 4/10$, $\gamma = 3/10$ and $r = 1/7$, $s = 1/2$, $t = 2/3$.

Step (B) from Figure 4.5, an independent partition is found as $\{b_1, b_2, b_3\} = \{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$.

Step (C) A mapping is chosen as $\left(\begin{array}{cc} b_1 & b_2 \\ s_1 & s_2 \end{array} \right)$ where s_1 and s_2 are the first and the second states of the three-state PSM.

In this mapping the first and second blocks b_1 and b_2 are assigned onto the first and second states s_1 and s_2 of the given three-state transition matrix. In other words, the sum of state probabilities in a block represented by three binary digits, or by the combined individual states of the three two-state matrices M_1, M_2 and M_3 of (4.10), is regarded as the state probability of the state mapped by the block in the three-state matrix. The resulted equations of this mapping are described in (4.13) through (4.23). From (4.14)

$$q_1 = 4\gamma(1-t) - 2\beta s = 4 \frac{3}{10} \frac{1}{3} - 2 \frac{4}{10} \frac{1}{2} = 0,$$

$$q_2 = 4\gamma(t) - 2\alpha(1-r) = 4 \frac{3}{10} \frac{2}{3} - 2 \frac{7}{10} \frac{6}{7} = -\frac{2}{5}.$$

The quantity of q_2 is negative so that another mapping which provides positive q -parameters, must be considered. Since there are three-way mapping of the third block b_3 onto each direction of three-dimensional space, we take the other mapping $\left(\begin{array}{c|c} b_1 & b_2 \\ s_1 & s_3 \end{array} \right)$, where s_3 is the third state of the three-state matrix.

The resulted equations of the second mapping are described in (4.27), and (4.29) through (4.35). From (4.27),

$$q_1 = 4\beta s - 2\gamma(1-t) = 4 \frac{4}{10} \frac{1}{2} - 2 \frac{3}{10} \frac{1}{3} = \frac{3}{5}$$

the quantity of q_2 is found as positive,

$$q_2 = 4\beta(1-s) - 2\alpha r = 4 \frac{4}{10} \frac{1}{2} - 2 \frac{7}{10} \frac{1}{7} = \frac{3}{5},$$

so that this mapping provides a set of positive q-parameters.

Step (D) From the quantities $\alpha, \beta, \gamma, r, s$ and t and the mapping (4.29), $p_{11} = \frac{7}{10}$, $p_{12} = \frac{1}{5}$, $p_{21} = \frac{1}{5}$, $p_{22} = \frac{3}{10}$, $q_1 = \frac{3}{5}$ and $q_2 = \frac{3}{5}$. From the second condition of the theorem,

$$\begin{aligned} \varepsilon &= (2 \frac{3}{10} + \frac{3}{5}) (2 \frac{7}{10} + \frac{3}{5}) - 4 \frac{1}{5} \frac{1}{5} = \frac{12}{5} - \frac{4}{25} \\ &= \frac{56}{25}, \end{aligned}$$

$$\begin{aligned} 2p_{12} \{2+q_1 - 2(p_{11}+p_{21})\} &= 2 \frac{1}{5} \{2 + \frac{3}{5} - 2(\frac{7}{10} + \frac{1}{5})\} \\ &= \frac{2}{5} (\frac{13}{5} - \frac{8}{5}) = \frac{2}{5} \frac{5}{5} = \frac{2}{5}. \end{aligned}$$

$$\begin{aligned} 2p_{21} \{2+q_2 - 2(p_{12}+p_{22})\} &= 2 \frac{1}{5} \{2 + \frac{3}{5} - 2(\frac{1}{5} + \frac{3}{10})\} \\ &= \frac{2}{5} (\frac{13}{5} - \frac{5}{5}) = \frac{2}{5} \frac{8}{5} = \frac{16}{25}. \end{aligned}$$

The first condition,

$$2 > \frac{3}{5} > 0 \text{ is held;}$$

the second condition

$$\frac{56}{25} > \frac{10}{25} > 0 \quad \text{and} \quad \frac{56}{25} > \frac{16}{25} > 0, \text{ and}$$

the third condition

$$\frac{56}{25} > \frac{10}{5} \frac{2}{7} = \frac{20}{35} \quad \text{and}$$

$$\frac{56}{25} - \frac{16}{5} \frac{2}{3} = 2 \frac{6}{25} - 2 \frac{2}{15} = \frac{8}{75} > 0 \text{ are also held,}$$

All three conditions of the theorem are satisfied; therefore, the example matrix is decomposable.

Step (E) From (4.30), (4.31), (4.32), (4.33), (4.34) and (4.35),

$$a = \frac{1}{50}, \quad b = \frac{21}{50}, \quad c = \frac{39}{60}, \quad d = \frac{7}{20}, \quad e = \frac{1}{7}$$

and $f = \frac{2}{7}.$

The decomposed matrices are

$$M_1(x_1) = \begin{pmatrix} \frac{49}{50} & \frac{1}{50} \\ \frac{21}{50} & \frac{29}{50} \end{pmatrix}, \quad M_2(x_1) = \begin{pmatrix} \frac{21}{60} & \frac{39}{60} \\ \frac{7}{20} & \frac{13}{20} \end{pmatrix}$$

and

$$M_3(x_1) = \begin{pmatrix} \frac{6}{7} & \frac{1}{7} \\ \frac{2}{7} & \frac{5}{7} \end{pmatrix}.$$

Theorem 4.5:

If all state stochastic matrices associated with each input symbol in a three-state PSM are decomposable, then the three-state PSM can be decomposed by three two-state PSM's.

Proof:

From Theorem 4.4, since each three-state matrix is decomposable, these exist decomposed three two-state matrices for each three-state matrix. By defining the first two-state PSM consisting of all the first two-state matrices, the second two-state PSM with all the second two-state matrices and the third two-state PSM with all the third two-state matrices, the original three-state

transitions are isomorphic to the transition of blocks which are used for decomposition. ■

The ϵ equation in Theorem 4.4 can be rewritten in a matrix form so that the decomposition, $n > 3$, can be handled in a similar manner.

In (4.14) and (4.29), p - q parameters are rewritten in:

$$\begin{pmatrix} 1-p_{11} & p_{12} \\ p_{21} & 1-p_{22} \\ q_1 & q_2 \end{pmatrix} = \begin{pmatrix} 1-p'_{11} & 2p'_{12} \\ 2p'_{21} & 1-p'_{22} \\ 4q'_1-2p'_{21} & 4q'_2-2p'_{12} \end{pmatrix}. \quad (4.50)$$

For the sake of simplicity, each parameter after the replacement (4.50) is expressed by its original notation without prime, thus we get the following equation.

$$\begin{aligned} \epsilon &= \{2(1-p_{11})+4q_1-2p_{21}\}\{2(1-p_{22})+4q_2-2p_{12}\}-4 \cdot 4p_{12}p_{21} \\ &= 4(1-p_{11})(1-p_{22})+8(1-p_{11})q_2-4p_{12}(1-p_{11})+8(1-p_{22})q_1 \\ &\quad + 16q_1q_2-8p_{12}q_1-4p_{21}(1-p_{22})-8p_{21}q_2+4p_{12}p_{21}-4 \cdot 4p_{12}p_{21} \\ &= 4\{(1-p_{11})(1-p_{22})-4p_{12}p_{21}\}+8\{(1-p_{22})q_2-p_{12}q_1\} \\ &\quad + 8\{(1-p_{22})q_1-p_{21}q_2\}-4p_{12}(1-p_{11})-4p_{21}(1-p_{22}) \\ &\quad - 8p_{12}p_{21}+16q_1q_2 \\ &= 4 \begin{vmatrix} 1-p_{11} & p_{12} \\ p_{21} & 1-p_{22} \end{vmatrix} + 8 \begin{vmatrix} 1-p_{11} & p_{12} \\ q_1 & q_2 \end{vmatrix} + 8 \begin{vmatrix} q_1 & q_2 \\ p_{21} & 1-p_{22} \end{vmatrix} \\ &\quad - 4p_{12}(1-p_{11})-4p_{21}(1-p_{22})-8p_{12}p_{21}+16q_1q_2 \end{aligned}$$

$$\begin{aligned}
&= 4 \left\{ \begin{vmatrix} 1-p_{11} & p_{12} \\ p_{21} & 1-p_{22} \end{vmatrix} + 4q_1q_2 \right\} + 4 \left\{ 2 \begin{vmatrix} 1-p_{11} & p_{12} \\ q_1 & -q_2 \end{vmatrix} - p_{21}(1-p_{22}) \right\} \\
&+ 4 \left\{ 2 \begin{vmatrix} q_1 & q_2 \\ p_{21} & 1-p_{22} \end{vmatrix} - p_{12}(1-p_{11}) \right\} - 8p_{12}p_{21}. \quad (4.51)
\end{aligned}$$

4.4 Decomposition of a Matrix Which Consists of Dependent Relations Among Its Rows

In the previous section, the discussion of decomposition is based on a linearly independent relation between rows of a given matrix. When rows in a matrix are linearly dependent, the required number of elemental two-state matrices is reduced in decomposition. This special case is discussed here.

Using an example,

$$M(x_1) = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{pmatrix} 1-\alpha & \alpha(1-r) & \alpha r \\ \beta s & 1-\beta & \beta(1-s) \\ \gamma t & \gamma(1-t) & 1-\gamma \end{pmatrix} \end{matrix} \quad (4.12)$$

If numerical quantities of β and s are dependent on α, r, γ and t , namely $\beta = f(\alpha, r, \gamma, t)$, $s = g(\alpha, r, \gamma, t)$, the required number of two-state matrices for decomposition is two. These are

$$M_1(x_1) = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} \quad \text{and} \quad M_2(x_1) = \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix}. \quad (4.52)$$

A matrix composed of the two preceding matrices using blocks: $\{00\}=\bar{0}=s_1$, $\{01,10\}=\bar{1}, \bar{2}=s_2$, $\{11\}=\bar{3}=s_3$ follows:

$$\begin{pmatrix} (1-a)(1-c) & (1-a)c+a(1-c) & ac \\ \frac{1}{2}\{(1-a)d+b(1-c)\} & \frac{1}{2}\{(1-a)(1-d)+(1-b)(1-c)\} & \frac{1}{2}\{a(1-d)+(1-b)c\} \\ bd & b(1-d)+(1-b)d & (1-b)(1-d) \end{pmatrix} \quad (4.53)$$

$$\begin{aligned} (1-a)(1-c) &= 1-\alpha, & (1-a)c+a(1-c) &= \alpha(1-r) \\ b(1-d)+(1-b)d &= \gamma(1-t), & (1-b)(1-d) &= 1-\gamma, \end{aligned}$$

we get

$$\begin{aligned} \alpha &= a+c-ac, & a+c-ac-ac &= \alpha(1-r) = \alpha-ac \\ \gamma &= b+d-bd, & b+d-bd-bd &= \gamma(1-t) = \gamma-bd \end{aligned} \quad (4.54)$$

$$r = \frac{ac}{a+c-ac} \quad \text{and} \quad t = \frac{bd}{b+d-bd}. \quad (4.55)$$

By inserting $a = \frac{\alpha r}{c}$ and $b = \frac{\gamma t}{d}$ into the right-hand sides of α and γ ,

$$\alpha = \frac{\alpha r}{c} + c - \frac{\alpha r}{c} c, \quad \gamma = \frac{\gamma t}{d} + d - \frac{\gamma t}{d} d \quad (4.56)$$

$$c^2 - (\alpha r + \alpha)c + \alpha r = 0$$

$$d^2 - (\gamma t + \gamma)d + \gamma t = 0$$

Hence

$$c = \frac{(\alpha r + \alpha) \pm \sqrt{\alpha^2(r+1)^2 - 4\gamma t}}{2} \quad (4.57)$$

$$d = \frac{(\gamma t + \gamma) \pm \sqrt{\gamma^2(t+1)^2 - 4\gamma t}}{2} \quad (4.58)$$

$$a = \frac{\alpha r}{c} = \frac{2\alpha r}{\alpha(r+1) \pm \sqrt{\alpha^2(r+1)^2 - 4\gamma t}} \quad (4.59)$$

$$b = \frac{\gamma t}{d} = \frac{2\gamma t}{\gamma(t+1) \pm \sqrt{\gamma^2(t+1)^2 - 4\gamma t}} \quad (4.60)$$

Decomposable conditions are $0 < a, b, c, d < 1$, so that we have to select the appropriate signs for these square roots.

4.5 Multiplication of Stochastic Matrices by Decomposed Two-State Matrices

State transitions of an n -state machine which processes an ℓ -length input string are the product of matrices $M_{m-\ell+1} \dots M_{m-1} M_m$, where M_i for $i=m-\ell+1, \dots, m-1, m$ is an $n \times n$ stochastic matrix. Since interconnected (decomposed) two-state component machines are isomorphic, the state transitions of the original n -state machine can be computed by matrix products of the component machines. When we are interested in a single state transition, s_i to s_j in the original n -state PSM, the probability p_{ij} of the state transition can be found in the element (i, j) of the product of the n -state matrices. The probability p_{ij} can also be computed by the product of the component (two-state stochastic) matrices along with the interconnection i to j with which we are concerned. Multiplication of the component matrices is presented in this section.

Thus the results of a two-state PSM in Chapter III may be applicable to all insights of the decomposed component machines and consequently to the original n -state machine.

Let A and B be matrices;

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ a_{\ell 1} & a_{\ell 2} & \dots & a_{\ell m} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix} \quad (4.61)$$

Definition 4.10:

By moving every row up one and inserting the displaced top row into the vacant bottom row, B becomes 1B such as

$$1_B = \begin{pmatrix} b_{21} & b_{22} & \dots & b_{2n} \\ b_{31} & b_{32} & \dots & b_{3n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ b_{11} & b_{12} & \dots & b_{1n} \end{pmatrix}$$

Similarly $2_B, \dots, {}^{m-1}_B$ are found

$$2_B = \begin{pmatrix} b_{31} & b_{32} & \dots & b_{3n} \\ b_{41} & b_{42} & \dots & b_{4n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ b_{21} & b_{22} & \dots & b_{2n} \end{pmatrix}$$

and

$${}^{m-1}_B = \begin{pmatrix} b_{m1} & b_{m2} & \dots & b_{mn} \\ b_{11} & b_{12} & \dots & b_{1n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ b_{m-11} & b_{m-12} & \dots & b_{m-1n} \end{pmatrix}$$

in this section, it is assumed that each row and column involved in matrix multiplication consist of each other so that the multiplication is always defined.

Definition 4.11:

The sum of i_B for $i=0,1,\dots,m-1$ is denoted as

$$R_B = {}^0_B + {}^1_B + {}^2_B + \dots + {}^{m-1}_B \quad (4.62)$$

where ${}^0_B = B$, and m is the number of rows.

Definition 4.12:

A special multiplication between A and B called dot-circle multiplication is defined as

$$A \odot B = A \cdot R_B = \frac{1}{m} (A \cdot B + A \cdot I_B + \dots + A \cdot I_B^{m-1}) \quad (4.63)$$

where \cdot indicates the regular matrix multiplication.

Definition 4.13:

Let A_{ij} and B_{ij} , which are referred as 'block matrices', be $q \times q$ submatrices in A and B, respectively. A and B become

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ A_{\ell 1} & A_{\ell 2} & \dots & A_{\ell m} \end{pmatrix} = (A_{ij}) \quad (4.64)$$

$$B = \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{pmatrix} = (B_{ij}) \quad (4.65)$$

$$A \odot B = \left(\sum_{k=1}^q A_{ik} \odot B_{kj} \right) = (C_{ij}) \quad (4.66)$$

Example

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned}
c_{11} &= A_{11} \cdot R_{B_{11}} + A_{12} \cdot R_{B_{21}} \\
c_{12} &= A_{11} \cdot R_{B_{12}} + A_{12} \cdot R_{B_{22}} \\
c_{21} &= A_{21} \cdot R_{B_{11}} + A_{22} \cdot R_{B_{21}} \\
c_{22} &= A_{21} \cdot R_{B_{12}} + A_{22} \cdot R_{B_{22}} .
\end{aligned}
\tag{4.67}$$

Definition 4.14:

Let A be

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} .$$

The total sum of elements in A divided by the row number m is called a "block sum" denoted by [A], such as

$$[A] = \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^n a_{ij} .$$

Theorem 4.6:

$$A \odot B = \frac{1}{m} (A 0^0 B + A 0^1 B + A 0^2 B + \dots + A 0^{m-1} B)
\tag{4.68}$$

Proof:

Without loss of generality, we use 4×4 matrices of A and B. A_{ij} and B_{ij} are 2×2 submatrices of A and B, respectively, where $i, j=1, 2$.

Left-hand side:

$$A \odot B = \frac{1}{4} A (0^0 B + 0^1 B + 0^2 B + 0^3 B)$$

$$= \frac{1}{m} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \left\{ \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} + \begin{pmatrix} b_{21} & b_{22} & \dots & b_{24} \\ b_{31} & b_{32} & \dots & b_{34} \\ b_{41} & b_{42} & \dots & b_{44} \\ b_{11} & b_{12} & \dots & b_{14} \end{pmatrix} + \begin{pmatrix} B_{21} & B_{22} \\ B_{11} & B_{12} \end{pmatrix} + \begin{pmatrix} b_{41} & \dots & b_{44} \\ \cdot & \dots & \cdot \\ \cdot & \dots & \cdot \\ b_{31} & \dots & b_{34} \end{pmatrix} \right\} \quad (4.69)$$

Right-hand side:

$$A_0^0 B + A_0^1 B' = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} + \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{21} & B_{22} \\ B_{11} & B_{12} \end{pmatrix} \quad (4.70)$$

$$= \begin{pmatrix} A_{11}^R B_{11}^R + A_{12}^R B_{21}^R + A_{11}^R B_{21}^R + A_{12}^R B_{11}^R & A_{11}^R B_{12}^R + A_{12}^R B_{22}^R + A_{11}^R B_{22}^R + A_{12}^R B_{12}^R \\ A_{21}^R B_{11}^R + A_{22}^R B_{21}^R + A_{21}^R B_{21}^R + A_{22}^R B_{11}^R & A_{21}^R B_{12}^R + A_{22}^R B_{22}^R + A_{21}^R B_{22}^R + A_{22}^R B_{12}^R \end{pmatrix} \quad (4.71)$$

Consider the (1,1) elements in (4.71) and (4.69),

$$A_{11}^0 (B_{11}^0 + B_{11}^1 + B_{21}^0 + B_{21}^1) + A_{12}^0 (B_{21}^0 + B_{21}^1 + B_{11}^0 + B_{11}^1) \quad (4.72)$$

By definition

$$B' = \begin{pmatrix} b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} \text{ and } B'' = \begin{pmatrix} b_{41} & b_{42} \\ b_{11} & b_{12} \end{pmatrix},$$

and comparison

$$A_{11} (B_{11}^1 + B_{21}^1) + A_{12} (B_{21}^1 + B_{11}^1) \text{ in (4.72) to}$$

$$A_{11} (B' + B'') + A_{12} (B' + B'') \text{ in (4.69).}$$

$$B_{11}^1 + B_{21}^1 = \begin{pmatrix} b_{21} & b_{22} \\ b_{11} & b_{12} \end{pmatrix} + \begin{pmatrix} b_{41} & b_{42} \\ b_{31} & b_{32} \end{pmatrix} = B' + B''$$

Comparing (1,2), (2,1) and (2,2) elements in (4.71) and (4.69), each element is

equal.



We shall present the scheme using the example of a three-state PSM.

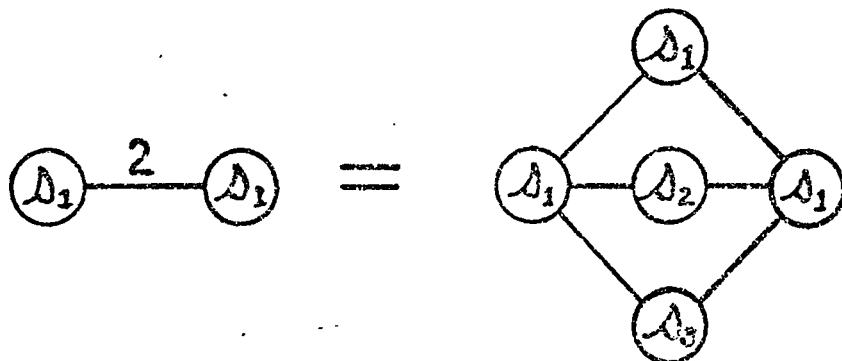
The scheme could easily be extended for n -state PSM, $n > 3$.

Suppose a partition $\{b_1, b_2, b_3\}$ is $\{\overline{0,4}; \overline{1,3}; \overline{2,5,6,7}\}$. Since the original PSM and the decomposed two-state PSM's are isomorphic, there must be an equivalent matrix multiplication on the decomposed matrices to the multiplication on the original three-state matrices.

Using state transfer notation "ij" which indicates s_i to s_j instead of parameters a, b, c, \dots , (4.11) becomes

$$\begin{array}{c}
 \begin{array}{cc} & \begin{array}{ccc} s_1 & & s_2 & & s_3 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{pmatrix}
 \begin{array}{cc|cc|cc|cc}
 00 & 04 & 01 & 03 & 02 & 05 & 06 & 07 \\
 40 & 44 & 41 & 43 & 42 & 45 & 46 & 47 \\
 \hline
 10 & 14 & 11 & 13 & 12 & 15 & 16 & 17 \\
 30 & 34 & 31 & 33 & 32 & 35 & 36 & 37 \\
 \hline
 20 & 24 & 21 & 23 & 22 & 25 & 26 & 27 \\
 50 & 54 & 51 & 53 & 52 & 55 & 56 & 57 \\
 60 & 64 & 61 & 63 & 62 & 65 & 66 & 67 \\
 70 & 74 & 71 & 73 & 72 & 75 & 76 & 77
 \end{array}
 \end{pmatrix}
 \end{array}
 \end{array}
 = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix}. \quad (4.73)$$

Consider the state transfer s_1 to s_1 after a multiplication denoted as $s_1 \xrightarrow{2} s_1$: there are three ways to reach s_1 .



This computation process is indicated by

$$\begin{aligned} [(M_{11} \ M_{12} \ M_{13}) \odot \begin{pmatrix} M'_{11} \\ M'_{21} \\ M'_{31} \end{pmatrix}] &= \frac{1}{2} [M_{11} \odot M'_{11} + M_{12} \odot M'_{21} + M_{13} \odot M'_{31}] \\ &= [M_{11}] [M'_{11}] + [M_{12}] [M'_{21}] + [M_{13}] [M'_{31}] \end{aligned} \quad (4.74)$$

where the prime indicates the second matrix.

Theorem 4.7:

If each element of an n -state transition matrix is a block matrix, then the state transition s_i to s_j after receiving two inputs is

$$\sum_{k=1}^n [M_{ik}] [M'_{kj}] = \frac{1}{m} [(M_{i1} \ M_{i2} \ \dots \ M_{in}) \odot \begin{pmatrix} M'_{1j} \\ M'_{2j} \\ \vdots \\ M'_{nj} \end{pmatrix}] \quad (4.75)$$

where m is the row number of M_{ik} .

Proof:

Without loss of generality, we shall use (4.73), M matrix, or (4.74), namely $s_i = s_1$ and $s_j = s_1$.

Consider the path, $s_1 - s_1 - s_1$ which means

$$[M_{11}] [M'_{11}] = \frac{1}{2} [M_{11} \odot M'_{11}] \quad (4.76)$$

where $M_{11} = \begin{pmatrix} 00 & 04 \\ 40 & 44 \end{pmatrix}$ and $M'_{11} = \begin{pmatrix} \overline{00} & \overline{04} \\ \overline{40} & \overline{44} \end{pmatrix}$.

Since

$$M_{11} \odot M'_{11} = \frac{1}{2} \{M_{11} \cdot M'_{11} + M'_{11} \cdot M_{11}\}$$

$$= \frac{1}{2} \begin{pmatrix} 00 & 04 \\ 40 & 44 \end{pmatrix} \left\{ \begin{pmatrix} \overline{00} & \overline{04} \\ \overline{40} & \overline{44} \end{pmatrix} + \begin{pmatrix} \overline{40} & \overline{44} \\ \overline{00} & \overline{04} \end{pmatrix} \right\} = \frac{1}{2} \begin{pmatrix} 00 & 04 \\ 40 & 44 \end{pmatrix} \begin{pmatrix} \overline{00+40} & \overline{04+44} \\ \overline{44+04} & \overline{40+00} \end{pmatrix}, \quad (4.77)$$

taking the "block sum" of the individual term in (4.77)

$$\begin{aligned} \frac{1}{2} [M_{11} \cdot M'_{11}] &= \begin{bmatrix} 00 & 04 \\ 40 & 44 \end{bmatrix} \left[\frac{1}{4} 2 (\overline{00+40+04+44}) \right] \\ &= [M_{11}] [M'_{11}]. \end{aligned} \quad (4.78)$$

Similarly, we can compute the path,

$s_1 \text{---} s_2 \text{---} s_1$, considering the following substitution,

$$00 \rightarrow 01, 04 \rightarrow 03, 40 \rightarrow 41, 44 \rightarrow 43,$$

$$\overline{00} \rightarrow \overline{10}, \overline{04} \rightarrow \overline{14}, \overline{40} \rightarrow \overline{30}, \overline{44} \rightarrow \overline{34},$$

We obtain

$$[M_{12}] [M'_{21}] = \frac{1}{2} [M_{12} \cdot M'_{21}]. \quad (4.79)$$

Consider the path, $s_1 \text{---} s_3 \text{---} s_1$

$$[M_{13}] = \frac{1}{2} \{02+05+06+07+42+45+46+47\}$$

$$[M'_{31}] = \frac{1}{4} \{\overline{20+24+50+54+60+64+70+74}\}.$$

we shall prove

$$[M_{13}] [M'_{31}] = \frac{1}{2} [M_{13} \cdot M'_{31}] = [M_{13} \{M'_{31} + {}^1M'_{31} + {}^2M'_{31} + {}^3M'_{31}\}] \quad (4.80)$$

$$[M_{13}] [M'_{31}] = \frac{1}{4} [M_{13}] \{\overline{20+24+50+54+60+64+70+74}\}.$$

and consider $02\{\overline{20+24+50+54+60+64+70+74}\}$

$$+ 05\{\overline{50+54+60+64+70+74+20+24}\}$$

$$+ 06\{\overline{60+64+70+74+20+24+50+54}\}$$

\vdots

$$\vdots$$

$$+ 47\{\overline{70}+\overline{74}+\overline{20}+\overline{24}+\overline{50}+\overline{54}+\overline{60}+\overline{64}\},$$

$$= \left[\frac{1}{2} \begin{pmatrix} 02 & 05 & 06 & 07 \\ 42 & 45 & 46 & 47 \end{pmatrix} \frac{1}{4} \left\{ \begin{pmatrix} \overline{20} & \overline{24} \\ \overline{50} & \overline{54} \\ \overline{60} & \overline{64} \\ \overline{70} & \overline{74} \end{pmatrix} + \begin{pmatrix} \overline{50} & \overline{54} \\ \overline{60} & \overline{64} \\ \overline{70} & \overline{74} \\ \overline{20} & \overline{24} \end{pmatrix} + \begin{pmatrix} \overline{60} & \overline{64} \\ \overline{70} & \overline{74} \\ \overline{20} & \overline{24} \\ \overline{50} & \overline{54} \end{pmatrix} + \begin{pmatrix} \overline{70} & \overline{74} \\ \overline{20} & \overline{24} \\ \overline{50} & \overline{54} \\ \overline{60} & \overline{64} \end{pmatrix} \right\} \right]$$

$$= \frac{1}{2} \left[\begin{pmatrix} 02 & 05 & 06 & 07 \\ 42 & 45 & 46 & 47 \end{pmatrix} \frac{1}{4} \{ M'_{31} + {}^1M'_{31} + {}^2M'_{31} + {}^3M'_{31} \} \right]$$

$$= \frac{1}{2} [M_{13} \cdot M'_{31}] .$$



Example 4.4:

Let M be

$$\overline{M} = \left(\begin{array}{cc|cc} .2 & .1 & .4 & .3 \\ .3 & .2 & .1 & .4 \\ \hline .1 & .3 & .2 & .4 \\ .6 & .1 & .1 & .2 \end{array} \right)$$

where

$$M_{11} = \begin{pmatrix} .2 & .1 \\ .3 & .2 \end{pmatrix}, \quad M_{12} = \begin{pmatrix} .4 & .3 \\ .1 & .4 \end{pmatrix}$$

$$M_{21} = \begin{pmatrix} .1 & .3 \\ .6 & .1 \end{pmatrix} \text{ and } M_{22} = \begin{pmatrix} .2 & .4 \\ .1 & .2 \end{pmatrix},$$

so that

$$M = \begin{pmatrix} [M_{11}] & [M_{12}] \\ [M_{21}] & [M_{22}] \end{pmatrix} = \begin{pmatrix} .4 & .6 \\ .55 & .45 \end{pmatrix}.$$

Consider the paths, $s_1 \text{---} s_1 \text{---} s_1$ and

$$s_1 \text{---} s_2 \text{---} s_1 ,$$

$$[M_{11}][M_{11}] = 0.4 \times 0.4 = 0.16$$

and

$$[M_{12}][M_{21}] = 0.6 \times 0.55 = 0.33$$

Using (4.77),

$$\begin{aligned} \frac{1}{2}[M_{11} \odot M_{11}] &= \frac{1}{2} \left[\frac{1}{2} \begin{pmatrix} .2 & .1 \\ .3 & .2 \end{pmatrix} \left\{ \begin{pmatrix} .2 & .1 \\ .3 & .2 \end{pmatrix} + \begin{pmatrix} .3 & .2 \\ .2 & .1 \end{pmatrix} \right\} \right] \\ &= \frac{1}{4} \left[\begin{pmatrix} .2 & .1 \\ .3 & .2 \end{pmatrix} \begin{pmatrix} .5 & .3 \\ .5 & .3 \end{pmatrix} \right] = \frac{1}{4} \left[2 \begin{pmatrix} .3 \\ .5 \end{pmatrix} \begin{pmatrix} .5 & .3 \end{pmatrix} \right] \\ &= \frac{1}{4} \left[2 \begin{pmatrix} .15 & .09 \\ .25 & .15 \end{pmatrix} \right] = \frac{1}{4} (.15 + .09 + .25 + .15) = .25 \times .64 = .16 \end{aligned}$$

and

$$\begin{aligned} \frac{1}{2}[M_{12} \odot M_{21}] &= \frac{1}{2} \left[\frac{1}{2} \begin{pmatrix} .4 & .3 \\ .1 & .4 \end{pmatrix} \left\{ \begin{pmatrix} .1 & .3 \\ .1 & .4 \end{pmatrix} + \begin{pmatrix} .6 & .1 \\ .6 & .1 \end{pmatrix} \right\} \right] \\ &= \frac{1}{4} \left[\begin{pmatrix} .4 & .3 \\ .1 & .4 \end{pmatrix} \begin{pmatrix} .7 & .4 \\ .7 & .4 \end{pmatrix} \right] = \frac{1}{4} \left[2 \begin{pmatrix} .7 \\ .5 \end{pmatrix} \begin{pmatrix} .7 & .4 \end{pmatrix} \right] \\ &= \frac{1}{4} \left[2 \begin{pmatrix} .49 & .28 \\ .35 & .20 \end{pmatrix} \right] = \frac{1}{4} (.49 + .28 + .35 + .20) \\ &= 0.25 \times 1.32 = .33 \end{aligned}$$

Dot-circle multiplication can be extended to determine state transition, s_i to s_j , on more than two input lengths. For example $s_i \xrightarrow{3} s_j$, typically, $s_i \xrightarrow{3} s_i$ in a three-state PSM will be

$$[(M_{11} M_{12} M_{13}) \odot \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix}] \odot \begin{pmatrix} M_{11} \\ M_{21} \\ M_{31} \end{pmatrix}. \quad (4.81)$$

Note: The commutative, distributive and associative laws on the operator, dot-circle multiplication, do not exist.

Contraction Coefficient:

The decomposed component matrix is shown in (4.11); each block matrix consists of the product of the coefficient matrix $\begin{pmatrix} r & 0 \\ 0 & s \end{pmatrix}$ and the stochastic matrix $\begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}$, where $0 \leq a, b \leq 1$ and $0 \leq r, s \leq 1$. Therefore, when a dot-circle multiplication is performed on a computation of state transition (or a product of block matrices) the result of the product no longer has the stochastic property (the sum of a row is one).

For example,

$$A = \begin{pmatrix} r_1 & 0 \\ 0 & s_1 \end{pmatrix} \begin{pmatrix} 1-a_1 & a_1 \\ b_1 & 1-b_1 \end{pmatrix} \begin{pmatrix} r_2 & 0 \\ 0 & s_2 \end{pmatrix} \begin{pmatrix} 1-a_2 & a_2 \\ b_2 & 1-b_2 \end{pmatrix}, \quad (4.82)$$

where $0 \leq a_1, a_2, b_1, b_2 \leq 1$ and $0 \leq r_1, r_2, s_1, s_2 \leq 1$.

The resulting product can be contracted due to the diagonal matrix in which all diagonal terms are less than one but are nonnegative. When $r_1 = s_1$ and $r_2 = s_2$, the product of (4.82) becomes

$$A = r_1 r_2 \begin{pmatrix} 1-a_1 & a_1 \\ b_1 & 1-b_1 \end{pmatrix} \begin{pmatrix} 1-a_2 & a_2 \\ b_2 & 1-b_2 \end{pmatrix}.$$

When $r_1 \neq s_1$ and $r_2 \neq s_2$, the following process may provide a simple way of computing the matrix product.

Define:

$$\mu = r_2/s_2,$$

$$\lambda_{11} = 1-a_2-\mu b_2, \quad \lambda_{12} = \mu-a_2-\mu b_2,$$

$$\lambda_{21} = \mu^{-1}-\mu^{-1}a_2-b_2, \quad \lambda_{22} = 1-\mu^{-1}a_2-b_2,$$

μ or μ^{-1} is attached on the term λ_{ij} which does appear in the (i,j) element in the matrix; the term A becomes

$$A = \begin{pmatrix} r_1 r_2 & 0 \\ 0 & s_1 s_2 \end{pmatrix} \begin{pmatrix} 1-a_2-a_1 \lambda_{11} & a_2+a_1 \lambda_{12} \\ b_2+b_1 \lambda_{21} & 1-b_2-b_1 \lambda_{22} \end{pmatrix}. \quad (4.83)$$

Equation (4.83) is very similar to the result in Chapter III. Note:

$\lambda_{ij} = 1-a_2-b_2$ for $i,j = 1,2$ when $r_2 = s_2$.

Equation (4.11) can be rewritten in block matrix form:

$$\begin{pmatrix} (1-c)(1-e) \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} & \begin{pmatrix} (1-a)e & 0 \\ 0 & be \end{pmatrix} \begin{pmatrix} 1-c & c \\ 1-c & c \end{pmatrix} & c(1-e) \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \begin{pmatrix} ae & 0 \\ 0 & (1-b)e \end{pmatrix} \begin{pmatrix} 1-c & c \\ 1-c & c \end{pmatrix} \\ \hline \begin{pmatrix} (1-c)f & 0 \\ 0 & df \end{pmatrix} \begin{pmatrix} 1-a & a \\ 1-a & a \end{pmatrix} & (1-a)(1-f) \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} & \begin{pmatrix} cf & 0 \\ 0 & (1-d)f \end{pmatrix} \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, a(1-f) \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} \\ \hline d(1-e) \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} & \begin{pmatrix} (1-a)e & 0 \\ 0 & be \end{pmatrix} \begin{pmatrix} d & 1-d \\ d & 1-d \end{pmatrix} & (1-d)(1-e) \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \begin{pmatrix} ae & 0 \\ 0 & (1-b)e \end{pmatrix} \begin{pmatrix} d & 1-d \\ d & 1-d \end{pmatrix} \\ \hline \begin{pmatrix} (1-c)f & 0 \\ 0 & df \end{pmatrix} \begin{pmatrix} b & 1-b \\ b & 1-b \end{pmatrix} & b(1-f) \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} & \begin{pmatrix} cf & 0 \\ 0 & (1-d)f \end{pmatrix} \begin{pmatrix} b & 1-b \\ b & 1-b \end{pmatrix}, (1-b)(1-f) \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix} \end{pmatrix}$$

Compute the path, $s_1 \xrightarrow{2} s_1$, the elements of the second matrix is represented with primes,

$$M_{11} \Theta_{M_{11}}' = (1-c)(1-e)(1-c')(1-e') \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix} \left\{ \begin{pmatrix} 1-a' & a' \\ b' & 1-b' \end{pmatrix} + \begin{pmatrix} b' & 1-b' \\ 1-a' & a' \end{pmatrix} \right\}$$

$$\begin{aligned}
 M_{12} \odot M'_{21} &= \begin{pmatrix} (1-a)e & 0 \\ 0 & be \end{pmatrix} \begin{pmatrix} 1-c & c \\ 1-c & c \end{pmatrix} \left\{ \begin{pmatrix} (1-c')f' & 0 \\ 0 & d'f' \end{pmatrix} + \begin{pmatrix} d'f' & 0 \\ 0 & (1-c)f' \end{pmatrix} \right\} \begin{pmatrix} 1-a' & a' \\ 1-a' & a' \end{pmatrix} \\
 &= ((1-c')f' + d'f') \begin{pmatrix} (1-a)e & 0 \\ 0 & be \end{pmatrix} \begin{pmatrix} 1-c & c \\ 1-c & c \end{pmatrix} \begin{pmatrix} 1-a' & a' \\ 1-a' & a' \end{pmatrix}
 \end{aligned}$$

$$M_{13} \odot M'_{31} = (c(1-e)) \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \begin{pmatrix} ae & 0 \\ 0 & (1-b)e \end{pmatrix} \begin{pmatrix} 1-c & c \\ 1-c & c \end{pmatrix},$$

$$\begin{aligned}
 &\left\{ \begin{pmatrix} d'(1-e') & \begin{pmatrix} 1-a' & a' \\ b' & 1-b' \end{pmatrix} \\ \begin{pmatrix} (1-c')f' & 0 \\ 0 & d'f' \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ b' & 1-b' \end{pmatrix} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} d'(1-e') & 0 \\ 0 & (1-c')f' \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ b' & 1-b' \end{pmatrix} \\ \begin{pmatrix} d'f' & 0 \\ 0 & d'(1-e') \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ 1-a' & a' \end{pmatrix} \end{pmatrix} \right. \\
 &+ \left. \begin{pmatrix} \begin{pmatrix} (1-c')f' & 0 \\ 0 & d'f' \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ b' & 1-b' \end{pmatrix} \\ \begin{pmatrix} d'(1-e') & 0 \\ 0 & d'(1-e') \end{pmatrix} \begin{pmatrix} 1-a' & a' \\ b' & 1-b' \end{pmatrix} \end{pmatrix} + \begin{pmatrix} \begin{pmatrix} a'f' & 0 \\ 0 & d'(1-e') \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ 1-a' & a' \end{pmatrix} \\ \begin{pmatrix} d'(1-e') & 0 \\ 0 & (1-c')f' \end{pmatrix} \begin{pmatrix} b' & 1-b' \\ b' & 1-b' \end{pmatrix} \end{pmatrix} \right\}
 \end{aligned}$$

These computations are carried out using the results of (4.83) and Chapter III.

Most properties in Chapter III are valid, if $\mu = r/s$, where r and s are the elements of a coefficient matrix, is closer to one.

CHAPTER V

N-STATE INPUT-TRACEABLE PROBABILISTIC SEQUENTIAL MACHINE

5. Introduction

The two-state, input-traceable machine and the decomposition of an n -state PSM by two-state machines have been studied in Section 3.6 and Chapter IV, respectively. From these results, we shall discuss a logical system class which traces all past system performances by knowing only the present state distribution. In this case, all past behaviors of a system can be played backwards like the playing back of a movie film. Most aspects of system evaluation (past state transition history, input sequence supplied, correctness of produced outputs, the most active subsystem, the most likely subsystem where errors occur, etc.) can be accomplished as straight backward tasks by using the playback of the system.

Balser [5] has shown that a playback technique is a very powerful tool for error detection/error correction. The technique is, however, only applicable to a precisely deterministic machine, whenever errors occur in the course of execution. We shall show that a playback technique for a system consisting of PSM's may be applicable, when certain conditions are satisfied, for determining the most likely subsystem in which errors occurred in the past.

Suppose a large system is decomposable by two-state absolutely isolated machines. These two-state machines have the input traceable property in the past as shown in Corollary 3.8. Since the set of decomposed component machines is isomorphic to the original system, we can determine most past system performances from the present state distribution. The input traceable property is a useful and key property of the (absolutely) isolated machines which shall be presented later. The past inputs supplied to the whole system are

the same as the inputs traced by absolutely isolated component machines; therefore, visiting frequencies of individual states can be determined. The state (or subsystem) which has the highest visiting frequency is the most-likely state in which errors occurred in the past machine operation. Thus, we call it an input traceable machine as well as a diagnoseable machine (or in more general terms, a performance traceable machine), if the machine is absolutely isolated after an error occurs.

5.1 Decomposable PSM by Kronecker Product Matrices

When a PSM is decomposable by interconnecting component PSM's with Kronecker product matrices, the PSM may be a past-input-traceable machine. Conditions for an input-traceable machine of a PSM are presented in this section.

Definition 5.1:

A and A' in a Kronecker product matrix $A \otimes A'$ are called the left (the first) and the right (the second) machine, respectively.

Definition 5.2:

Two or more PSM's which have the same number of states may be considered as a single PSM by introducing combined input symbols of the original machines (see Chapter III). The single machine is called a combined machine.

Example 5.1:

Let M_1 and M_2 be two-state PSM's. The state transition matrices of the PSM's are as follows:

$$M_1(x_1) = \begin{pmatrix} 1-a & a \\ b & 1-b \end{pmatrix}, \quad M_2(y_1) = \begin{pmatrix} 1-c & c \\ d & 1-d \end{pmatrix}, \quad \text{and} \quad M_2(y_2) = \begin{pmatrix} 1-e & e \\ f & 1-f \end{pmatrix}.$$

The combined machine M_c is defined by introducing new input symbols ' x_1y_1 ' and ' x_1y_2 '. The corresponding state transition matrices are found as follows:

$$M_c(x_1y_1) = \begin{pmatrix} 1-g & g \\ h & 1-h \end{pmatrix} \quad \text{and} \quad M_c(x_1y_2) = \begin{pmatrix} 1-i & i \\ j & 1-j \end{pmatrix}$$

where $g = a+c-a(c+d)$, $h = b+d-b(c+d)$, $i = a+e-a(e+f)$, and $j = b+h-b(g+h)$.

Note: g, h, i and j are found by matrix multiplication, $M_1(x_1) \cdot M_2(y_1)$ and $M_1(x_1) \cdot M_2(y_2)$.

A matrix product of two Kronecker product matrices is another Kronecker product matrix. Each element of the Kronecker product is a matrix product of two left machines or two right machines. This is described in the following lemma.

Lemma 5.1:

Let $A = (a_{ij})$, $B = (b_{ij})$, $A' = (a'_{ij})$, and $B' = (b'_{ij})$ be matrices, then

$$(A \otimes A') \cdot (B \otimes B') = (AB) \otimes (A'B'). \quad (5.1)$$

Proof:

Let A, B and A', B' be $m \times n$ and $p \times q$ order matrices, respectively.

$$A \otimes A' = (a_{ij} a'_{kl}) = (c_{ik, jl}) = C$$

where the double indices ik, jl of the element of C are ordered lexicographically,

$$ik = 11, 12, \dots, 1p, 21, 22, \dots, 2p, \dots, m1, m2, \dots, mp;$$

$$jl = 11, 12, \dots, 1q, 21, 22, \dots, 2q, \dots, n1, n2, \dots, nq.$$

The ik, mn entry in $A \otimes A'$ is $a_{im} a'_{kn}$, and the mn, jl entry in $B \otimes B'$ is $b_{mj} b'_{nl}$.

in the left side of (5.1). In the right side, the ij entry in AB is $\sum_t a_{it} b_{tj}$, and the kl entry in $A'B'$ is $\sum_r a'_{kr} b'_{rl}$. Therefore, the ik,jl entry in $(AB) \otimes (A'B')$ is $\sum_t a_{it} b_{tj} \sum_r a'_{kr} b'_{rl}$.

$$\sum_m \sum_n a_{im} a'_{kn} \cdot b_{mj} b'_{nl} = \sum_m \sum_n a_{im} b_{mj} a'_{kn} a'_{nl} \quad (5.2)$$

Since the summation of n is separable from the summation of m , (5.2) is rewritten as

$$\sum_m \sum_n a_{im} b_{mj} a'_{kn} b'_{nl} = \sum_m a_{im} b_{mj} \sum_n a'_{kn} b'_{nl} \quad (5.3)$$


The right side of (5.3) is the same as the right side of (5.1). The reverse proof is similar to this proof, so the reverse proof is omitted. \square

Suppose an N -state machine is decomposed by a Kronecker product. From Lemma 5.1, a state transition, represented by the multiplication of Kronecker product terms in the left side of (5.1), can be computed in another way. Namely, the individual terms in the right side of (5.1), AB and $A'B'$, are matrix products of the component machines. A Kronecker product operation on these products represents the state transition. Therefore, the decomposable PSM by Kronecker product component machines may be an input traceable machine.

Theorem 5.1:

If a PSM which consists of four states is decomposable by Kronecker product component machines and if each combined machine of all the left component machines/all the right component machines is an absolutely isolated machine with respect to the (1,2) or (2,1) elements, then the PSM is an input traceable machine.

Proof:

In reference to Figure 4.3, the input symbols to the PSM are in a vector form consisting of input symbols of component machines. From Lemma 5.1, the transition of the PSM is determined by the Kronecker product of the combined left and right component machines. From the imposed assumption, each combined component machine is an absolutely isolated machine. Thus, all past input symbols (string) to each component machine can be determined uniquely using Corollary 3.8, and the past inputs to the PSM can also be determined, since each original symbol to the PSM is a vector of input symbols of each component machine. 

Example 5.2:

A PSM which has four states/four symbols is given as follows:

$$M(y_1) = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} \\ \frac{1}{4} & \frac{7}{10} & \frac{1}{4} & \frac{3}{10} & \frac{3}{4} & \frac{7}{10} & \frac{3}{4} & \frac{3}{10} \\ \frac{1}{16} & \frac{1}{2} & \frac{1}{16} & \frac{1}{2} & \frac{15}{16} & \frac{1}{2} & \frac{15}{16} & \frac{1}{2} \\ \frac{1}{16} & \frac{7}{10} & \frac{1}{16} & \frac{3}{10} & \frac{15}{16} & \frac{7}{10} & \frac{15}{16} & \frac{3}{10} \end{pmatrix} \quad (5.4)$$

$$M(y_2) = \begin{pmatrix} \frac{2}{3} & \frac{1}{5} & \frac{2}{3} & \frac{4}{5} & \frac{1}{3} & \frac{1}{5} & \frac{1}{3} & \frac{4}{5} \\ \frac{2}{3} & \frac{1}{10} & \frac{2}{3} & \frac{9}{10} & \frac{1}{3} & \frac{1}{10} & \frac{1}{3} & \frac{9}{10} \\ \frac{11}{15} & \frac{1}{5} & \frac{11}{15} & \frac{4}{5} & \frac{4}{15} & \frac{1}{5} & \frac{4}{15} & \frac{4}{5} \\ \frac{11}{15} & \frac{1}{10} & \frac{1}{15} & \frac{9}{10} & \frac{4}{15} & \frac{1}{10} & \frac{4}{15} & \frac{9}{10} \end{pmatrix} \quad (5.5)$$

$$M(y_3) = \begin{pmatrix} \frac{2}{4} & \frac{4}{5} & \frac{2}{4} & \frac{1}{5} & \frac{2}{4} & \frac{4}{5} & \frac{2}{4} & \frac{1}{5} \\ \frac{2}{4} & \frac{3}{10} & \frac{2}{4} & \frac{7}{10} & \frac{2}{4} & \frac{3}{10} & \frac{2}{4} & \frac{7}{10} \\ \frac{3}{10} & \frac{4}{5} & \frac{3}{10} & \frac{1}{5} & \frac{7}{10} & \frac{4}{5} & \frac{7}{10} & \frac{1}{5} \\ \frac{3}{10} & \frac{3}{10} & \frac{3}{10} & \frac{7}{10} & \frac{7}{10} & \frac{3}{10} & \frac{7}{10} & \frac{7}{10} \end{pmatrix} \quad (5.6)$$

and

$$M(y_4) = \begin{pmatrix} \frac{3}{4} & \frac{2}{5} & \frac{3}{4} & \frac{3}{5} & \frac{1}{4} & \frac{2}{5} & \frac{1}{4} & \frac{3}{5} \\ \frac{3}{4} & \frac{11}{40} & \frac{3}{4} & \frac{29}{40} & \frac{1}{4} & \frac{11}{40} & \frac{1}{4} & \frac{29}{40} \\ \frac{7}{8} & \frac{2}{5} & \frac{7}{8} & \frac{3}{5} & \frac{1}{8} & \frac{2}{5} & \frac{1}{8} & \frac{3}{5} \\ \frac{7}{8} & \frac{11}{40} & \frac{7}{8} & \frac{29}{40} & \frac{1}{8} & \frac{11}{40} & \frac{1}{8} & \frac{29}{40} \end{pmatrix} \quad (5.7)$$

By inspection, we find

$$\begin{aligned} M(y_1) &= M_1(x_1) \otimes M_2(x_2) \\ M(y_2) &= M_1(x_2) \otimes M_2(x_3) \\ M(y_3) &= M_1(x_3) \otimes M_2(x_4) \\ M(y_4) &= M_1(x_4) \otimes M_2(x_1) \end{aligned} \quad (5.8)$$

$$\text{where } M_1(x_1) = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} \\ \frac{1}{16} & \frac{15}{16} \end{pmatrix}, \quad M_1(x_2) = \begin{pmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{11}{15} & \frac{4}{15} \end{pmatrix} \quad (5.9)$$

$$M_1(x_3) = \begin{pmatrix} \frac{2}{4} & \frac{2}{4} \\ \frac{3}{10} & \frac{7}{10} \end{pmatrix}, \quad \text{and} \quad M_1(x_4) = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{7}{8} & \frac{1}{8} \end{pmatrix}$$

$$M_2(x_1) = \begin{pmatrix} \frac{2}{5} & \frac{3}{5} \\ \frac{11}{40} & \frac{29}{40} \end{pmatrix}, \quad M_2(x_2) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{7}{10} & \frac{3}{10} \end{pmatrix} \quad (5.10)$$

$$M_2(x_3) = \begin{pmatrix} \frac{1}{5} & \frac{4}{5} \\ \frac{1}{10} & \frac{9}{10} \end{pmatrix}, \quad \text{and} \quad M_2(x_4) = \begin{pmatrix} \frac{4}{5} & \frac{1}{5} \\ \frac{3}{10} & \frac{7}{10} \end{pmatrix}$$

Therefore,

$$y_1 = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad y_2 = \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}, \quad y_3 = \begin{pmatrix} x_3 \\ x_4 \end{pmatrix} \quad \text{and} \quad y_4 = \begin{pmatrix} x_4 \\ x_1 \end{pmatrix}.$$

Consider a product of $M(y_i)M(y_j)$ for $i, j = 1, 2, 3, 4$,

$$\begin{aligned} M(y_i) \cdot M(y_j) &= \{M_1(x_i) \otimes M_2(x_{i+1})\} \{M_1(x_j) \otimes M_2(x_{j+1})\} \\ &= \{M_1(x_i) \cdot M_1(x_j)\} \otimes \{M_2(x_{i+1}) \cdot M_2(x_{j+1})\} \end{aligned}$$

where $i+1$, $j+1$ and $k+1$ are found by $\text{mod}(4)$ operation, residues of the power 4.

$$\begin{aligned} M(y_i) \cdot M(y_j) \cdot M(y_k) &= [\{M_1(x_i)M_1(x_j)\} \otimes \{M_2(x_{i+1}) \cdot M_2(x_{j+1})\}] \\ &\quad \cdot [M_1(x_k) \otimes M_2(x_{k+1})] \\ &= \{M_1(x_i)M_1(x_j)M_1(x_k)\} \otimes \{M_2(x_{i+1}) \cdot M_2(x_{j+1}) \cdot M_2(x_{k+1})\}. \end{aligned} \tag{5.11}$$

Thus the behavior of the PSM for any input string $y^* \in Y = \{y_1, y_2, y_3, y_4\}$ is determined by the behaviors of the combined left and the combined right machines, M_1 and M_2 , by corresponding input strings $x^* \in X$, where $X = \{x_1, x_2, x_3, x_4\}$, to y^* .

We shall investigate the combined left and the combined right machines using theorems developed in Chapter III.

The Left Machine

Consider the combined left machine described in (5.9). By using notations in Chapter III and applying Lemma 3.3

$$M_1(x_i) = \begin{pmatrix} 1-\alpha_i & \alpha_i \\ \beta_i & 1-\beta_i \end{pmatrix}$$

$$a' = \max_i \{\alpha_i\} = \frac{3}{4} \quad \text{and} \quad c = \min_i \{\alpha_i\} = \frac{1}{4}$$

$$\lambda_a = \max_i \{1-\alpha_i-\beta_i\} = \frac{1}{5},$$

$$\lambda_c = \min_i \{1-\alpha_i-\beta_i\} = -\frac{1}{8}.$$

The combined left machine is classified in Case 2 of Lemma 3.3 so that

$$S_{\max} = \frac{a}{1-\lambda_a} = \frac{3}{4} \cdot \frac{5}{4} = \frac{15}{16}, \quad S_{\min} = c + a \frac{\lambda_c}{1-\lambda_a} = \frac{1}{4} - \frac{3}{4} \cdot \frac{1}{8} \cdot \frac{5}{4} = \frac{17}{16 \cdot 8} > 0,$$

and $R_{\min} = 0$ by Definition 3.2.

From Theorem 3.3, we have four α_i , $\frac{3}{4}$, $\frac{2}{4}$, $\frac{1}{3}$, and $\frac{1}{4}$. Considering a pair of adjacent α_i and each eigenvalue of each matrix of the machine, we have

i	1	2	3	4
α_i	$\frac{3}{4}$	$\frac{2}{4}$	$\frac{1}{3}$	$\frac{1}{4}$
λ_i	$\frac{3}{16}$	$\frac{1}{5}$	$-\frac{1}{15}$	$-\frac{1}{8}$
β_i	$\frac{1}{16}$	$\frac{3}{10}$	$\frac{11}{15}$	$\frac{7}{8}$

Since λ_1 and λ_2 are positive, the case for the pair (1,2) belongs in Case 1 of Theorem 3.3 where $\alpha_1 - \alpha_2 > \lambda_2 S_{\max}$ must be satisfied.

$$\frac{3}{4} - \frac{2}{4} = \frac{1}{4} > \frac{1}{5} \cdot \frac{15}{16} = \frac{3}{16} \text{ is true.}$$

For the pair (2,3), the case is classified as Case 2 of Theorem 3.3, since $\lambda_2 > 0$ and $\lambda_3 < 0$. No constraint is applied.

The case for the pair (3,4) is determined as Case 4, since λ_3 and λ_4 are negative $\alpha_3 - \alpha_4 > -\lambda_3$ S_{\max} should be held

$$\frac{1}{3} - \frac{1}{4} = \frac{1}{12} > \frac{1}{15} - \frac{15}{16} = \frac{1}{16} \text{ is true.}$$

Similarly, adjacent pairs of β_i are satisfied with the conditions of Theorem 3.3 using $a = \max_i \{\beta_i\} = \frac{7}{8}$. Hence, the combined left machine is an absolutely isolated machine.

The Right Machine

Consider the combined right machine, (5.10), and apply Lemma 3.3 and Theorem 3.3.

i	1	2	3	4
α_i	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{1}{2}$	$\frac{1}{5}$
λ_i	$\frac{1}{10}$	$\frac{1}{8}$	$-\frac{2}{10}$	$-\frac{1}{10}$
β_i	$\frac{1}{10}$	$\frac{11}{40}$	$\frac{7}{10}$	$\frac{3}{10}$

$$a = \max_i \{\alpha_i\} = \frac{4}{5}, \quad c = \min_i \{\alpha_i\} = \frac{1}{5}, \quad \lambda_a = \frac{1}{8} \text{ and } \lambda_c = -\frac{1}{5},$$

This case is classified as Case 3 in Lemma 3.3,

$$S_{\max} = a \frac{1+\lambda_a}{1-\lambda_c^2} = \frac{4}{5} \frac{1+\frac{1}{8}}{1-\frac{1}{25}} = \frac{4}{5} \frac{75}{64} = \frac{1}{1} \frac{15}{16} = \frac{15}{16},$$

and

$$S_{\min} = c + a \frac{\lambda_c(1+\lambda_a)}{1-\lambda_c^2} = \frac{1}{5} - \frac{1}{5} \frac{1}{5} \frac{9}{2} \frac{25}{24} = \frac{1}{5} - \frac{9}{48} = \frac{48-45}{240} = \frac{3}{240} > 0,$$

so $R_{\min} = 0$.

Since λ_1 and λ_2 are positive, the case for the pair (1,2) is Case 1 of Theorem 3.3.

$$\frac{4}{5} - \frac{3}{5} = \frac{1}{5} > \frac{1}{5} \frac{15}{16} = \frac{3}{16} \text{ is satisfied.}$$

Since $\lambda_2 > 0$ and $\lambda_3 < 0$, the case for the pair (2,3) is Case 2 of Theorem 3.3 and no constraint is imposed.

The case for the pair (3,4) is Case 4, $\alpha_3 - \alpha_4 > -\lambda_3 S_{\max}$ should be satisfied since λ_3 and λ_4 are negative

$$\frac{1}{2} - \frac{1}{5} = \frac{3}{10} > \frac{2}{10} \frac{15}{16} \text{ is held.}$$

Similarly, adjacent pairs of β_i are satisfied with the conditions of Theorem 3.3 using $a = \max_i \{\beta_i\} = \frac{7}{10}$. Therefore, the combined right machine is also an absolutely isolated machine.

Lemma 5.2:

$$\begin{aligned} & (A_1 \otimes A_2 \otimes \dots \otimes A_n) \cdot (B_1 \otimes B_2 \otimes \dots \otimes B_n) \\ &= (A_1 B_1) \otimes (A_2 B_2) \otimes \dots \otimes (A_n B_n). \end{aligned} \quad (5.12)$$

Proof:

By the mathematical induction, $n=2$, the lemma holds as shown in Lemma 5.1. Suppose we have the following equation at $n = m-1$.

$$\begin{aligned} & (A_1 \otimes A_2 \otimes \dots \otimes A_{m-1}) \cdot (B_1 \otimes B_2 \otimes \dots \otimes B_{m-1}) \\ &= (A_1 B_1) \otimes (A_2 B_2) \otimes \dots \otimes (A_{m-1} B_{m-1}). \end{aligned}$$

The above equation is rewritten as

$$A \cdot B = C$$

where

$$A = A_1 \otimes A_2 \otimes \dots \otimes A_{m-1},$$

$$B = B_1 \otimes B_2 \otimes \dots \otimes B_{m-1},$$

$$C = (A_1 B_1) \otimes (A_2 B_2) \otimes \dots \otimes (A_{m-1} B_{m-1}).$$

By multiplying $(A \otimes B)$ on C from the right in the sense of a Kronecker product and applying Lemma 5.1,

$$C \otimes (A \otimes B) = (A \otimes A) \cdot (B \otimes B)$$

is held at $n=m$. This case is the Equation (5.12). □

Theorem 5.1 can be extended to the PSM which consists of 2^n states, $n > 2$.

Theorem 5.2:

If a PSM which consists of 2^m states, $m > 2$, is decomposable by Kronecker product (two-state) component machines and if each combined machine of the first, the second, ... the m th component machines (in Kronecker product) is an absolutely isolated machine with respect to the (1,2) or the (2,1) element, then the PSM is an input traceable machine.

Proof:


The proof is similar to that of Theorem 5.1 by using Lemma 5.2 and Corollary 3.8, so it is omitted. □

When interconnecting component machines in a Kronecker product structure are the k th absolutely isolated machines, the composed PSM is the last $k+1$ input symbol traceable machine.

Corollary 5.3:

If a PSM which consists of 2^m states, $m > 2$, is decomposable by Kronecker product (two-state) component machines and if each combined machine of the first, the second, ... the m th component machines (in Kronecker product) is at least the k th absolutely isolated machine with respect to the (1,2) or the (2,1) elements, then the PSM is the last $k+1$ input symbol traceable machine.

Proof:

This is a finite case of an infinite length input string in Theorems 5.1 and 5.2. This proof is similar to those theorems by using Lemma 5.2 and Corollary 3.9, so the proof is omitted. 

5.2 Decomposable PSM by $N(N-1)/2$ Two-State PSM's

A decomposable PSM by interconnecting component PSM's of a Kronecker product structure is presented in the previous section as an input traceable machine, if each combined component machine in a Kronecker product interconnection is an absolutely isolated machine.

The decomposition presented in Chapter IV, Theorems 4.4 and 4.5, is a parallel interconnection of component machines. From the parallel interconnection and the property of the isolated machine, a decomposable PSM by Theorem 4.5 may be an input traceable machine.

Theorem 5.4:

If a PSM is decomposable by Theorem 4.5 and if each component machine determined by the theorem is an absolutely isolated machine, then the PSM is an input traceable machine.

Proof:

Since the PSM is decomposable, there are three two-state PSM's and

each two-state PSM is an absolutely isolated machine. By Corollary 3.8, all past input strings to each component machine can be traceable. The past input strings (each component machine and the interconnection of component machine) are known. Therefore, the PSM is an input traceable machine. ■

Similarly, with Corollary 5.3, we have the following corollary.

Corollary 5.5:

If a PSM is decomposable by Theorem 4.5 and if each component machine determined by the theorem is at least the k th absolutely isolated machine, then the PSM is an input traceable machine to the last $k+1$ symbol process.

Proof:

This is an finite case of an infinite length input string in Theorem 5.4. This proof is similar to that of Theorem 5.4 by using Corollary 3.9 instead of Corollary 3.8. Therefore, this proof is omitted. ■

Remark: The original input traceable machine may be affected by an error occurrence during machine operation. In other words, after an error occurs the character of the machine changes. Therefore, the eigenvalue of the state transition matrix of the machine may be increased in the magnitude, hence, the property of the absolutely isolated machine may be lost. However, as long as the absolutely isolated property is held, the machine is an input traceable machine and is a diagnosable machine by the playback process.

Generally speaking, necessary conditions of the input traceable property; (1) an N -state PSM is decomposable by two-state component machines and further more (2) all the decomposed component machines are absolutely isolated machines, are fulfilled by a class of N -state PSM's. In other words, appli-

cability of the input traceable property is limited within the class. More powerful technique for estimating the past performance of an N-state PSM is presented in the following chapter.

Chapter VI

ERROR DETECTION IN A LARGE COMPUTER SYSTEM

6. Introduction

When correcting an error in hardware or software, systematic and immediate detection of the subsystem which possibly contains the error is very helpful. We shall present a statistical estimation technique to economically find an erroring subsystem[†].

The statistical technique has advantages when compared with the trial and error procedure; the advantages can be demonstrated by the following example. Suppose that a system has ten subsystems and that one subsystem contains an error. A maintenance engineer has ten checkboards to detect the error for each individual subsystem. He picks a subsystem to check. If this subsystem is error free, then he picks another subsystem and so on. He continues to check subsystems until the subsystem which contains the error is found. The order of choosing subsystem is random, and no prior knowledge of error occurrences exists. Neither the frequency of the past error occurrences nor a diagnostic table for error symptoms is used. Thus the engineer assumes the equal probability of error occurrence in each subsystem, namely 1/10, and therefore the expected value to find the subsystem which contains the error is:

$$E[n] = \sum_{i=1}^{10} i \frac{1}{10} = 5.5.$$

Over a long maintenance time span, the average time to find the erroring subsystem is 5.5 by the trial and error method.

[†]It is assumed that error occurrence may be proportional to subsystem activity in hardware. In software, it should be said performance measurement of subsystem activity.

The proposed statistical technique provides a probability distribution of each subsystem's activity. Suppose the engineer has the probability distribution (0.08,0.03,0.21,0.05,0.04,0.02,0.19,0.07,0.3,0.01) for activities of subsystems 1,2,...,10. He checks the ninth subsystem first because of its largest probability 0.3. If the ninth subsystem is error free then he must pick the third subsystem, because it has the second largest probability 0.21 and so on. The expected value is found as follows:

$$\begin{aligned} E[n] &= 0.3+0.21 \times 2+0.19 \times 3+0.08 \times 4+0.07 \times 5+0.05 \times 6 \\ &\quad +0.04 \times 7+0.03 \times 8+0.02 \times 9+0.01 \times 10 = 3.04 \end{aligned}$$

using the statistical technique, the average time to find the subsystem which contains the error is 3.04 in a long maintenance time span.

Since the working cost of a large computer system is very expensive, a reduction in computer downtime (in the example downtime reduction from 5.5 to 3.04 in ratio) not only improves computer efficiency but also decreases the loss of manpower due to computer downtime.

In Section 6.1, an iterative process of model building (analysis of a computer system, design of a model, testing of the model, reanalysis, redesign, and retesting to improve the model) is emphasized. Techniques to estimate the most active subsystem (in which error occurrence may be proportional to activity) are presented. An optimization to estimate the most likely active subsystem is mathematically formulated; the optimization becomes a nonlinear mathematical programming problem which is described in Section 6.2.

An application of the estimation techniques developed in this research to a real computer system is presented in Sections 6.3 and 6.4. In Section 6.3, a model of a computer system is built. In the last Section, daily computer operation is simulated. The optimization of the nonlinear mathematical programming is applied on the simulated result in order to find an order of subsystem activities.

6.1 Model Building

Modeling a computer system with a Moore type generalized probabilistic sequential machine is discussed in Chapter II from a viewpoint of multi-levels and hierarchy structures of the system. Some difficulties in modeling are emphasized in Section 2.7. In this Section, modeling is discussed from another viewpoint how the difficulties can be overcome.

Large scale, complex system analysis and modeling do not lend themselves to a conventional approach. There are no overall methods to be used. There is no clear place from which to start or end. A system behaves, however, according to a set of relations. This is the precise point of attack. Even though initially the rules for interrelationships may not be consistent, they form the basis for model structure.

A model must start somewhere. Since there are numerous choices, it is difficult to develop a model without going through a process of setting down hypotheses and abstractions. A representation of the system - the rules and relationships that describe it - is defined as the model. In a sense, the model becomes an algorithm, a procedure statement of the problem, or a set of numerical equations. Once a coarse model of the overall system concept is established, the model becomes a definition of the system problem and a working tool.

Adequate data for a model are usually not available but a model under investigation needs (1) data, (2) system constraints, and (3) validation. While the GIGO principle (garbage in, garbage out) is well accepted, something can be learned from "garbage in". The input data, while not verified, can be made to represent the range of possible values. Insight is gained in spite of the lack of adequate input data. The results can be fed back to the system analysis and modeling. This process is definitely cyclic, and the series of steps overlap each other.

In a formulation of the system problem, there must be a statement of objectives involved, since formulating the problem without formulating the objectives may mean solving for system symptoms only. It should be emphasized that the objectives can not be treated separately from the model formulation. In some cases, moreover, the idea of a model and the idea of an objective are completely confounded, and the form by which the model is stated is in fact that of the objective itself. The interrelationship of the objective and of the model is subtle and intricate, but we have to be quite clear about the way in which they differ and to understand the way in which they interact with each other.

We are concerned, in model building with some understanding of reality. Reality is observable, measurable, and systematic. Observable means the ability to see some aspects of what is happening, to try and understand the characteristics of these aspects, and to be able to build some form of prediction not only of behavior but also to compare the results of this hypothesis by making further observations with actual performances of the subject.

Another criterion of reality is that certain aspects should be measurable. In some cases, measurability is not possible; however, measurability should be able to estimate some events in common terms.

The third aspect of reality is a set of causes and effects interacting simultaneously in a complex manner. The sets of causes and effects which can be allowed to operate on actual systems can be numerous and in general more than we can conveniently handle or would like to handle. We have to have a meaningful way of classifying these causes and effects in groups.

The system operation is continuous in time with discrete states of the system, and we are able to describe the present state of the system at any time. The richness involved is such that we cannot hope to consider and tabulate every bit of the system's information, and hence, the system may be classified rather than described. The sets of causes, states, and objectives are linked together by means of basic assumptions, conclusions, and a model.

The objectives are admitted in order that we can devise a useful and meaningful hypothesis which can often be formulated as a qualitative statement of the system problem. The qualitative statement gives an outline of what is termed a model. A set of states of the model can be derived from the set of causes (or from reality) by prediction. Then the behavior of the model is compared with reality. Clearly there is never an exact correspondence between them. Discrepancies can be determined by direct experimentation on a small scale or by observation on a large scale. There

are three ways to judge a discrepancy: (1) it is within an acceptable level, (2) it may be explained by chance causes depending on the criterion for acceptance, and (3) it is too large to be acceptable. The last case must return to the previous stage in order to improve the model.

In early stages of model building, we may often neglect to consider what the objectives are. Frequently, objectives are not realized until the end of the process of model building. If this is the case, then looping back of the process has to take place so that objectives can be reformulated.

We shall build a GPSM model of computer system with the remaining of the process on how to overcome difficulties of model building. From the reality of observable and measurable system behavior, input job types and output activities of the model are determined. Internal states of the model are defined by considering causes and effects of the system.

The next phase is to determine the steady state distribution and output activity distribution for each individual input job type. Typical benchmark computer programs which belong in each job type are selected as real input symbols of the model or the GPSM. Each set of benchmark programs which represent a job type is executed in the computer. During execution, the performance of the computer system is monitored by the software package or hardware instrument in order to decide patterns of these two distributions. Execution of sets of all job type benchmark programs produces a set of steady states and output activity distributions, which is the result of a calibration of the system performance. This phase is called the calibration phase. The calibration data is used in an optimization process described in Section 6.2.

Assume that output activity distribution during normal computer operation is available. When an error (or a default) has occurred in the computer operation, estimation of the most likely active subsystem (to which the probability of error occurrences may be proportional) could help maintenance personnel quickly to find the error. The estimation is an optimization process using the output activity distributions in normal operation and in the calibration phase. The result of the optimization is the most-likely mixing ratio of the input job types. Next, the summation of the steady states weighted by the most-likely mixing ratio for each input job type indicates the past activity of each subsystem. The estimation can be improved by combining prior knowledge of error frequency and the other estimation techniques; for example, maximum likelihood estimation and Bayes estimation, etc. The calibration phase and the estimation phase of subsystem activity distributions are shown in Figures 6.1 and 6.2, respectively.

Determination of the state transition probabilistic matrix for each input job type and of the transfer matrix of states versus output activity are not necessary to detect the most-likely active subsystems; however, this determination may be useful in GPSM model simulation of the behavior of the computer system.

Given the matrices in the terms of GPSM, S is the internal states and, X is the input job types, Y is the output activity, $M(x)$ where $x \in X$ are the state transition probabilistic matrices, and T is the transfer matrix of state versus output activity. There are several ways to determine the matrices. For example, the matrices could be estimated directly by analysis of the operating system by guessing required services from individual input

Figure 6.1 The Calibration Phase

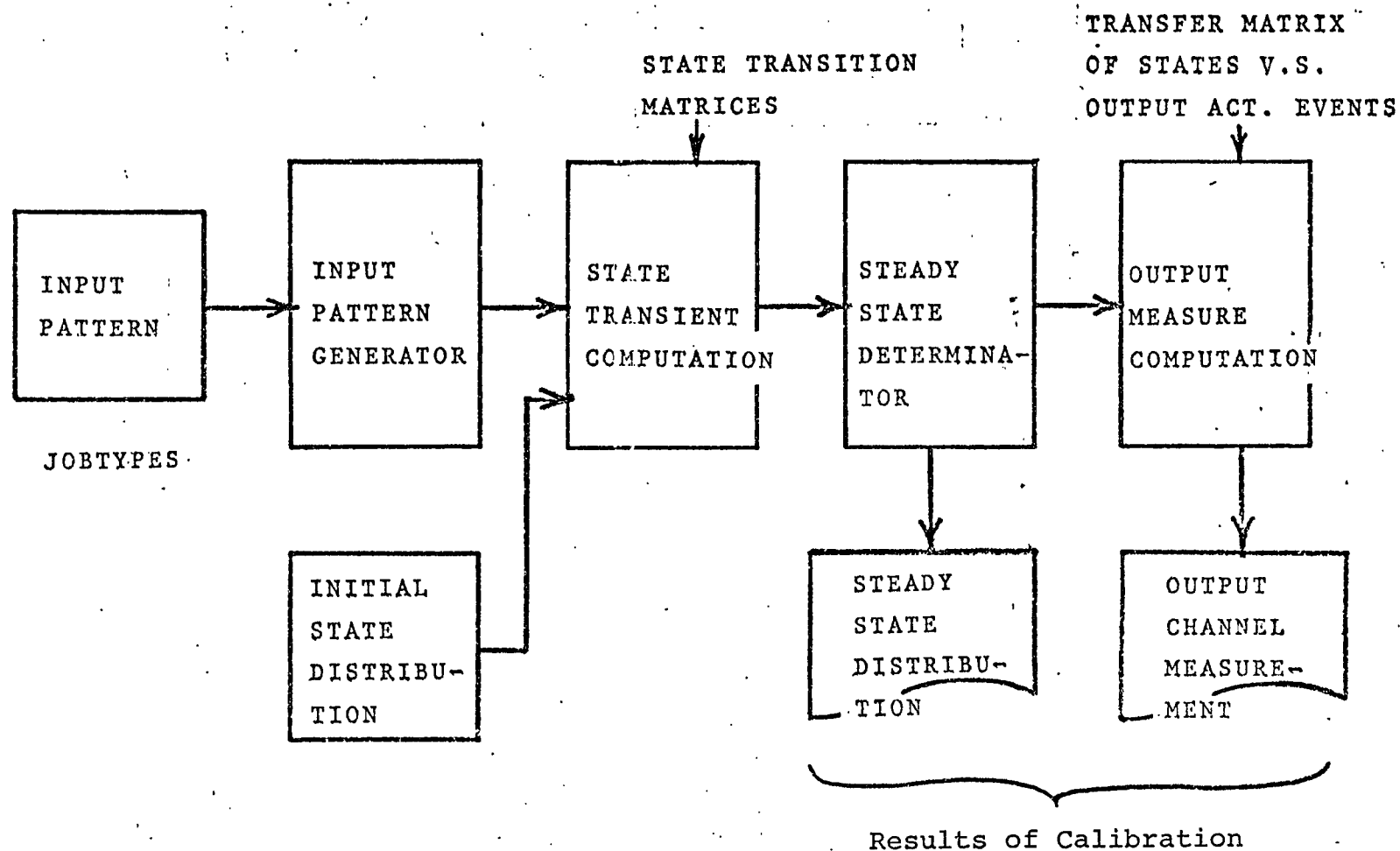
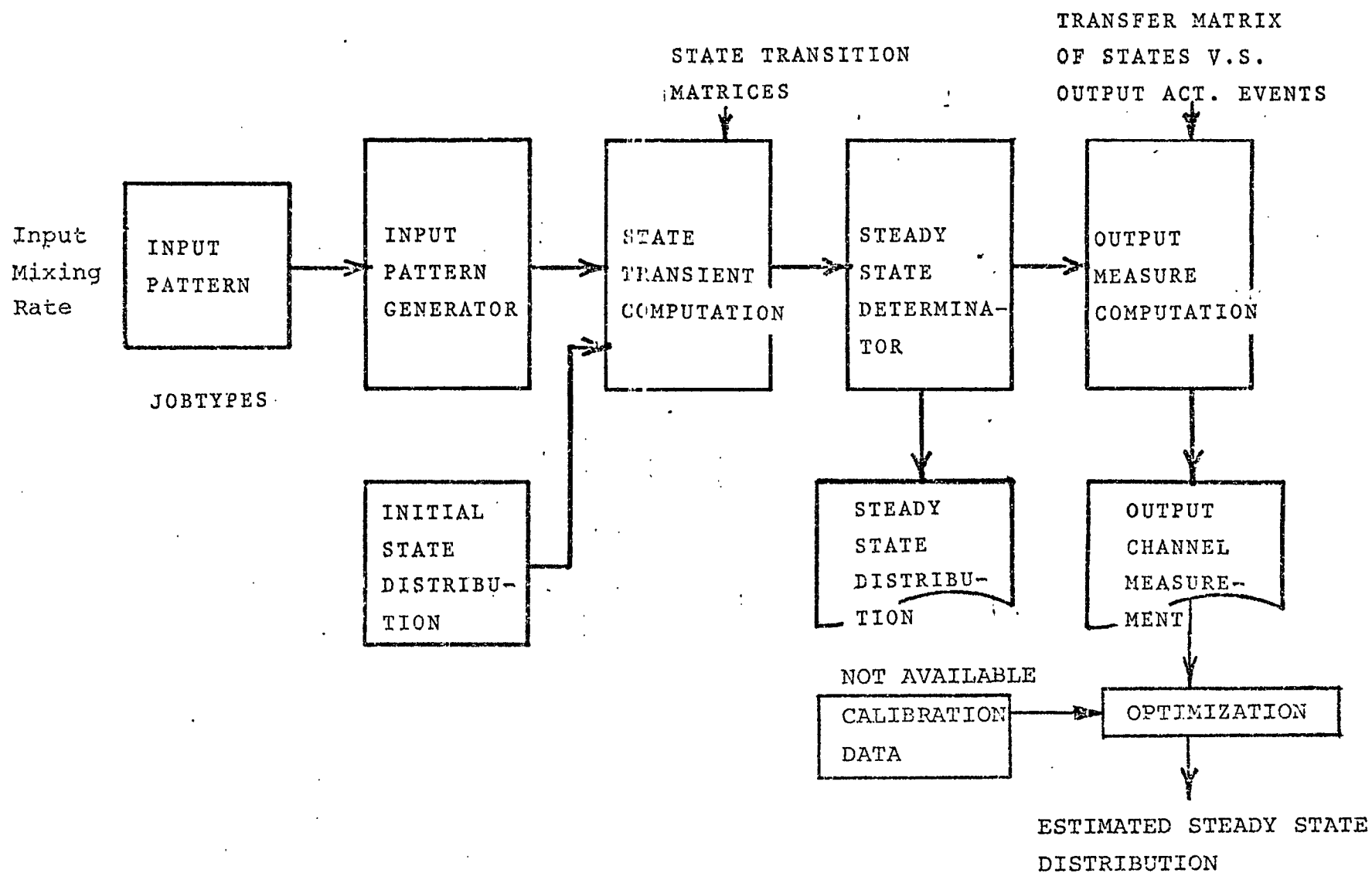


Figure 6.2 Estimation Phase of Channel Activities



job types, and by collecting statistics of the system behavior from system monitoring (software or hardware monitor) during the calibration phase. Also, normal system operation can be used to form the matrices. When statistics are available, the problem is to determine the unknown entries of the matrices using several observed operational points of system behavior; the least mean square method (or other curve fitting techniques) can be applicable to establish the matrix entries.

In general, the hardware measurement technique easily provides information as to what happened in a system, and the software measurement technique easily provides information as to why something happened in a system. Most of the software techniques have the attribute of inserting themselves into the normal flow of programming execution courses to obtain the required information, but almost all data acquisition may be performed with software techniques. Due to the insertion, software measurement techniques do use facilities of the host system. They take not only Central Processing Unit (CPU) time to execute, but also space in processor storage, space in auxiliary storage, and time to read out the results of the measurement. These system facility requirements are the principle disadvantage of software measurement, which causes a disproportionate interference with normal system operation.

Principal reasons for utilizing hardware measurement techniques are ease of use, removal of system overhead by measurement, and the ability to obtain data in a way which does not interfere with the workload in process by the host system. However, one of the most critical considerations in the application of hardware techniques is a design of the interface

connection between the host system and measurement devices. The functional capability of hardware techniques is based on types of information that could be obtained from the host system. A general characteristic of the interface is that signals in the host system are either sensed or derived by combinational logic within the system itself and then driven through an appropriate signal cable connecting to a measurement device. This kind of circuitry is necessary to sense as well as amplify for transmission is quite time consuming and costly to construct in circuit locations of the host system. Furthermore, when considering various functions that the hardware techniques can perform, it would not be unusual to find interface requirements of over 200 signal lines. This is the case if memory address generation is distributed over several functional areas. Complex and long wiring requirements in this situation may cause signal interferences with the normal circuit operation.

A data reduction process is usually needed on collected raw data in order to produce essential and required sets of measurement data in both software and hardware techniques.

The following basic assumptions for the model building are imposed from the proceeding discussion:

1. Estimation or collection of data in output channel activity is easily accomplished by observation or by monitoring output channels in the calibration phase and the estimation phase.
2. Estimation or collection of steady state distribution could

be accomplished in the calibration phase because of: (a) relatively short time executions and (b) known predefined services required by benchmark programs.

3. Collection of steady state distribution is almost a prohibitive process in regular system operation due to the time consuming and very expensive interface in hardware measurement techniques and due to large interferences (unacceptable system overhead and heavy influences to computer workload) in software measurement techniques during long term system operation.

The following special case (in which a system holds the input traceable property) does not need to assume the restrictions described in the above:

Application of Input Traceable Property

If the modeled GPSM or all state transition probabilistic matrices are decomposable into a set of two-state component machines as described in Chapter V and if the two-state component machines are absolutely isolated machines, then the GPSM is an input traceable machine. Thus the sequence of the past input job types supplied to the system can be determined as well as the histories of state distributions of component machines so that the history of the state distribution of the GPSM is revealed. From the history, the past subsystem activity distribution of the GPSM can be computed; the computed distribution could be the same as the estimation derived from the optimization. There may be some discrepancy between the results of the two calibrations of the computer system with the set of the benchmark programs taken before and after an error occurred. This discrepancy of the two calibrations may be caused by the error. Therefore, the discrepancy is a clue to determining the state or subsystem which contains the error.

6.2 An Estimation of the Most-Likely Active Subsystem

When a computer system is calibrated with n job types, the results of the calibration are (1) n steady state distributions and (2) n m -output activity distributions for each input job type. Let the steady state distributions and the m -output activity distributions be:

$s_{i1}, s_{i2}, s_{i3}, \dots, s_{il}$ and $e_{i1}, e_{i2}, e_{i3}, \dots, e_{im}$ for $i=1, 2, \dots, n$, respectively, where

l is the number of internal states (or subsystems) of the computer system, m is the number of output activities (or output channels) and n is the number of input job types. The output activities as denoted by: $P_r(E_j | x_i) = e_{ij}$ are

normalized by numerical quantities A_i , where x_i and E_j are the i th input and the j th output activity, respectively. The probability e_{ij} can be tabulated

as input x_i versus output activity E_j . Note $\sum_{j=1}^m e_{ij} = 1$ for all i 's.

Given an output activity distribution, denoted as a_1, a_2, \dots, a_m

and measured during a normal computer operation, an optimum job type distribution $\Pi_1, \Pi_2, \dots, \Pi_n$ produces the closest job type mixing ratio with the minimum difference between a_1, a_2, \dots, a_m and $B_1 P_r(E_1), B_2 P_r(E_2), \dots, B_m P_r(E_m)$ where

$$P_r(E_j) = \sum_{i=1}^n \Pi_i e_{ij}, \quad (6.1)$$

and B_j for $j=1, 2, \dots, m$ is a factor defined in (6.7).

The probability $P_r(x_i | E_j)$ is found as:

$$P_r(x_i | E_j) = \frac{\Pi_i e_{ij}}{P_r(E_j)} \quad (6.2)$$

We introduce an unknown probability r_{ij} which constitutes

$$\Pi_i = \sum_{j=1}^m r_{ij} \quad \text{for } i=1,2,\dots,n, \quad (6.3)$$

$$p_r(E_j) = \sum_{i=1}^n r_{ij} \quad \text{for } j=1,2,\dots,m \quad (6.4)$$

and
$$\sum_{i=1}^n \Pi_i = \sum_{i=1}^n \sum_{j=1}^m r_{ij} = 1. \quad (6.5)$$

A mathematical formulation of the optimization becomes a nonlinear mathematical programming problem to find the most likely input job type, mixing ratio of a set $\Pi_1, \Pi_2, \dots, \Pi_n$.

$$\text{Min}_{r_{ij}} \sum_{j=1}^m \{a_j - B_j p_r(E_j)\}^2 = \sum_{j=1}^m \{a_j - B_j (\sum_{i=1}^n \Pi_i e_{ij})\}^2 \quad (6.6)$$

subject to (6.5), where B_j for $j=1,2,\dots,m$ is defined in (6.7).

Note $r_{ij} \geq 0$ for all i 's and j 's since r_{ij} is a probability. In order to find the optimum mixing ratio Π_i , $i=1,2,\dots,n$, the Lagrange multiplier method is used to solve the nonlinear mathematical programming problem easily.

$$B_j \sum_{i=1}^n e_{ij} \Pi_i = \sum_{i=1}^n A_i e_{ij} \Pi_i \quad \text{for } j=1,2,\dots,m \quad (6.7)$$

where A_i is the normalized factor of the output activity distribution of the i th job type.

The nonlinear mathematical programming problem described in (6.5) and (6.6) is optimized by the Lagrange multiplier method which is shown here:

$$\text{Min}_{\Pi_i} f = \sum_{j=1}^m \{a_j - \sum_{i=1}^n A_i e_{ij} \Pi_i\}^2$$

subject to

$$h = \sum_{i=1}^n \Pi_i - 1 \text{ and } \Pi_i \geq 0 \text{ for } i=1,2,\dots,n.$$

where m and n are numbers of output activity channels and of job types in computer system modeling, respectively. Define a function $g = f + \lambda h$.

Calculate Jacobian of g with respect to $\Pi_1, \Pi_2, \dots, \Pi_n$ and λ . The minimum of

g exists at the point which is the solution of the following equations:

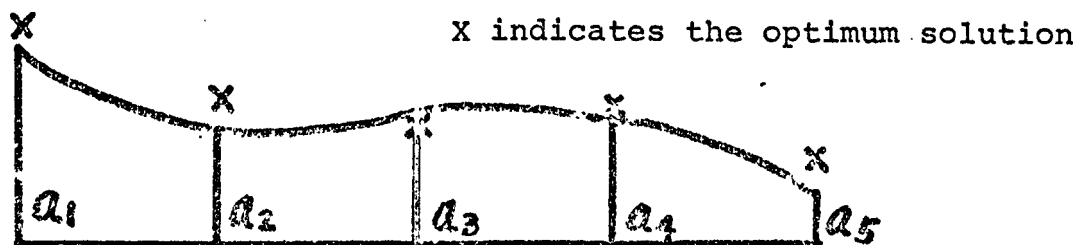
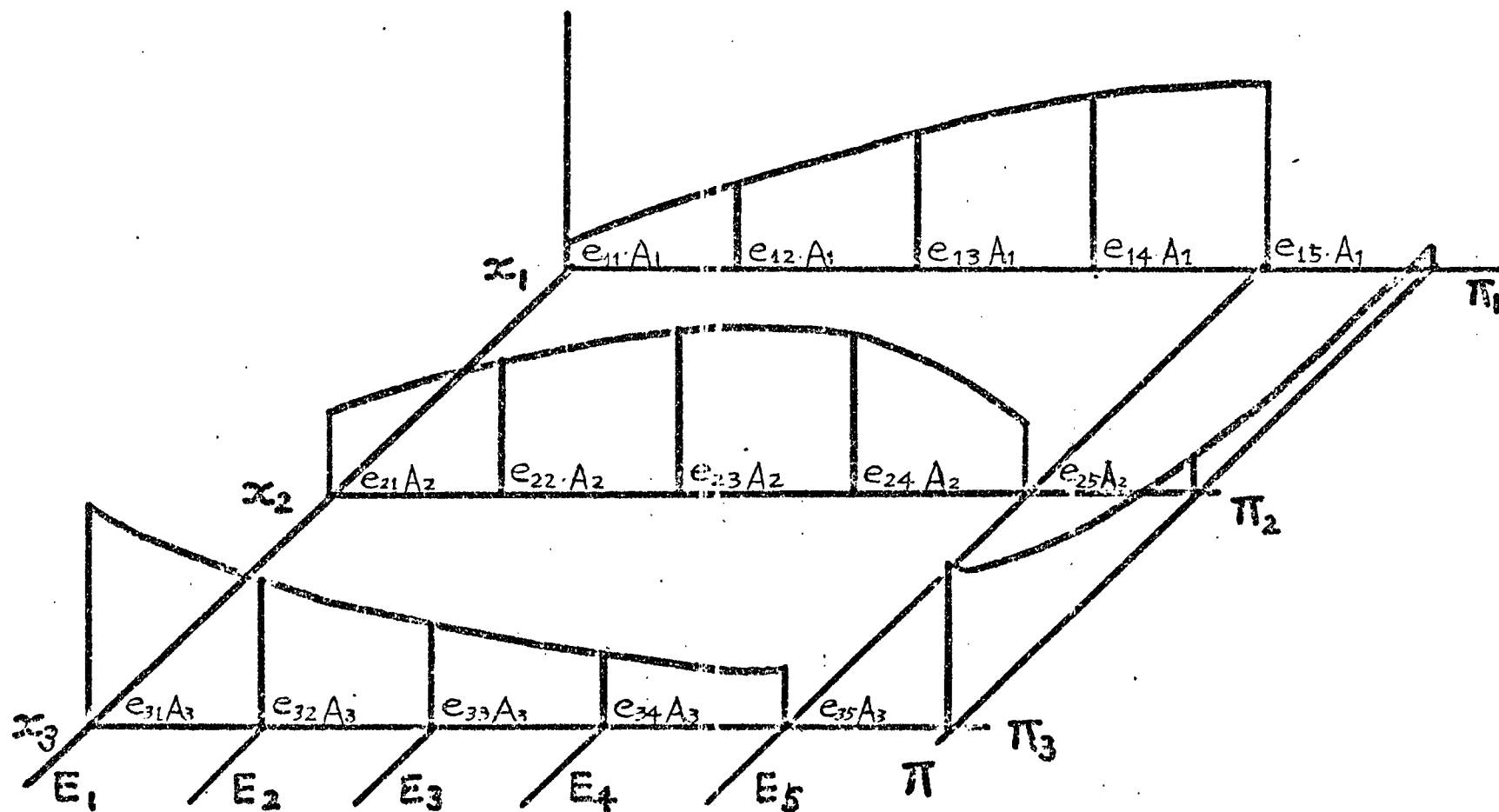
$$\begin{aligned} \frac{\partial g}{\partial \Pi_i} &= 2 \sum_{j=1}^m \{ a_{ij} - \sum_{k=1}^n A_{kj} e_{ik} \Pi_k \} (-A_{ij} e_{ik}) \\ &= 2 \sum_{j=1}^m A_{ij} e_{ij} \left(\sum_{k=1}^n A_{kj} e_{ik} \Pi_k \right) - 2 \sum_{j=1}^m a_{ij} A_{ij} e_{ij} = 0 \text{ for } i = 1, 2, \dots, n \\ \frac{\partial g}{\partial \lambda} &= h = 0 \end{aligned}$$

Thus we have $n + 1$ unknown variables, $\Pi_1, \Pi_2, \dots, \Pi_n$ and λ and $n + 1$ equations in Jacobian of g . The unique solution of the simultaneous equations can be found by the Gauss-Jordan method. The unique solution may not satisfy the other constraint, $\Pi_i \geq 0$ for all i 's.

The function g is a linear combination of the functions f and h , and the functions f and h are quadratic and linear, respectively. Therefore, the surface of g is unimodal. When a Π_k among the solution of the simultaneous equations is negative, the optimum value of the minimization along with the Π_k axis is zero because of the unimodal characteristic. The rest of Π_i is needed to compute again the simultaneous equations without the Π_k term (or by forcing $\Pi_k = 0$). Computing examples of the optimization shall be shown in Section 6.4.

An illustrated example of an optimization is in Figure 6.3. Three job types and five output activities are assumed in the example. The input

Figure 6.3 An Example of an Optimization



job type mixing ratio is given as: Π_1, Π_2, Π_3 for job types x_1, x_2, x_3 and the

optimum solution is indicated with 'X's' in the figure.

After the optimum solution, which is the optimum mixing distribution $\Pi_1, \Pi_2, \dots, \Pi_n$ of the job type is found, the most likely active state (or subsystem) in the past normal computer operation can be determined by calculating the state sum distribution $s_1^*, s_2^*, \dots, s_\ell^*$ of the calibrated steady states weighted by the optimum mixing distribution:

$$(s_1^*, s_2^*, \dots, s_\ell^*) = (\Pi_1, \Pi_2, \dots, \Pi_n) \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1\ell} \\ s_{21} & s_{22} & \dots & s_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & s_{n\ell} \end{pmatrix} \quad (6.8)$$

The maximum s_i^* among $i = 1, 2, \dots, \ell$ is the most likely active state in the past computer operation based on the past output activity distribution a_1, a_2, \dots, a_m and the calibration results, $s_{i1}, s_{i2}, \dots, s_{i\ell}$, and $e_{i1}, e_{i2}, \dots, e_{im}$

for $i = 1, 2, \dots, n$. The second largest s_i^* among $i = 1, 2, \dots, \ell$ is the second most likely active state and so on.

6.3 An Example of Computer System Modeling

In Section 2.7, the computer model used is a generalized probabilistic sequential machine. The computer system in Figure 2.6 is a block diagram of the UNIVAC 1110 EXEC8, and we shall continue to use this system as a discussion example. Another block diagram of the system is depicted in Figure 6.4, and the connectivity matrix of the figure indicating the logical execution course of the computer system is shown in Table 6.1. As emphasized in Section 2.1, the probabilistic property of a computer system can be due to unknown job sequence, unknown job characteristics of required services, asynchronous interruption, interlocks of hardware resources, and exhaustion of a given time slice for a task. Random state transition from one state to another in which both states are not connected directly in the connectivity matrix may happen due to characteristics of the computer system operation.

For simplicity, we shall reduce the number of the states of the computer system from 34 to 16 by reasonable combination of a few states. The reduced states in a graph is shown in Figure 6.5. User activity in Figure 6.4 is split into two sections, the user's area and the user's ER (Exec Request) [72], in the new Figure 6.5.

We shall use system monitoring data of EXEC 8 computer systems provided by UNIVAC [74], [75]. The data are collected from two computer systems, designated the "A" and the "B". The two computer systems are differentiated by the applications they support. The "A" system processes most of the scientific programs in batch mode only. The "B" system processes most of the commercial (business) workload and three time-sharing/online programming real-time mode. Figures 6.6, 6.7, 6.8, and 6.9 were obtained by continuously monitoring 24-hour periods arbitrarily defined to start at or before 1200 hours daily. The explanations for the figures are quoted from the UNIVAC reports.

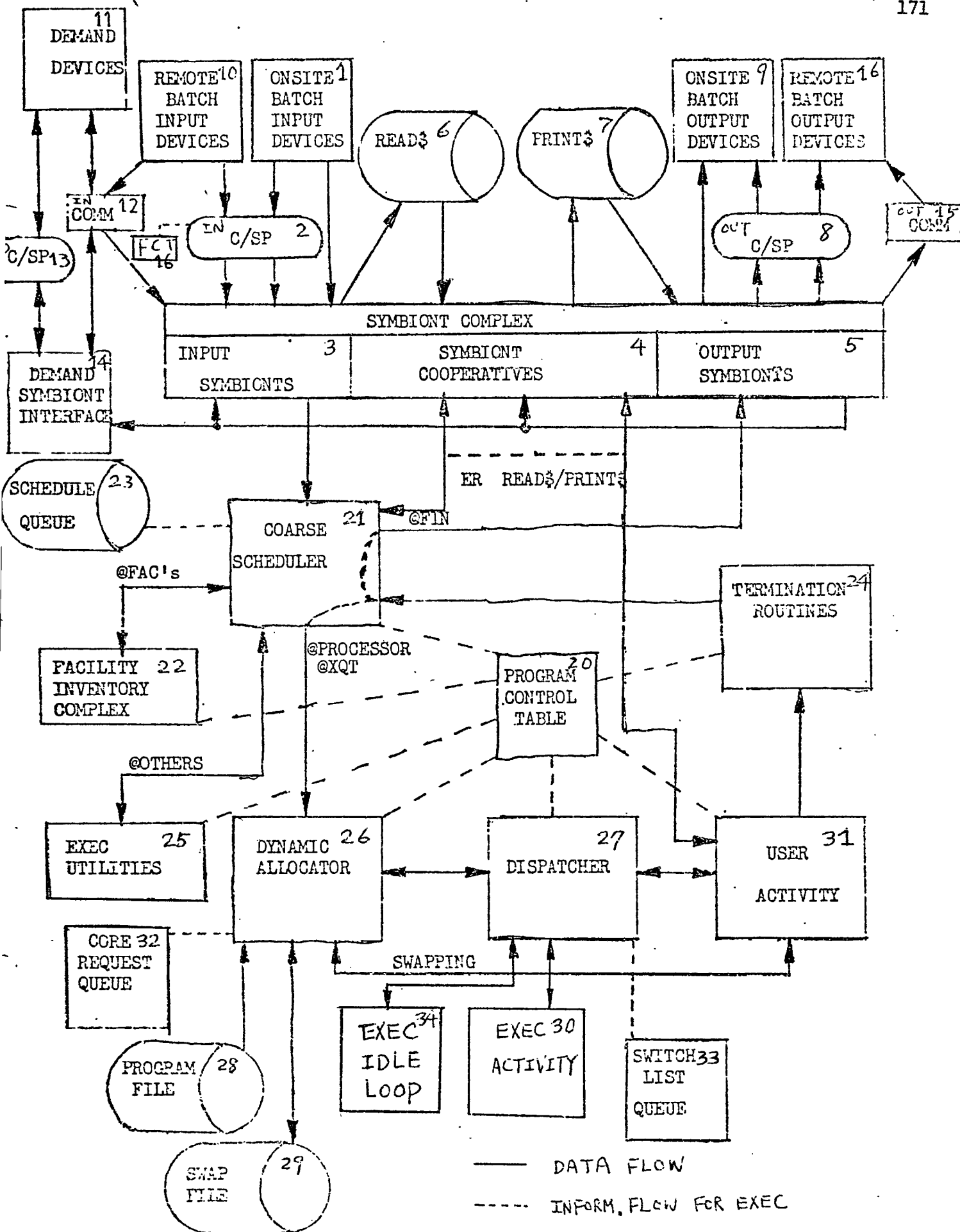


Figure 6.4 Thirty four States of a Computer System

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
1		1	1																															
2			1													1																		
3																						1												
4							1															1												
5			1	1				1	1					1	1																			
6																																		
7							1																											
8																	1																	
9																																		
10		1												1																				
11														1	1																			
12			1																															
13																																		
14																																		
15																																		
16		1																																
17																																		
18																																		
19																																		
20																								1		1	1	1				1		
21					1	1															1			1	1		1	1						
22																																		
23																																		
24																																		
25																																		
26																																		
27																																		
28																																		
29																																		
30																																		
31				1																	1				1			1	1					
32																																		
33																																		
34																																		

Table 6.1 The Connectivity Matrix of the Computer System

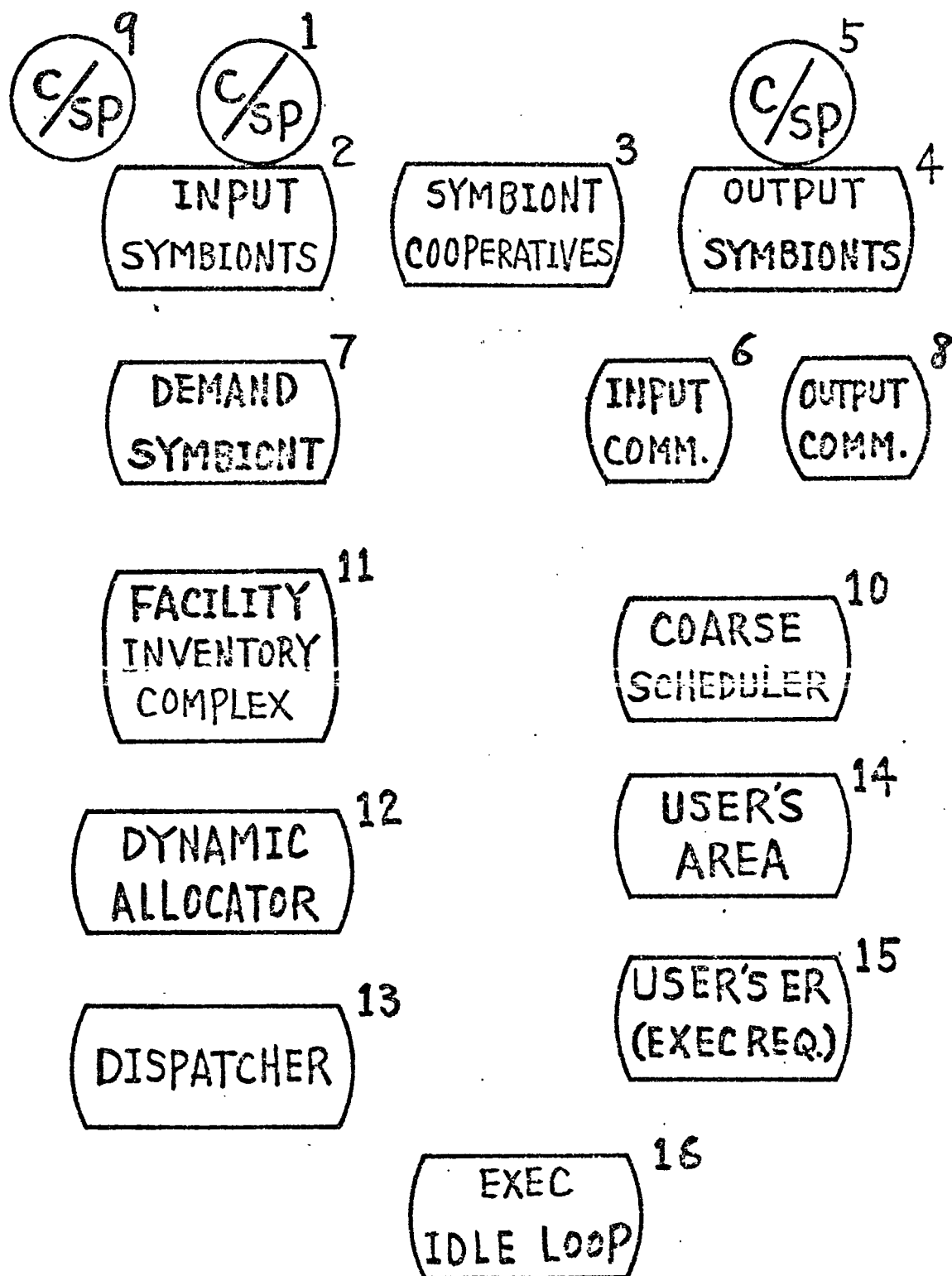


Figure 6.5 Sixteen States of a Computer System

Figure [6.6] is representative of CP (Central Processor) behavior on the "A" system. It is an hourly plot of CP utilization from 1000 hours on October 19 until 1000 hours on the 20th. The upper curve traces the percent of wall time that the CP is executing any code except the idle loop in the exec. The lower curve traces the percent of wall time that the CP is executing EXEC 8 code. The difference between the two curves, then, is the percent of wall time that the CP is executing applications program code commonly known as guard mode time. Note that the hourly CP idle value is the difference between a hypothetical horizontal line drawn through the "100%" value on the left and the top curve. For example, the hour between 1000 and 1100 can be accounted for as follows: 20% of the time the CP was executing EXEC 8 instructions; about 67% of the time it was executing applications program instructions; therefore, the CP remains about 90% utilized--save the valley at 1930 hours. The Dynaprobe data (which is a UNIVAC hardware monitoring system) for that interval indicated operator intervention; but, the major cause of the dip was most likely forced processor idle due to FASTRAND I/O (file system loading).

While Figure [6.6] establishes an ideal processor profile for a batch system, Figure [6.7] depicts something less desirable. The data presented in this plot was collected during October 5 and 6 when turnaround for the large FORTRAN application mentioned earlier was waning. Like the previous time series, it is an hourly summary of CP activity. In the first 10 hours, it approximates the series in Figure [6.6]. However, after 2200 hours, the two series have little in common. For 12 hours or so the "A" system exhibits extremely cyclical behavior. What is captured here are extreme periods of forced processor idle due to the 8460 Disc subsystem which (is hardware of file systems.)

Both the business computation in batch mode and the time sharing/on-line of user real-time applications co-exist on the "B" system. During normal working hours, however, the batch processing which comes from a card reader is superseded due to the requirement of fast response time of the time sharing/online application to remote terminals. A small number of remote batch jobs can be executed in the working hours by issuing a "@START" EXEC 8 command [76].

(Figures 6.8 and 6.9) supply sufficient evidence to the contrary. In both figures, the CP utilization is plotted on an hourly basis from 0600 hours one day to 0600 hours the next. The lower curve traces the percent of elapsed time that the CP is executing EXEC 8 code. The upper curve traces the percent of elapsed time that the CP is executing any code except the idle loop in the Exec. The difference between the two curves, then, is the percent of time that the CP is executing instructions from applications pro-

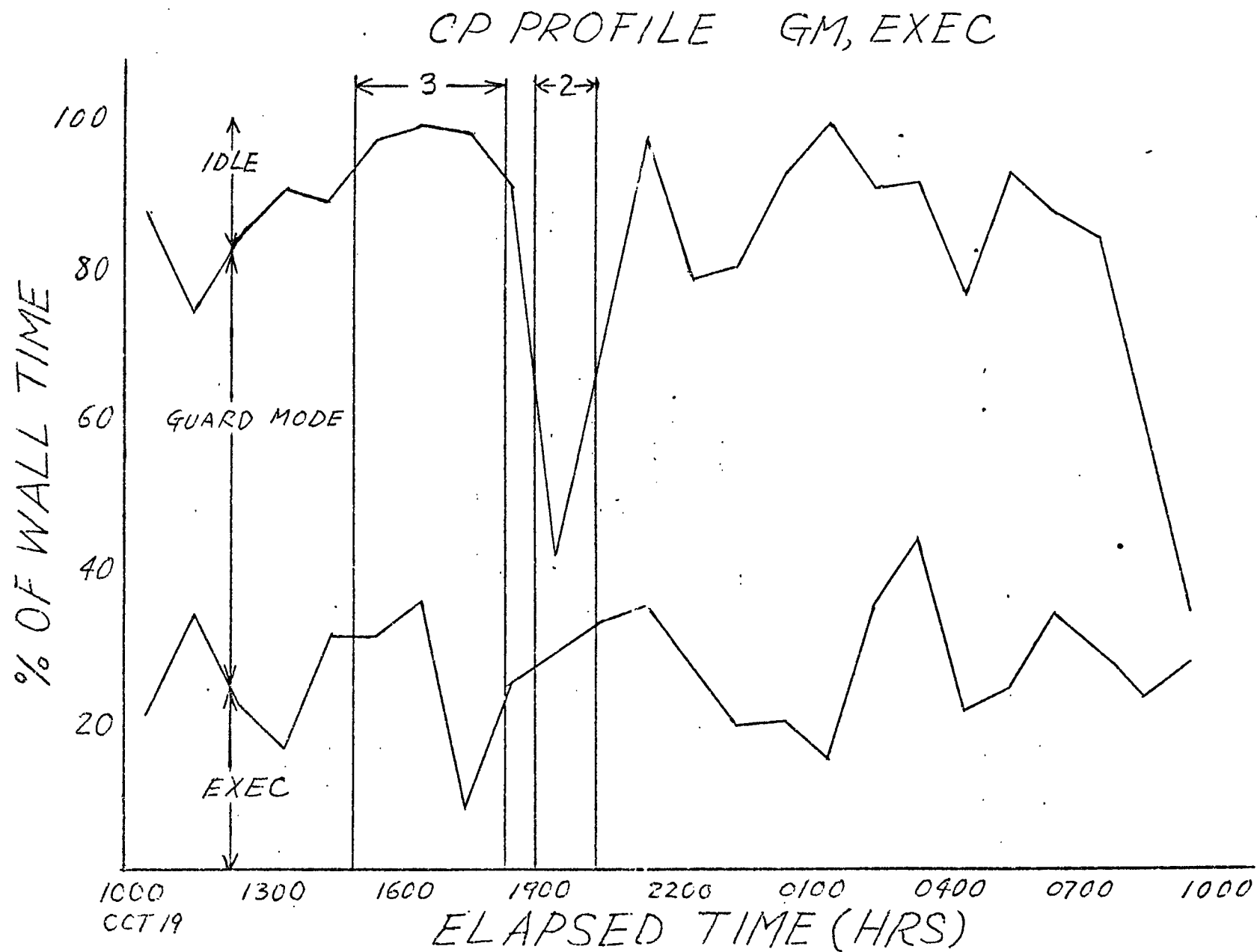
grams. Note that the hourly CP idle value is the difference between a hypothetical line drawn through the "100% busy" level and the top curve. In Figure [6.8] for instance, the hour between 0600 and 0700 can be accounted for as follows: 15% of the time the CP was executing EXEC 8 instructions; about 20% of the time it was executing applications program instructions; therefore, the CP was busy about 35% of the hours.

The CP shows some very definite trends in Figure [6.8] that have not been discussed previously. Starting from a reasonably idle state in the morning, the CP utilization climbs steadily from 0730 hours to 1400 hours. It begins to trail off until about 1630 hours. At that point, a rather significant change takes place and the CP climbs to a totally saturated state in three hours. It reacts slightly at about 2400 hours. A surge is made again, but soon the CP activity is trailing off in the morning hours.

Such behavior can be explained quite readily. First, the time-frame of 0300 hours to 0700 hours is a period in which the system is often idle. At about 0700 hours, a spike in the utilization could have been caused by the initialization of real time files and programs and/or to the commencement of ESI (communication line) network management functions (time sharing.) More likely than not, this transient was due to some contingency since a similar spike does not appear in Figure [6.9] at that time. The steady rise in CP activity during the morning and early afternoon hours is closely correlated to the real time transaction load. It increases as more terminal users come on line. Similarly, it trails off toward 1630 hours as users finish making their daily inquiries and sign off. However, a more substantial reason for this particular trail off can be identified. Figure [6.8] is drawn from data collected on a Friday/Saturday boundary. In the initial description of the shop schedule, it was stated that all real time applications are removed at 1630 hours on Friday. Since a critical checkpoint of the master real time files is taken immediately thereafter, batch runs are constrained from entering the system at that time. At the completion of the checkpoint, there is an additional 123,000 words of allocatable program space for batch work. Eventually the system becomes totally CP bound three hours into the batch production period. The drop at 2400 hours is probably due to a shift change. Finally, the drop after 0100 implies the machine is running out of work to do.

The time series in Figure [6.8] is somewhat different from that of Figure [6.9]. In the latter, the total CP utilization, upper curve, shows no tendency to decline in the late afternoon. Thus, neither the lull at 1630 hours nor the fantastic differential between the real time and batch only processing periods is evident. In both sets of curves, however, the EXEC busy line, the lower plot, shows the characteristic day time rise/fall. The explanation for this observation lies again with the oper-

Figure 6.6 CP Profile of EXEC 8 on the "A" System



CP PROFILE - GM, EXEC

Figure 6.7 CP Profile of EXEC 8 on the "A" System

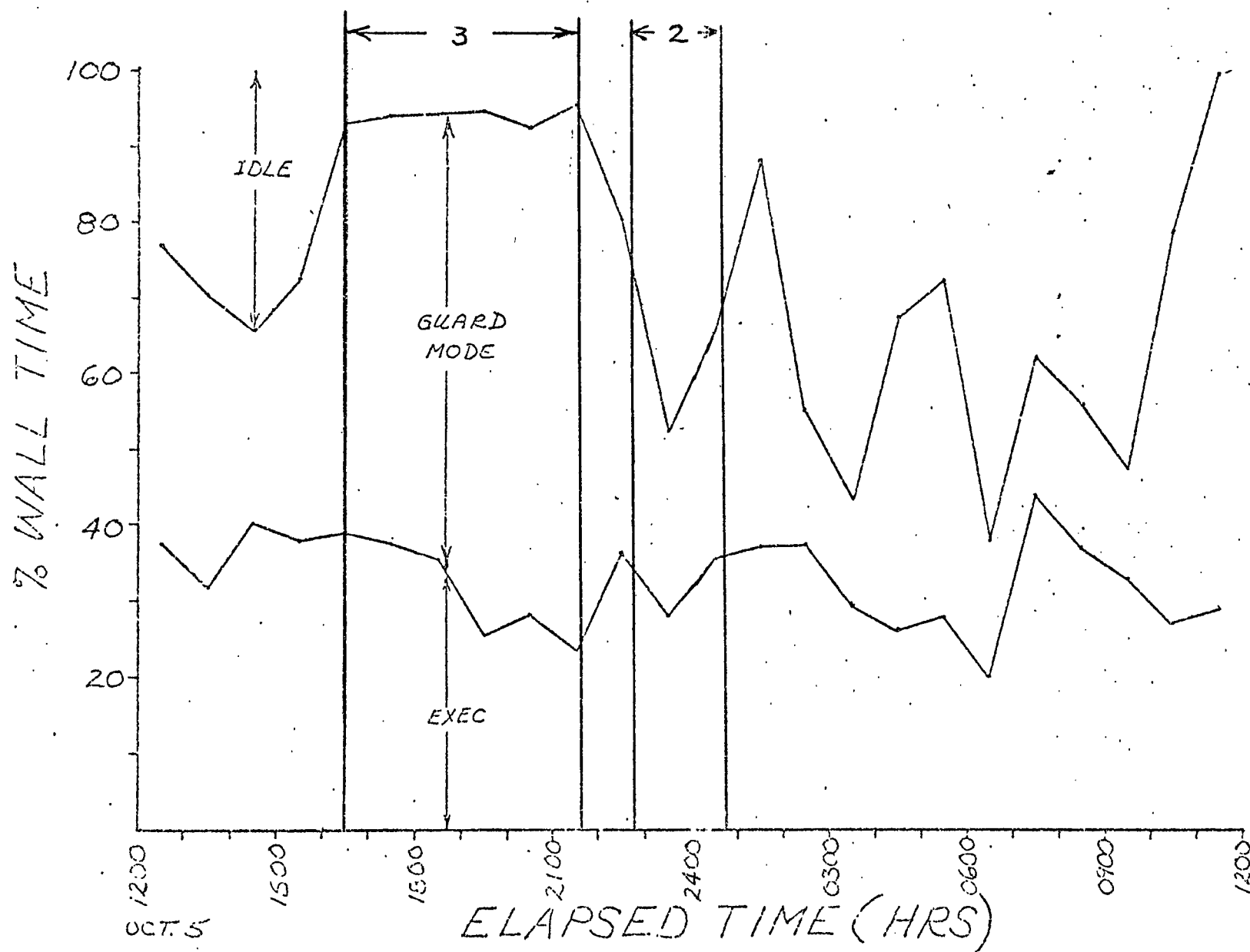


Figure 6.8 CP Profile of EXEC 8 on the "B" System

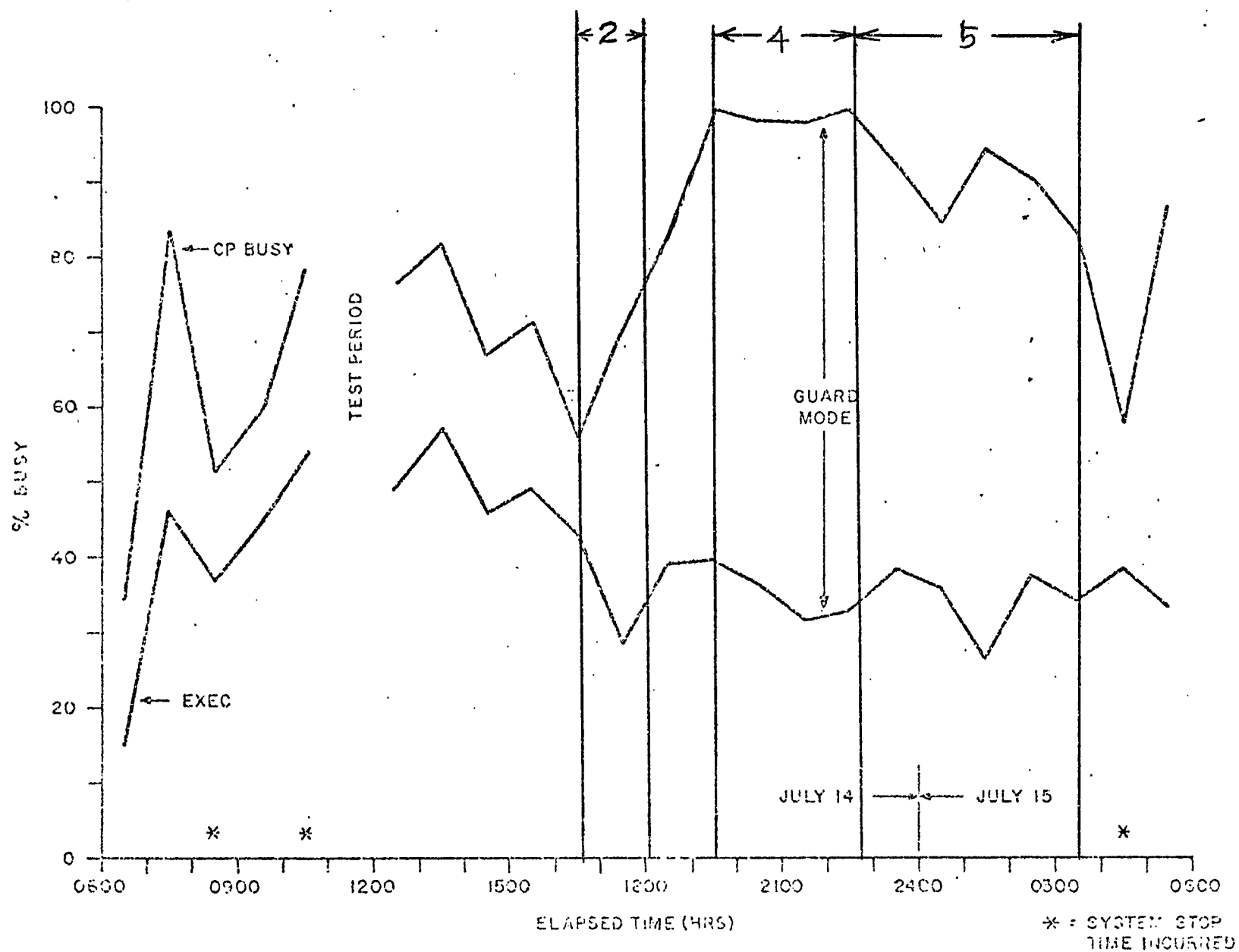
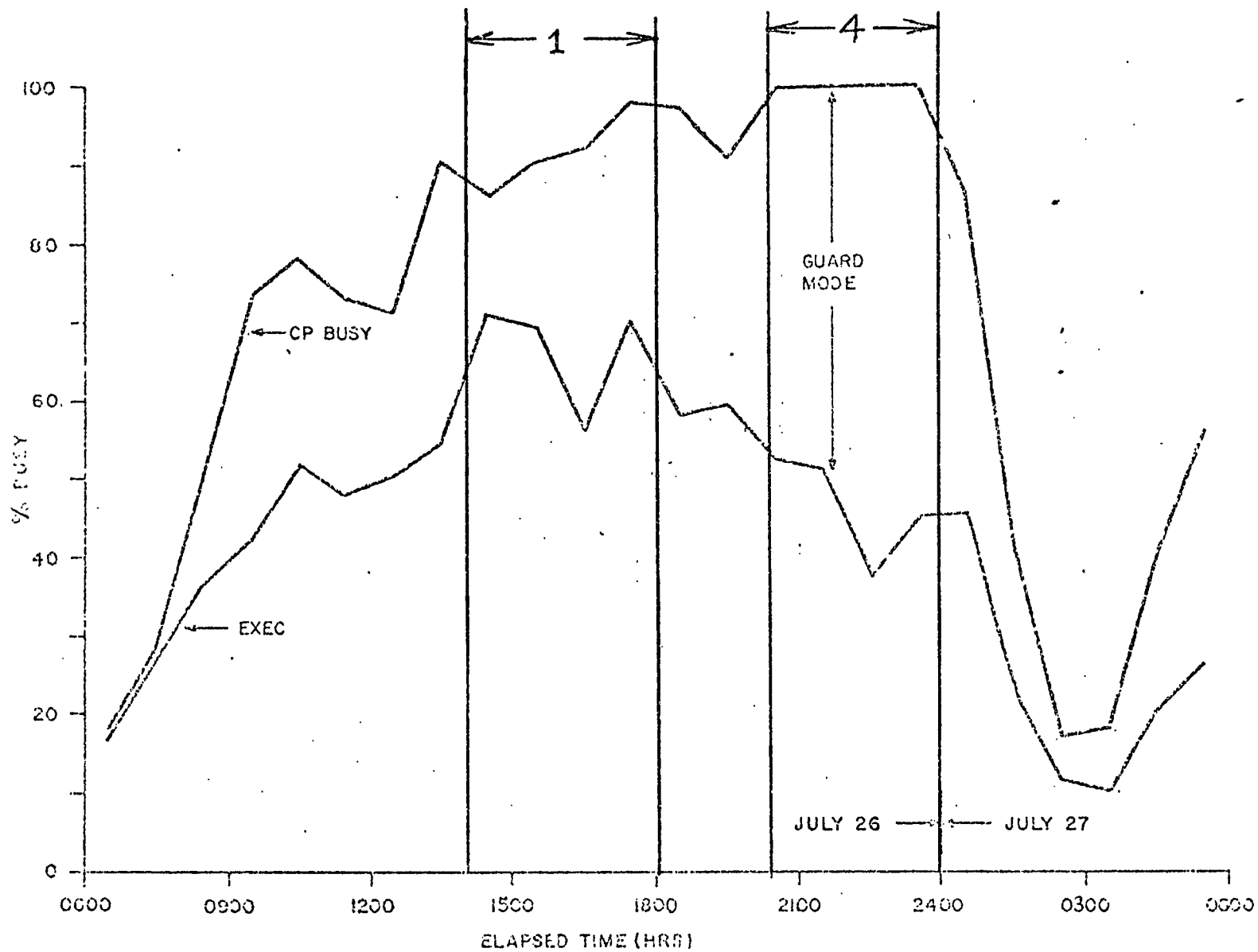


Figure 6.9 CP Profile of EXEC 8 on the "B" System



ations schedule. Since Figure [6.9] is based on data collected across a Wednesday/Thursday-boundary, the time series reflects the fact that IMPACT (a time-sharing program) is still in the system past 1630 hours. No checkpoint is being taken on Wednesday, therefore batch runs are allowed in and quickly use the available CP capacity. One observation that can be made from either of these figures is that the time spent processing applications programs, referred to as guard mode time, is always less during periods of real time activity. This phenomenon is caused by two factors: the overall CP utilization is actually lower during periods of real time processing; and the EXEC 8 activity is higher in the same timeframe.

From the monitoring data, we can determine the five typical input job types.

1. Time sharing with the background of a small number of batch jobs indicated by the time period 1400 to 1800 in Figure 6.9.
2. File manipulation such as FASTRAND (mass storage term in EXEC 8) I/O and the checkpoint of the computer operation indicated by the time periods 1900 to 2020, 2240 to 2440, and 1630 to 1800 in Figures 6.6, 6.7, and 6.8, respectively.
3. Scientific computation indicated by the time periods 1500 to 1820 and 1630 to 2130 in Figures 6.6 and 6.7, respectively.
4. Business computation with the normal I/O load indicated by the time periods 2020 to 2400 and 1930 to 2240 in Figures 6.9 and 6.8, respectively.
5. Business computation with the heavy I/O load indicated by the time period 2240 to 0330 in Figure 6.8.

The CP utilizations by activities, EXEC, user, and idle loop in each job type are tabulated in Table 6.2.

By analyzing the operating system EXEC 8, we have estimated state transition matrices for each input job type to realize the Central Processor

(CP) utilizations in the above five time ranges, which are defined from the monitoring data.

Job Type	Reading from the Monitoring Data			Computed Steady State from each Job Type Transition Matrix		
	Exec	User	Idle	Exec	User	Idle
1	70	23	7	87.2	11.4	1.4
2	33	33	34	36.7	29.0	34.3
3	24	74	2	25.1	73.4	1.5
4	40	57	3	38.8	56.9	4.2
5	38	48	14	46.3	40.6	12.8

Table 6.2 The CP Utilization in Percentage for Each Job Type

The state transition matrices for each job type are in Tables 6.3 through 6.7. Computed steady states from the state transition matrices of each job type are tabulated in the right frame of Table 6.2. Both quantities in Table 6.2 from the monitoring data and from the computed steady states for each job type are very similar to each other.

The computer system configurations of the "A" and "B" systems are very similar. Each input/output data channel is described as follows. Channel 0 leads high speed drums as used Swap\$ file for multiprogramming/time-sharing purposes. Channels 1 and 2 are a dual channel system to connect high speed/medium speed drums or disks as auxiliary memories holding the nonresident elements of the operating system and using temporary files of user's programs. Channel 3 is utilized as mass storage or as principle hardware of the computer file system which is stored in a slow speed drum. Channel 4 is a magnetic tape data channel. Channel 5 is used as a line printer. Channel 6 is connected to the communication terminal modular control unit as access to communication lines. Channel 7 is a complex subsystem used as a card reader, a card

Table 6.3. - State Transition Table of Time Sharing.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	.0	.07	.08	.07	.0	.08	.07	.08	.07	.07	.05	.05	.15	.05	.08	.03
2	.0	.20	.10	.03	.0	.09	.12	.09	.10	.08	.0	.0	.15	.0	.04	.0
3	.0	.14	.21	.13	.0	.06	.06	.06	.07	.08	.0	.0	.15	.02	.02	.0
4	.02	.05	.13	.17	.03	.08	.08	.08	.08	.01	.04	.01	.15	.04	.04	.0
5	.01	.06	.07	.07	.0	.09	.10	.10	.09	.03	.03	.03	.20	.05	.06	.01
6	.0	.08	.05	.03	.0	.10	.11	.10	.10	.04	.04	.04	.18	.06	.07	.0
7	.0	.05	.02	.02	.02	.20	.20	.20	.15	.01	.01	.01	.08	.01	.02	.0
8	.0	.10	.08	.02	.0	.10	.20	.20	.20	.0	.0	.0	.08	.0	.02	.0
9	.0	.10	.09	.05	.0	.10	.25	.10	.08	.03	.03	.0	.13	.02	.02	.0
10	.0	.15	.10	.0	.0	.02	.03	.0	.05	.10	.15	.15	.15	.0	.10	.0
11	.0	.06	.06	.06	.0	.08	.08	.05	.08	.20	.05	.03	.20	.03	.02	.0
12	.0	.01	.01	.01	.0	.08	.08	.08	.08	.0	.0	.0	.40	.10	.15	.0
13	.0	.06	.06	.06	.0	.08	.08	.08	.08	.0	.0	.0	.20	.15	.10	.05
14	.0	.06	.06	.06	.0	.08	.08	.08	.08	.0	.0	.0	.15	.15	.15	.05
15	.0	.06	.06	.06	.0	.08	.08	.08	.08	.0	.0	.0	.15	.10	.20	.05
16	.01	.06	.06	.06	.01	.07	.08	.08	.08	.03	.03	.05	.23	.05	.06	.03

Table 6.4. - State Transition Table of File Manipulation.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	.0	.03	.03	.03	.03	.02	.02	.02	.02	.0	.05	.02	.15	.10	.18	.30
2	.0	.02	.03	.03	.03	.02	.02	.02	.03	.04	.05	.02	.20	.12	.20	.16
3	.0	.02	.02	.02	.02	.02	.02	.02	.02	.15	.05	.02	.20	.12	.20	.10
4	.0	.02	.02	.02	.02	.02	.02	.02	.02	.20	.08	.04	.10	.12	.20	.10
5	.0	.06	.06	.06	.04	.03	.03	.03	.03	.03	.10	.05	.10	.10	.18	.10
6	.0	.05	.08	.05	.03	.05	.10	.10	.10	.02	.02	.02	.06	.05	.10	.17
7	.0	.05	.08	.05	.03	.05	.10	.15	.08	.02	.02	.02	.05	.05	.10	.15
8	.0	.05	.08	.05	.03	.05	.10	.05	.15	.02	.02	.02	.08	.05	.10	.15
9	.0	.08	.10	.15	.06	.05	.05	.05	.02	.02	.02	.02	.08	.05	.10	.15
10	.0	.02	.03	.03	.02	.0	.01	.01	.01	.08	.22	.25	.15	.10	.02	.05
11	.0	.02	.03	.03	.01	.01	.01	.01	.01	.10	.10	.10	.12	.10	.05	.30
12	.0	.02	.03	.02	.01	.01	.0	.0	.0	.04	.08	.10	.20	.11	.08	.30
13	.01	.02	.02	.02	.01	.0	.01	.01	.01	.03	.08	.15	.08	.10	.18	.27
14	.0	.0	.02	.02	.02	.0	.0	.0	.0	.0	.05	.02	.10	.12	.30	.35
15	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.07	.05	.08	.10	.20	.50
16	.0	.02	.03	.03	.03	.0	.0	.01	.03	.0	.0	.0	.15	.15	.15	.40

Table 6.5. - State Transition Table of Scientific Computation.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	.02	.08	.14	.02	.02	.0	.0	.0	.0	.06	.04	.05	.06	.50	.01	.0
2	.01	.04	.12	.04	.02	.0	.0	.0	.0	.06	.04	.06	.05	.50	.03	.03
3	.01	.02	.10	.08	.05	.0	.0	.0	.0	.06	.04	.05	.04	.50	.04	.01
4	.01	.02	.12	.05	.08	.0	.0	.0	.0	.06	.05	.06	.05	.43	.04	.03
5	.01	.02	.10	.05	.10	.0	.0	.0	.0	.08	.06	.05	.05	.41	.03	.04
6	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
7	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
8	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
9	.0	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
10	.0	.01	.02	.01	.01	.0	.0	.0	.0	.08	.20	.07	.05	.50	.03	.02
11	.0	.01	.02	.01	.01	.0	.0	.0	.0	.11	.07	.15	.17	.38	.05	.02
12	.0	.02	.08	.04	.02	.0	.0	.0	.0	.02	.02	.10	.15	.50	.04	.01
13	.0	.01	.02	.01	.01	.0	.0	.0	.0	.01	.03	.08	.11	.65	.05	.02
14	.0	.01	.02	.01	.01	.0	.0	.0	.0	.01	.01	.02	.10	.70	.10	.01
15	.0	.01	.02	.01	.01	.0	.0	.0	.0	.02	.02	.06	.10	.65	.07	.03
16	.0	.02	.04	.02	.02	.0	.0	.0	.0	.03	.03	.09	.20	.40	.05	.10

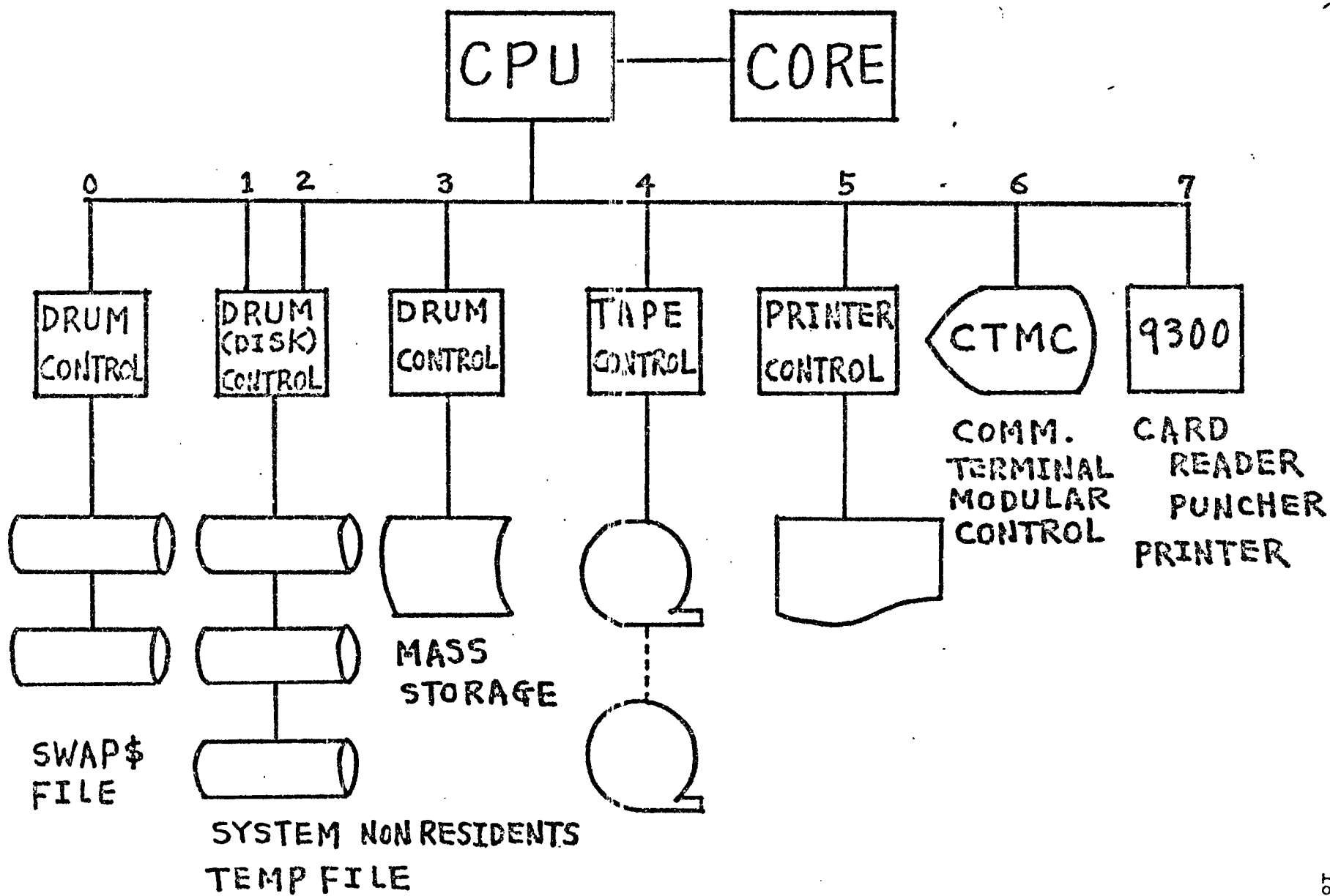
Table 6.6. - State Transition Table of Business Computation 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	.02	.08	.14	.02	.02	.0	.0	.0	.0	.05	.04	.05	.06	.41	.06	.05
2	.01	.04	.12	.06	.05	.0	.0	.0	.0	.06	.05	.07	.06	.38	.05	.05
3	.01	.03	.10	.08	.07	.0	.0	.0	.0	.07	.05	.06	.05	.40	.04	.04
4	.01	.03	.12	.06	.10	.0	.0	.0	.0	.06	.05	.06	.06	.35	.06	.04
5	.01	.03	.10	.05	.12	.0	.0	.0	.0	.08	.06	.05	.05	.37	.05	.03
6	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
7	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
8	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
9	.0	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
10	.0	.03	.08	.04	.05	.0	.0	.0	.0	.08	.10	.07	.05	.50	.03	.02
11	.01	.02	.08	.06	.05	.0	.0	.0	.0	.06	.04	.08	.09	.40	.07	.04
12	.0	.03	.10	.06	.04	.0	.0	.0	.0	.03	.03	.04	.08	.50	.06	.03
13	.01	.02	.08	.05	.04	.0	.0	.0	.0	.02	.03	.04	.04	.60	.05	.02
14	.0	.03	.07	.05	.04	.0	.0	.0	.0	.02	.03	.02	.05	.60	.05	.04
15	.0	.04	.06	.05	.05	.0	.0	.0	.0	.03	.05	.06	.08	.52	.03	.03
16	.0	.04	.06	.04	.04	.0	.0	.0	.0	.04	.05	.09	.09	.30	.10	.15

Table 6.7. - State Transition Table of Business Computation 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	.01	.08	.12	.06	.04	.0	.0	.0	.0	.07	.06	.07	.08	.28	.06	.07
2	.01	.04	.12	.07	.04	.0	.0	.0	.0	.07	.06	.08	.07	.30	.06	.08
3	.01	.03	.10	.07	.06	.0	.0	.0	.0	.08	.06	.07	.06	.29	.07	.10
4	.01	.03	.11	.06	.10	.0	.0	.0	.0	.07	.06	.07	.07	.25	.06	.11
5	.01	.04	.10	.06	.12	.0	.0	.0	.0	.08	.04	.06	.07	.25	.07	.10
6	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
7	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
8	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
9	.0	.0	.0	.0	.0	.0	.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
10	.0	.03	.08	.06	.05	.0	.0	.0	.0	.06	.10	.08	.05	.33	.06	.10
11	.01	.03	.10	.06	.05	.0	.0	.0	.0	.04	.05	.07	.07	.32	.07	.13
12	.0	.03	.11	.07	.05	.0	.0	.0	.0	.02	.02	.05	.08	.35	.08	.14
13	.01	.02	.10	.06	.05	.0	.0	.0	.0	.02	.03	.06	.05	.39	.07	.14
14	.0	.03	.07	.05	.04	.0	.0	.0	.0	.03	.04	.05	.06	.45	.06	.12
15	.0	.04	.08	.05	.05	.0	.0	.0	.0	.04	.05	.08	.09	.35	.05	.12
16	.0	.04	.06	.04	.04	.0	.0	.0	.0	.04	.05	.06	.07	.30	.10	.20

Figure 6.10 A Computer Configuration



Utilization Percentages of Data Channels

Channel Job Type	0	1	2	3	4	5	6	7
1	70	20	65	30	5	15	35	10
2	7	35	80	50	30	20	5	15
3	16	25	35	35	25	75	0	35
4	20	40	60	30	35	80	0	35
5	15	15	30	10	15	55	0	20

Table 6.8a Collected and Expected Channel Activities

Channel Job Type	0	1	2	3	4	5	6	7
1	75.6	27.2	86.6	37.8	13.1	25.4	41.4	13.2
2	3.0	22.4	73.9	54.3	21.4	32.5	6.1	18.5
3	18.1	32.9	10.4	32.0	44.0	81.7	3.6	49.1
4	16.9	17.2	5.0	30.7	33.8	72.6	3.3	38.7
5	10.5	12.1	24.6	37.0	26.0	58.8	3.0	28.5

Table 6.8b The Computed Channel Activities

Channel State	0	1	2	3	4	5	6	7
1	2	2	20	30	5	25	2	3
2	1	1	20	30	4	35	2	2
3	2	2	40	40	3	40	3	2
4	2	2	40	40	5	30	1	4
5	3	1	50	50	6	30	2	3
6	170	8	15	26	1	-12	95	2
7	179	10	180	26	2	15	90	1
8	172	9	15	15	1	-10	85	2
9	182	8	16	25	2	-14	90	2
10	5	4	10	30	10	90	5	10
11	8	5	12	30	8	70	3	9
12	7	3	18	40	20	80	2	6
13	6	5	250	30	12	50	3	50
14	38	4	-64	10	50	100	4	65
15	-100	380	280	200	110	35	3	10
16	20	-130	-10	20	-20	-10	2	5

Table 6.9 The Transfer Matrix State versus Channel Activities

punch, and a line printer. These channels are displayed in Figure 6.10.

Utilization percentages of output data channel activities of channels 0 through 4 are reported in the Univac documents. These utilization percentages are also used in the model. Utilization percentages of the other channels are estimated from characteristic job categories and the internal executive functions of the EXEC 8 operating system. These utilization percentages are tabulated in Table 6.8a.

The next step is to build the transfer matrix between steady state versus output data channel activity. We determine the transfer matrix of the model by trial and error as described in Table 6.9. Negative values in entries of the matrix indicate interference with the normal operation of the channel by the state activity, for example, priority schemes used in the operating system.

The computed utilization percentages from the steady states of each job type and the determined transfer matrix are tabulated in Table 6.8b. The discrepancy between the two tables in Table 6.8 looks large, but it is an acceptable range for the first model.

6.4 Computer Simulation of the Example

In the preceding section, the model of a computer system is built using a generalized probabilistic sequential machine (GPSM). Five input job types (input symbols), sixteen states, eight output items, five state transition matrices for each job type, and a transfer matrix from state distribution to the output items have been determined from the computer configuration and from the monitored computer performance data given in the UNIVAC report. As defined in the preceding section, the calibration phase and the estimation phase are involved in an application of the statistic technique. The cali-

bration phase is finished by model building. In order to present the estimation phase, daily operation of the computer system is simulated in a computer based on the model which has been built. Then the optimization process is applied on a simulated result of the daily computer operation. The estimated error location in a subsystem is compared with the computed simulation result of the subsystem activities.

The essentials of the simulation are mixing of input job types in accordance with pseudorandom numbers generated in a computer. A mixing ratio of job types (input symbol distribution) is predetermined; therefore, the next state distribution is computered using the state transition matrix chosen by a generated random number. When a state distribution is reached in a steady state, the simulation process is terminated. The steps of the simulation are described in the following paragraphs.

1. Read
 State transition matrix and
 the transfer matrix, state versus output.
2. Generate
 A random number vector.
3. Print
 The matrices and the distribution of the generated random
 numbers.
4. Read
 The initial state distribution and a mixing ratio of input
 job types.
5. Print
 The initial state distribution and the mixing ratio.
6. Loop Entry
 Get a job type index from the random number vector.

7. Compute
The next state distribution.
8. Determine
Is the present state distribution a steady state?
9. Branch
If it is not a steady state, then compute the present output activity. Go to the loop entry.
10. Terminate
If it is a steady state then terminate.

A definition of the steady state for computational convenient is introduced as follows;

Define ϵ_j such as

$$\epsilon_j = |S_j(t_m) - W_1 S_j(t_{m-1}) - W_2 S_j(t_{m-2}) - W_3 S_j(t_{m-3}) - \dots|$$

for $j = 1, 2, 3, \dots, 16,$

where $S_j(t_m)$ is the j th state probability at a time t_m and W_k is the weighting factor at the time t_{m-k} . The weighting is assumed geometric, namely $W_1 = r$, $W_2 = r^2, \dots, W_k = r^k, \dots$ so that the sum of W_k for $k = 1, 2, 3, \dots$ must be one

$$r + r^2 + r^3 + r^4 + \dots + r^k + \dots = \frac{r}{1-r} \rightarrow 1 \text{ as } k \rightarrow \infty,$$

thus $r = \frac{1}{2}$.

Only when the maximum ϵ_j for $j=1, 2, 3, \dots, 16$ becomes less than a given criterion ϵ , the present state distribution $S_j(t_m)$ for $j=1, 2, 3, \dots, 16$, is regarded as a steady state.

An example of a computer simulation is presented here. A job type mixing ratio is given in Table 6.10a. Computed steady state distribution and

output channel activities are tabulated in Tables 6.10b and 6.10c, respectively. Since the probabilities of the state distribution and the utilization of output channels are in a steady state, this situation of the computer system shall be continued in a certain time period until the system moves into a transient period to settle down in another steady state. Suppose an error occurred during the steady state. An estimation of the subsystem's activity is performed by optimizing the nonlinear mathematical programming problem described in Section 6.2.

The Lagrange multiplier method is used to solve the nonlinear mathematical programming problem. The optimum solution of the problem (the estimation of the job type mixing ratio) is presented in Table 6.10d. The estimated steady state and the difference between the estimated steady state and the computed steady state are tabulated in Table 6.10e. Activity orders of the estimated steady state and the computed steady state are presented in Table 6.10f.

Job Type No.	1	2	3	4	5
Probability	.025	.025	.025	.024	.91

Table 6.10a Mixing Ratio of Job Type Used in a Computer Simulation

State No.	1	2	3	4	5	6
Probability	.003185	.02859	.07437	.05415	.05923	.02008
State No.	7	8	9	10	11	12
Probability	.03210	.03294	.03454	.03871	.04123	.05359
State No.	13	14	15	16		
Probability	.05775	.29751	.06009	.11195		

Table 6.10b Computed Steady State

Channel No.	1	2	3	4	5	6
Percentage Utilization	30.36	11.69	28.81	35.28	22.89	51.16

Channel No.	7	8
Percentage Utilization	13.36	25.28

Table 6.10c Output Channel Activity in the Computed Steady State

Job Type No.	1	2	3	4	5
Probability	.254	.110	.0	.636	.0

Table 6.10d The Optimal Job Type Mixing Ratio (The Estimated Ratio)

State No.	1	2	3	4	5	6
Estimated	.002463	.04395	.07359	.04990	.03584	.02576
Computed	.003185	.02859	.07437	.05415	.05923	.02008
Difference	-.000722	.01536	-.00078	-.00415	-.02339	.00568

State No.	7	8	9	10	11	12
Estimated	.03381	.02890	.02840	.03283	.03464	.03337
Computed	.03210	.03294	.03454	.03871	.04123	.05359
Difference	.00171	-.00404	-.00614	-.00588	-.00658	-.02021

State No.	13	14	15	16
Estimated	.08624	.35539	.06718	.06774
Computed	.05775	.29751	.06009	.11195
Difference	.02839	.05788	.00709	.04421

Table 6.10e Most-Likely Steady State (Estimated Probability) and the Difference of the Probability from the Computed Steady State Probability (6.10b).

Activity Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Estimated State	14	13	3	16	15	4	2	5	11	7	12	10	8	9	6	1
Computed State	14	16	3	15	5	13	4	12	11	10	9	8	7	2	6	1

Table 6.10f The Activity Orders of the Computed Steady State and of the Estimated Steady State.

Although the estimated job type mixing ratio by the optimization deviates wildly from the imposed mixing ratio in the simulation (as shown in Tables 6.10a and 6.10d), the estimation of subsystem activities indicated in Table 6.10e is very good. The estimated result is nearly consistent with the

computed activity order (see Table 6.10f) except states 2,13 and 12,16. The estimated activities of states 2,13 and 12,16 become higher and lower than the computed activities, respectively. The higher estimated probabilities of job types 1 and 2 compared with the forced job mixing probabilities in the simulation may cause higher activity in states 2,13 and lower activity in states 12,16. A deviation from the uniform distributed random number which is used in the switching of job type mixing could result in the higher estimated probabilities. Nevertheless, the estimated state activities are close enough to the computed state activities in the example. The expected values, $\sum_{i=1}^{16} i P_r(s_i)$ where $P_r(s_i)$ is the probability of the i th steady state shown in Table 6.10e, are about 4.80 and 5.12 in the estimated steady state and the computed steady state, respectively.

Another application results of the statistical estimation technique are given in Table 6.11. Probabilities of job type mixing ratios in the simulation and the optimization are shown in Table 6.11a and activity orders of the computed (simulated) steady state and of the estimated steady state are presented in Table 6.11b. The expected values in the simulated steady state and the estimated steady state are about 3.41 and 3.12, respectively.

The optimization provides a good solution consisting with the simulated result where the probability of a state is clearly distinguishable (larger or smaller) but the optimized solutions of states, where probability differences between the states in a set are not large, do not match exactly to the simulated results. Nevertheless, the expected values in the simulation and the estimation are close enough to each other, so that this technique is very useful to find a subsystem which contains an error over a long maintenance time period.

The detailed results of these computing examples and the computer

program listings are attached in Appendix A.

Job Type No.	1	2	3	4	5
Estimated	.178	.0	.726	.096	.0
Computed	.10	.10	.60	.10	.10

Table 6.11a Probabilities of Job Type Mixing Ratio

Activity Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Estimated State	14	13	15	3	12	2	4	7	10	11	8	9	6	16	5	1
Computed State	14	13	15	12	8	9	7	3	6	11	10	16	4	5	2	1

Table 6.11b The Activity Orders of the Computed Steady State and of the Estimated Steady State.

This statistical technique can be applied iteratively on a subsystem, which is handled as a whole system in order to examine its relative subsystems. After a determination of the subsystem activity order in a system, any deterministic error (fault) detection methods [69], [65] shall be applied on each subsystem to locate an error.

CHAPTER VII

CONCLUDING REMARKS AND SUGGESTED FUTURE RESEARCH

The probabilistic sequential machine has been broadened into a more general form. This generalization includes uncertainty of incoming input symbols and a vector form of the outputs. Based on this generalized probabilistic sequential machine (GPSM) theory, a new approach to the computer system modeling and error detection problem (regardless of the computer structure level under consideration) has been presented. The representation of computer systems by generalized probabilistic sequential machines and the subsequent analysis are shown to be a useful tool in error detection as well as in system performance measurement.

The two-state absolutely isolated probabilistic sequential machine (PSM), which is obtained from newly defined state transferable ranges of the PSM, has been constructed. These ranges are defined by considering the signs of eigenvalues of each state transition matrix and are generally narrower than those of the well known completely isolated machine; therefore, the absolutely isolated machine includes the completely isolated machine as a subclass. It is pointed out that the absolutely isolated properties (such as the past input traceability and the past state or subsystem activities, etc.) are applicable to error detection. In addition, both the k th term approximation of the infinite series that represents the isolated properties and error bounds of the approximation have been discussed. Based on this study, an input string cutpoint to a probabilistic automaton (PA) is determined using the error bounds. The set of input strings recognized by the PA is much larger than the set recognized by the well known completely isolated PA.

In general, the initial state distribution of a PSM plays an important role in the probabilistic sequential machine theory. However, it has been proven that the influence of the initial state distribution diminishes as the length of an input string approaches infinity. This initial state independence property, one of the absolutely isolated properties, is discussed in detail in Chapter III. This property may be used in estimating the upper bound influenced by the deviation of the initial state distribution from $(1,0)$ or the first state $s_1 = 1$ and the second state $s_2 = 0$. Furthermore, when state transferable ranges of an absolutely isolated machine are spaced with more marginal distance than the upper bound of the initial state distribution influence, it is shown that the absolutely isolated machine is initial state independent.

A method for decomposing a three-state PSM into three two-state PSM's has been presented. The method can be extended to the decomposition of a general n -state PSM with $n > 3$, that is, an n -state PSM may be composed of two-state isolated machines, so the PSM has the absolutely isolated properties. The past input traceability of an n -state PSM can be applied to system performance analysis as well as error detection, if it is isolated and decomposable. A method for computing a state transition along with a particular path from the i th state to the j th state in a three-state (or an n -state) PSM has been discussed with decomposed two-state transition matrices which consist of component PSM's. The method provides a state transition allotment among substates of a state in the decomposed machine. The substate transition allotment computed with this method is a loading distribution of substate activities which may be useful for component reliability analysis in the realized PSM.

As discussed, a GPSM is suitable for modeling large computer systems. In order to determine the past subsystem activities of a GPSM model which may or may not have the input-traceable property, a statistical technique which is a nonlinear mathematical programming problem has been formulated. The optimization of the programming problem provides the most-likely past subsystem activities based on collected data in two phases. These data can be gathered in a relatively simple way so that the technique is easily applicable to a real situation. A GPSM model of a UNIVAC EXEC 8 computer system has been built, and daily computer operation of the model is simulated in a computer. Incoming inputs to the model are determined from pseudo random numbers. It is demonstrated that the most-likely state distribution obtained by this technique closely agrees with the steady state distribution computed from state transition matrix multiplication in examples of the simulation.

The theory and techniques presented here do not pretend to solve all problems with which a computer system is faced today, but its value rests on the new insight it provides to the hardware designer and the maintenance engineer as well as the software engineer. The result is a preliminary report on the subject of computer system modeling and error detection. Much fruitful work remains in many aspects. Some possible research areas are described in the following paragraphs.

Future research topics evolving from Chapter II include improvements in model building techniques. A technique to determine better constants (parameters), methods to insert measuring probes in the computer system, and a systematic separation of subsystems are possible subjects.

Chapter III suggests a relationship study of the isolation of (1,2) elements with (a) eigenvalues, (b) the (2,1) elements of the matrices, or (c) a number of input symbols. The (1,2) elements are elements of state transition matrices of a two-state multisymbol PSM.

From Chapter IV, a decomposition using weighted state transitions among substates of a state, instead of assuming equal weighting of the substates, may expand the decomposable class of PSM's. Serial decomposition with $n(n-1)/2$ component machines, where n is a number of states, may bring to research a new area of PSM decomposition. A review of the dot-circle multiplication from the matrix theory would be another interesting topic.

Studies of a relationship between decomposability and isolation in an n -state PSM and of the input traceability as opposed to other characteristics such as eigenvalues and steady state of the PSM may be interesting topics in Chapter V. Applications of the input traceability to other fields are also worthwhile.

Combinations of the technique presented in this paper with other statistical estimation techniques in order to improve accuracy and optimization development for ergodic states are immediate research topics. Direct application of stochastic programming method [77] to the estimation of past performance of a computer system is an exciting subject in place of nonlinear mathematical programming. Applications of the GPSM modeling and the statistic estimation technique to other fields such as economics and social science are expandable.

BIBLIOGRAPHY

1. Allied Computer Technology, Inc., Computer Performance Monitor Operator Guide, Allied Computer Technology, Inc., June 1970.
2. S. B. Akers, On a Theory of Boolean Functions, J. Soc. Ind. App. Math., 7, Dec., 1959, pp.487-498.
3. H. Asai, and S. C. Lee, Design of Queuing Buffer Register Size, Information Processing Letter, Vol. 3/5, 1975, pp.147-152.
4. G. C. Bacon, The Decomposition of Stochastic Automata, Inf. and Control, 7, 1964, pp.320-339.
5. R. M. Balzer, EXDAMS-Extendable Debugging and Monitoring System, Proc. of Spring Joint Computer Conference, Vol. 34, 1969, pp.567-580.
6. F. L. Baner, Advanced Course on Software Engineering, Spring-Verlag, 1973.
7. C. G. Bell, and A. Newell, Computer Structures: Readings and Examples McGraw-Hill Co., 1971.
8. T. L. Booth, Sequential Machine and Automata Theory, John Wiley and Son, Inc., 1967.
9. E. G. Ceffman, et al., System Deadlock, ACM Computing Surveys, Vol. 3, No. 2, June 1972, pp.67-78.
10. H. Y. Chang, Figure of Merit for the Diagnostic of a Digital Systems, IEEE Trans. Reliability, R-17, No. 3, 1967, pp.147-153.
11. J. D. Couger and R. W. Knapp, ed., System Analysis Techniques, John Wiley and Sons, Inc., 1974.
12. J. B. Dennis, Concurrency in Software System, Chapter 2.E., Advanced Course on Software Engineering, Ed. F. L. Bauer, Spring-Verlog, 1973.
13. E. W. Dijkstra, The Structure of the "T.H.E." Multiprogramming System, CACM Vol. 5, No. 3, March 1968, pp.341-346.
14. E. W. Dijkstra, A Constructive Approach to the Problem of Program Correctness, BIT, Vol. 8, No. 3, 1968, pp.174-186.
15. J. J. Donovan, System Programming, McGraw-Hill Co., 1972.
16. M. E. Drummond, Evaluation and Measurement Techniques for Digital Computer Systems, Prentice-Hall, Inc., 1973.
17. R. D. Eldred, Test Routines Based on Symbolic Logic Statements, J. of ACM, Vol. 6, No. 1, Jan., 1959, pp.33-36.

18. B. Elspas, et al., Software Reliability, IEEE Compt., Vol. 20, No. 1, Jan., 1971.
19. W. Feller, An Introduction to Probability Theory and its Applications Vol. I and II, John-Wiley and Sons, Inc., 1968, 1971.
20. R. L. Gauthier, and S. D. Ponto, Designing Systems Programs, Prentice-Hall, Inc., 1970.
21. D. P. Gaver and G. L. Thompson, Programming and Probability Models in Operations Research, Brooks/Cole Pub. Co., 1973.
22. S. Ginsburg, An Introduction to Mathematical Machine Theory, Addison-Wesley Pub. Co., 1962.
23. S. Ginsburg, The Mathematical Theory of Context Free Languages, McGraw-Hill Co., 1966.
24. R. M. Graham, Protection in an Information Processing Utility, CACM, Vol. 11, No. 5, 1968, pp. 365-369.
25. Hardic and Suhocki, Design and Use of Fault Simulation for Saturn Computer Design, IEEE Trans., EC-16, 4 1967, pp. 412-429.
26. L. Harold, Parallelism in Hardware and Software, Prentice-Hall, Inc., 1972.
27. M. A. Harrison, Lectures on Linear Sequential Machine, Academic Press., 1969.
28. J. Hartmanis, and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Inc., 1966.
29. N. Hawkes, Ed., Lecture Notes in Economics and Mathematical Systems, International Seminar on Trends in Mathematical Modeling, Venice, 13-18, Dec., 1971, Springer-Verlag, 1973.
30. W. Henahple, A Proof of Correctness of the Reference Mechanism to Automatic Variables in the F-Compiler, LN 25.3.048, IBM Vienna Laboratory.
31. F. S. Hillier, and G. J. Lieberman, Introduction to Operation Research, Holden-Day, Inc., 1968.
32. A. W. Holt, et al., Marked Directed Graphs, J. of Compt. and System Science, Vol. 5, 1971, pp. 511-523.
33. J. E. Hopcroft, and J. D. Ullman, Formal Languages and Their Relation to Automata, Addison-Wesley Pub. Co., 1969.
34. S. T. Hu, Mathematical Theory of Switching Circuits and Automata, Berkeley, University of California Press, 1968.

35. H. Katzan, Operating System, A Pragmatic Approach, Van Nostrand Reinhold Co., 1973.
36. K. Krohn, and J. Rhodes, Algebraic Theory of Machines. I. Prime Decomposition Theorem for Finite Semigroups and Machines, Trans. Am. Math. Soc., 116, 1965, pp.450-464.
37. K. Krohn, R. Matheosian, and J. Rhodes, Method of the Algebraic Theory of Machines. I. Decomposition Theorem for Generalized Machines, Properties Preserved under Series and Parallel Compositions of Machines, J. of Computer and System Science, 1, 1967, pp.55-85.
38. K. R. London, Decision Tables, Auerbach Publish, 1972.
39. H. C. Lucas, Performance Evaluation and Monitoring, ACM Computer Surveys, Vol. 3, 1971, pp.79-91.
40. D. G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley Pub. Co., 1973.
41. Z. Manna, and R. J. Waldinger, Toward Automatic Program Synthesis, CACM Vol. 14, No. 3, March 1971, pp.151-165.
42. Manning and Chang, A Comparision of Fault Simulation for Digital System, Digest of the First Annual IEEE Computer Conf., 1967, pp.10-13.
43. P. R. Menon, On Sequential Machine Decompositions for Reducing the Number of Delay Elements, Inf. and Control, 15, 1969, pp.274-287.
44. M. D. Mesarovic, D. Macko, and Y. Takahara, Theory of Hierarchical, Multilevel, Systems, Academic Press, 1970.
45. M. L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Inc., 1967.
46. A. Mukhopadhyay, Ed., Recent Developments in Switching Theory, Academic Press, 1971.
47. M. C. Newey, P. C. Poole, and W. M. Waite, Abstract Machine Modeling to Produce Portable Software - a Review and Evaluation, Software, 2, 1972, pp.107-136.
48. A. R. Northhouse, and K. S. Fu, Dynamic Scheduling of Large Digital Computer System Using Adaptive Control and Clustering Techniques, IEEE Trans. System, Man and Cyb. 3, No. 3, May 1973, pp.225-234.
49. C. V. Page, Equivalences Between Probabilistic Sequential Machines, Ph.D. thesis, U. of Michigan, 1969.
50. S. S. Patil, Coordination of A Synchronous Events, Project Mac Technical Report TR-72, MIT (Cambridge, Mass.) June, 1970.

51. A. Paz, Introduction to Probabilistic Automata, Academic Press, 1971.
52. M. O. Rabin, Probabilistic Automata, *Inf. and Control*, 6, 1963, pp.230-245.
53. C. V. Ramamoorthy, Analysis of Graphs by Connectivity Considerations, *J. of ACM*, Vol. 13, No. 2, 1966, pp.211-222.
54. C. V. Ramamoorthy, A structural Theory of Machine Diagnosis, *Proc. of Spring Computer Conf.* 30, 1967, pp.743-756.
55. J. Reitman, Computer Simulation Applications Discrete - Event Simulation for Synthesis and Analysis of Complex Systems, Wiley-Interscience, John Wiley and Sons, Inc., 1971.
56. P. Rivett, Principles of Model Building, John Wiley and Sons, 1972.
57. S. Rosen, Ed., Programming Systems and Languages, McGraw-Hill Co., 1967.
58. J. P. Roth et al., Techniques for the Diagnosis of Switching Circuit Failures, *Trans. IEEE Comm. and Electronics*, Sept., 1964, pp.509-514.
59. J. P. Roth, Diagnosis of Automata Failures: A Calculus and a Method, *IBM J. Res. Develop.*, Vol. 10, 1966, pp.278-291.
60. R. Rustin, Ed., Debugging Techniques in Large Systems, Prentice-Hall, Inc., 1971.
61. E. S. Santos, Identification of Stochastic Finite Systems, *Inf. and Control*, 21, 1972, pp.1-20.
62. A. P. Sayers, Ed., Operating Systems Survey, Auerbach Pub., 1971.
63. M. Schroeder, and J. Saltzer, A Hardware Architecture for Implementing Protection Rings, *Proc., the third Symp. on O.S. Principle*, Oct. 18-20, pp.42-54.
64. E. F. Sellers, et al., Analyzing Errors with the Boolean Difference, *IEEE Trans., Compt.*, C-17, July, 1968, pp. 676-683.
65. Seshu and Freman, The Diagnosis of Asynchronous Sequential Switching System, *IRE Trans.*, EC-11, 4, 1962, pp.459-465.
66. M. K. Starr, Systems Management of Operations, Prentice-Hall, Inc., 1971.
67. S.Y.H. Su, and Y.C. Cho, A New Approach to the Fault Location of Combinational Circuits, *IEEE Trans., Compt.*, C-21, Jan., 1972, pp.21-36.
68. C. P. Tan, On Two-State Isolated Probabilistic Automata, *Inf. and Control*, 21, 1972, pp.483-495.

69. A. Thayse, and M. Davio, Boolean Differential Calculus and Its Application to Switching Theory, IEEE Trans., Compt., C-22, No. 4 (April) 1973, pp. 409-420.
70. United Software Co., Technical Description of the 1100 Software Instrumentation Package (SIP), United Software Co., Minneapolis, Minn., #85084, July, 1973.
71. Univac, Large Scale Systems MATH-PACH, UP-4051 Rev. 2, 1967, 1970, Univac Div., Sperry Rand Co.
72. Univac, Univac 1100 Series Operating System Programmer Reference, Univac, Sperry Rand Co., UP-4144, Rev. 3, 1973.
73. Univac Americas Division, Systems and Application Group, Systems Performance Analysis at Burndy Corp., Univac, Salt Lake City, Americas Division Report, Feb. 1973.
74. Univac Americas Division, Systems and Application Group, System Performance Analysis at Electric Boat Division of General Dynamics, Univac, Salt Lake City, Americas Division Report, Oct., 1972.
75. Univac Americas Division, Systems and Application Group, System Performance Analysis at Electric Boat Division of General Dynamics Part 2, Univac, Salt Lake City, Americas Division Report, Jan., 1973.
76. Univac, Education Center, 1100 EXEC 8 Internal Structure Level 32, Release 1, 1100 Systems Education. Depart., Roseville, Minn., Sep., 1974.
77. S. Vajda, Probabilistic Programming, Academic Press, New York, 1972.
78. T. Yasui, and S. Yajima, Some Algebraic Properties of Sets of Stochastic Matrices, Inf. and Control, 14, 1960, pp. 319-357.
79. T. Yasui, and S. Yajima, Two-State Two-Symbol Probabilistic Automata, Inf. and Control, 16, 1970, pp. 203-224.
80. S. S. Yau, and K. C. Wang, Linearity of Sequential Machines, IEEE Trans., EC-15, No. 3, 1966, pp. 337-354.
81. S. S. Yau, and Y. S. Tang, An Efficient Algorithm for Generating Complete Test Sets for Combinational Logic Circuits, IEEE Trans., Compt., C-20, Nov., 1971, pp. 1245-1251.
82. L. A. Zadeh, and C. A. Desoer, Linear System Theory, McGraw-Hill Book Co., N. Y., 1963.

APPENDIX A

1. Simulation Program of Computer System Operation Modeled by Generalized Probabilistic Sequential Machine.
2. Simulation Examples; Example 1 and Example 2.
3. Optimization Program of the Statistical Estimation Technique.
4. Optimization Examples of the Simulation Examples.
5. Block Diagrams of Computations.

AM

ED: CODE(1) 000754; DATA(0) 011322; BLANK COMMON(2) 000000

REFERENCES (BLOCK, NAME)

ANGER
NIP
TMO
UTPRI
LO
NIR
DUS
OZS
RDS
OIS
TOP

SIMULATION PROGRAM

ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

00101 1236	0001	000102 1316	0001	000107 1356	0001	000115 1401
00102 1626	0001	000173 1756	0001	000205 2036	0001	000212 2036
00243 2266	0001	000254 2366	0001	000312 2556	0001	000324 2601
00372 2601	0001	000355 3026	0001	000363 3076	0001	000456 3101
00521 3026	0001	000431 3336	0001	000546 3501	0001	000576 3601
00534 3736	0001	000565 3801	0001	000551 4056	0001	000640 4301
00564 4456	0001	000676 4536	0001	000703 4601	0001	000711 5001
01207 501F	0000	011211 502F	0000	011216 503F	0000	011222 5001
01241 505F	0000	011247 507F	0000	011257 508F	0000	011271 5001
00716 5201	0000 R	010674 AST	0000 R	010777 ATACM	0000 P	011047 AT
01173 5201	0000 R	011160 DELTA	0000 P	010720 EPS	0000 I	011153 I
01103 1ACM	0000 I	011167 IACMAX	0000 I	011170 IACMIN	0000 I	011166 IAC
01143 1N	0000 I	011144 IC	0000 I	011174 IPRINT	0000 I	011164 IC
01154 0	0000 I	011163 IPRINT	0000 I	011155 K	0000 I	011172 KK
01147 L	0000 I	011146 LL	0000	011275 LSP	0000 I	011150 M
01151 N	0000 I	011145 NDIM	0000 R	010650 OUT	0000 R	010314 OUT
00744 RANIN	0000 R	011176 SCALE	0000	011275 SP	0000 R	010624 ST
01156 SUM	0000 R	011023 TACM	0000 R	011073 TACMP	0000 R	011117 TEF

```

* DIMENSION RAND(1500)
* DIMENSION TRANS(20,20,7)
* DIMENSION OUTMS(20,10)
* DIMENSION ST(20),OUT(20),AST(20),EPS(20),RANIN(7),STNY(20)
* DIMENSION ATACM(20),TACM(20),ATACMP(20),TACMP(20),TEPS(20)
* EQUIVALENCE (LSP,SP)
* C RAND RANLGN NO.
* C TRANS TRANSIENT TABLE
* C OUTMS OUTPUT MEASURE
* C ST, STATE, OUT, OUTPUT-MEASURE, AST, ACUM STATE, EPS, EPSHIRON
* IN=5
* IO=6
* NDIM=20
* WRITE(IN,510)
500 FORMAT(10110)

```

```

C      NO REPEAT, NO INPUT SYMB, NO STATE, NO OUTPUT MEASURE
C      TRANSFER MATRICES
C      OUTPUT MEASURE MATRIX
C      INITIAL STATE
C      RANDOM GENERATION TABLE
      READ(10,500) LL,L,M,N
      M1=M+1
      DO 10 I=1,L
      DO 10 J=1,M
      READ(10,501) (TRANS(J,K,I),K=1,M)
C      M1 TH COLUMN IS THE SUM OF ROW OF EACH
      SUM=0.
      DO 12 K=1,M
      SUM=SUM+TRANS(J,K,I)
12    CONTINUE
      TRANS(J,M1,I)=SUM
10    CONTINUE
501  FORMAT(8F10.7)
      DO 15 I=1,M
      READ(10,501) (OUTMS(I,J),J=1,N)
15    CONTINUE
      IRAN=1500
      CALL          RANGEN(RAND,IRAN)
      DELTA=0.0002
502  FORMAT(/10X,10F TRANSIENT TABLE ,110 )
      DO 20 I=1,L
      WRITE(10,502) I
      DO 20 J=1,M
      WRITE(10,503) (TRANS(J,K,I),K=1,M1)
503  FORMAT((10X,5(1X,L12.7)))
20    CONTINUE
      WRITE(10,504)
      DO 25 I=1,M
      WRITE(10,503) (OUTMS(I,J),J=1,N)
25    CONTINUE
504  FORMAT(/10X,20H OUTPUT-MEASURE TABLE )
      DO 100 II=1,LL
      WRITE(10,510)
510  FORMAT(1H1)
      KPRINT=20
      JPRINT=20
      IK=1
      IACM=1
      IACMP=1
      IACHAX=400
      IACHIN=100
      IACHLK=410
      READ(10,501) (SI(I),I=1,M)
      READ(10,501) (RANIN(I),I=1,L)
      WRITE(10,505)
505  FORMAT(/10X,30H INITIAL STATE AND INPUT PATTERN )
      WRITE(10,503) (SI(I),I=1,M)
      WRITE(10,503) (RANIN(I),I=1,L)
      DO 26 I=1,M
      AST(I)=0.
      ATAC(I)=0.
      ATACMP(I)=0.
      IACM(I)=SI(I)
      IACMP(I)=SI(I)
26    CONTINUE
C      MAIN REPEAT ENTRY
28    CONTINUE

```

```

CALL RANTR(TRA,K,RANIN,L)
CALL MATMUL(1,M,N,ST,TRANS(1,1,KK),STNX,NDIM)
DO 30 I=1,M
  AST(I)=AST(I)+0.5
30 CONTINUE
  BMAX=0.
  DO 32 I=1,M
    EPS(I)=ABS(STNX(I)-AST(I))
    IF(STNX(I).LE.1.0E-35) GO TO 31
    IF(STNX(I).LE.1.0E-27) EPS(I)=EPS(I)/STNX(I)
31 CONTINUE
    AST(I)=AST(I)+STNX(I)
    ST(I)=STNX(I)
    TACMP(I)=TACMP(I)+STNX(I)-ATACMP(I)
    TACM(I)=TACM(I)+STNX(I)-ATACM(I)
    BMAX=AMAX1(BMAX,EPS(I))
32 CONTINUE
    IF(1R.GT.IFAN) GO TO 50
    IPRINT=IR
    IF(BMAX.LT.DELTA) GO TO 46
    IF(1R.LT.IACMIN) GO TO 33
    DO 29 I=1,M
      ATACMP(I)=TACMP(I)/IACMIN
29 CONTINUE
33 CONTINUE
    IF(1R.LT.IACMAX) GO TO 35
    DO 34 I=1,M
      ATACM(I)=TACM(I)/IACMAX
34 CONTINUE
35 CONTINUE
    IF(1R.LT.IACHK) GO TO 36
    BMAX=0.
    DO 37 I=1,M
      TEPS(I)=ABS(ATACM(I)-ATACMP(I))
      BMAX=AMAX1(BMAX,TEPS(I))
37 CONTINUE
38 CONTINUE
    IF(BMAX.LT.DELT) GO TO 46
    IF(IPRINT.GE.JPRINT) GO TO 56
    GO TO 28
36 CONTINUE
    JPRINT=JPRINT+KPRINT
    CALL MATMUL(1,M,N,ST,OUTMS,OUT,NDIM)
    CALL OUTPRT(ST,OUT,AST,EPS,M,IR)
    WRITE(10,508)
    WRITE(10,509) (TACMP(I),I=1,M)
    WRITE(10,509) (ATACMP(I),I=1,M)
    WRITE(10,509) (TACM(I),I=1,M)
    WRITE(10,509) (ATACM(I),I=1,M)
    GO TO 28
46 CONTINUE
    WRITE(10,506)
306 FORMAT(1X,23(' *** STEADY STATE *** '))
    GO TO 52
50 CONTINUE
    WRITE(10,507)
307 FORMAT(1X,23(' *** NONE STEADY STATE STOP *** '))
308 FORMAT(1H0,1X,48('INTEGRATED SUM IN SHORT AND LONG TIME PER SEC'))
309 FORMAT(1H0,(18X,5(1X,12.7)))
52 CONTINUE
    CALL MATMUL(1,M,N,ST,OUTMS,OUT,NDIM)
    CALL OUTPRT(ST,OUT,AST,EPS,M,1)

```

```
* DIMENSION SCALE(1)  
* CALL PLOT(NDIM,1,ST,SCALE,M,1)  
100 CONTINUE  
* STOP  
* END
```

F COMPILATION: NO DIAGNOSTICS.

SE MATMUL ENTRY POINT 000135

USED: CODE(1) 000150; DATA(0) 000042; BLANK COMMON(2) 000000

REFERENCES (BLOCK, NAME)

ERR3

ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

00062 1056	0001 000005 1106	0001 000067 1146	0000 I 000000 I
00001 J	0000 I 000003 K	0000 R 000002 SUM	

```

*      SUBROUTINE MATMUL(L,M,N,A,B,C,MD)
*      DIMENSION A(L,MD),B(MD,N),C(L,N)
*      DO 10 I=1,L
*      DO 10 J=1,N
*      SUM=0.
*      DO 5 K=1,M
*      SUM=SUM+A(I,K)*B(K,J).
*      5  CONTINUE
*      C(I,J)=SUM
*      10 CONTINUE
*      RETURN
*      END
  
```

COMPILATION: RC DIAGNOSTICS.

OUTPUT ENTRY POINT 000105

SED: CODE(1) 000123; DATA(0) 000040; BLANK COMMON(2) 000000

REFERENCES (BLOCK NAME)

ED01
021
015
LRK31

ASSIGNMENT (BLOCK TYPE, RELATIVE LOCATION, NAME)

00027 1156	0001	000042 1246	0001	000055 1326	0001	000070 1406
00008 501F	0000	000017 502F	0000	I 000001 I	0000	000024 1406

```

SUBROUTINE OUTPUT(S,O,R,E,NO,MO)
DIMENSION S(NO),O(NO),R(NO),E(NO)
IO=6
WRITE(IO,500) NO
500 FORMAT(17B1,1B1) ITERATION .110)
501 FORMAT(25X, 40HSTATE,OUTPUT-MEASURE,ACUM.STATE AND EPS )
WRITE(IO,501)
WRITE(IO,502) (S(I),I=1,NO)
WRITE(IO,502) (O(I),I=1,NO)
WRITE(IO,502) (R(I),I=1,NO)
WRITE(IO,502) (E(I),I=1,NO)
502 FORMAT((18X,5(1X,E12.7)))
RETURN
END

```

COMPILATION: NO DIAGNOSTICS.

E RANGER ENTRY POINT 000100
RAND ENTRY POINT 000141

SED: CODE(1) 000157; DATA(0) 000060; BLANK COMMON(2) 000000

REFERENCES (BLOCK, NAME)

RANDU
RANDU
1015
1025
ERAB3

ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

00010 1116	0001	000104 12L	0001	000043 1216	0001	000067 17
00020 1	0000	000044 INOPS	0000	I 000025 IU	0000	I 000000 JP
00030 RAN						

```

SUBROUTINE RANGER(RAND,IRAN)
  DIMENSION RAND(IRAN)
  RAND(1)=1577.
  CALL RANDU(RAND,IRAN)
  DIMENSION JRV(21)
  IC=0
  DO 5 I=1,IRAN
    K=RAND(I)*20.+1.
    JRV(K)=JRV(K)+1
5  CONTINUE
  WRITE(10,500) (JRV(I),I=1,21)
500 FORMAT(/18X,15H RAND. V. DIST. ,/18X,21I3)
  RETURN
  ENTRY RANLE(IR,KK,RANIN,L)
  DIMENSION RANIN(L)
  RAN=RANU(1P)
  IR=IR+1
  DO 10 I=1,L
    IF(RAN.LE.RANIN(I)) GO TO 12
10  CONTINUE
    K=I
  RETURN
12  CONTINUE
    K=1
  RETURN
END

```

COMPLETION: 10 01:05:50

```
SEED: COLC(1) 000527; DATA(0) 000513; BLANK COMMON(2) 000000
```

DOJ
DOJ
DOJ
DOJ
DOJ

0001	16L	0001	000164	11L	0001	000221	12L	0001	000062	12L
0002	164G	0001	000212	173G	0001	000241	207G	0001	000315	273G
0003	27L	0001	000446	28L	0001	000450	301G	0000	000036	50L
0006	502F	0000	000070	502F	0001	000123	9L	0000	R 000070	50L
00022	1	0000	000250	1PJP	0000	I 000034	IMO	0000	I 000032	15L
00016	10	0000	1 000017	J	0000	I 000031	JPL	0000	I 000026	JP
00030	PDATA	0000	1 000012	PLUS	0000	P 000020	PMAX	0000	I 000015	PM
00020	RANGE	0000	R 000027	RECTA	0000	R 000025	SCALF	0000	I 000073	50L
00020	SYMBL	0000	I 000010	ZERO						

```

SUBROUTINE PLOT(MF,NL,DATA,SCALE,M,I0)
DIMENSION DATA(MD,ML), SCALE(ND,5),LSP(101),SP(101),SYMBL(10)
EQUIVALENCE(LSP,SP)
INTEGER SYMBL,PLUS,ZERO,BLANK,PNO,SP
DATA SP/101*' ' //
DATA SYMBL/'+', '-', '*', '/', '%', '^', '&', '@', '#', '$', '%', '&' //
1,'.', ',', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!', '!'
DATA PLUS/'+' //
DATA ZERO/'0' //
DATA BLANK/' ' //
C NL,NL ARE DATA DIM.
C M,N ARE DATA SIZE
C SCALE IS JUST SPACE
I0=0
DO 10 J=1,N
PMAX=0.
QMAX=0.
DO 8 I=1,M
IF(DATA(I,J).GT.PMAX) PMAX=DATA(I,J)
IF(DATA(I,J).LT.QMAX) QMAX=DATA(I,J)
8 CONTINUE
SCALE(J,2)=PMAX
SCALE(J,3)=QMAX
RANGE=PMAX-QMAX
SCALE(J,4)=RANGE
IRP=0
SCALE=10.
IF(RANGE.LT.1.) SCALE=.1

```



```

      9 CONTINUE
      IF (RANGE.LT.1.0) GO TO 11
      UR=RANGE
      IF (UR-RANGE.LT.0.0) GO TO 10
11 CONTINUE
      RANGE=RANGE/SCALE
      IRP=IRP+1
      GO TO 9
10 CONTINUE
      DO 14 J=1,N
      SCALE(J,1)=1.0
      IF (2.*RANGE.LT.10.) SCALE(J,1)=2.0
      RFACT=1.0
      DO 12 I=1,IRP
      IF (I.EQ.1) GO TO 12
      RFACT=RFACT*SCALE
12 CONTINUE
      SCALE(J,5)=RFACT/SCALE(J,1)
14 CONTINUE
      SP(1)=PLUS
      DO 20 J=1,N
      QMAX=SCALE(J,3)
      AQMAX=ABS(QMAX)
      JPL=AQMAX*100.*SCALE(J,1)
      IPL=JPL+1
      SP(IPL)=ZERO
      SYMB=SYMBL(J)
      WRITE(10,500) SYMB,SP,SCALE(J,5)
500 FORMAT( 10X,A1,10X,101A1,210.5)
501 FORMAT( 21X, 101H+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+ )
      1+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+ )
      SP(IPL)=BLANK
20 CONTINUE
      WRITE(10,501)
      INO=0
      DO 30 I=1,M
      PNO=BLANK
      DO 25 J=1,N
      SYMB=SYMBL(J)
      PDATA=(DATA(I,J)-SCALE(J,3))/SCALE(J,4)
      JPL=PDATA*100./SCALE(J,5)
      IF (SCALE(J,5).LT.1.0) JPL=PDATA*100.*SCALE(J,5)
      IPL=JPL+1
      IF (IPL.GT.101) IPL=101
      SP(IPL)=SYMB
25 CONTINUE
      INO=INO+1
      IF (INO.LT.5) GO TO 27
      PNO=I
      INO=0
      WRITE(10,502) PNO,SP
502 FORMAT( 10X,I6,5X,101A1 )
      GO TO 28
27 CONTINUE
      WRITE(10,503) PNO,SP
503 FORMAT( 10X,A6,5X,101A1 )
28 CONTINUE
      DO 29 J=2,101
      SP(J)=BLANK
29 CONTINUE
      SP(1)=PLUS
30 CONTINUE

```

RETURN

216

END

COMPILATION: NO DIAGNOSTICS.

04-11:04

PROG SIZE(1/D)=5501/7437

LEVEL 70-1

CTION - TIME 1.257 SECONDS

INITIAL STATE AND INPUT PATTERN

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.1000000+01				
.2500000-01	.5000000-01	.7500000-01	.9900000-01	.9500000+00

ITERATION 20

STATE, OUTPUT-MEASURE, ACCUM. STATE AND EPS				
.2941676-02	.3003804-01	.7266410-01	.5525000-01	.5328462-01
.2007909-01	.3209978-01	.3293716-01	.3454145-01	.3676768-01
.4226986-01	.5744957-01	.6122396-01	.2795379+00	.6392842-01
.1249801+00				
.2958345+02	.1139900+02	.3154093+02	.3607105+02	.2223392+02
.4958104+02	.1331978+02	.2439047+02	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				
.4599010-02	.5697771-01	.1253677+00	.9808998-01	.8844797-01
.6546162-01	.8504095-01	.8458944-01	.8890079-01	.6969392-01
.8367875-01	.9896832-01	.1424325+00	.4423416+00	.1751887+00
.2897074+00				
.1284741-02	.3098070-02	.1996043-01	.1241002-01	.1812127-01
.2530343-01	.2085040-01	.1871513-01	.1981788-01	.3841420-02
.8009816-00	.1593081-01	.1998453-01	.1162341+00	.4733186-01
.3970521-01				

INTEGRATED SUM IN SHORT AND LONG TIME PERIODS

.5651724-01	.6065960+00	.1407304+01	.1041566+01	.1027935+01
.3417516+00	.3790820+00	.3661086+00	.3855724+00	.7415677+00
.8044606+00	.1029089+01	.1383208+01	.5755012+01	.1388426+01
.3285799+01				

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				

.5651724-01	.6065960+00	.1407304+01	.1041566+01	.1027935+01
.3417516+00	.3790820+00	.3661086+00	.3855724+00	.7415677+00
.8044606+00	.1029089+01	.1383208+01	.5755012+01	.1388426+01
.3285799+01				

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				

*** STEADY STATE ***

ITERATION 37

STATE, OUTPUT-MEASURE, ACCUM. STATE AND EPS				
.3164943-02	.2859068-01	.7436777-01	.5414658-01	.5923331-01
.2007909-01	.3209978-01	.3293716-01	.3454145-01	.3676768-01
.4122540-01	.5758662-01	.6774565-01	.2975132+00	.6009481-01
.1119450+00				
.3035057+02	.1168930+02	.2881262+02	.3527657+02	.2289457+02
.5116288+02	.1335600+02	.2528291+02	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				

Example 2

INITIAL STATE AND INPUT PATTERN

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.1000000+01				
.1000000+00	.2000000+00	.8000000+00	.9000000+00	.9900000+00

ITERATION 20

STATE, OUTPUT-MEASURE, ACUM. STATE AND EPS

.1886948-02	.9320746-02	.2992015-01	.1493363-01	.1295812-01
.9785303-01	.1239974+00	.1071937+00	.1013401+00	.1917536-01
.2050570-01	.5578092-01	.5122744-01	.3239308+00	.3110662-01
.1380299-01				
.8632607+02	.1579866+02	.3109319+02	.2663503+02	.2222333+02
.4303337+02	.4066032+02	.2583262+02	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				
.2899963-02	.5310139-01	.7998384-01	.5021623-01	.2431538-01
.1928120+00	.2422404+00	.2101905+00	.1996035+00	.3804168-01
.3757904-01	.5656297-01	.1565346+00	.5187754+00	.9647305-01
.4059050-01				
.8739152-00	.3433090-01	.2014855-01	.2034898-01	.1600853-02
.2894095-02	.5754448-02	.4196932-02	.3076658-02	.3000478-03
.3432571-02	.1499087-01	.5412971-01	.1390661+00	.3425981-01
.1298452-01				

INTEGRATED SUM IN SHORT AND LONG TIME PERIODS

.2284759-01	.4453766+00	.7769937+00	.5132435+00	.3447829+00
.1136964+01	.1316515+01	.1198229+01	.1206753+01	.4241055+00
.4770019+00	.7123168+00	.1667331+01	.6542792+01	.1280409+01
.1934334+01				

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				

.2284759-01	.4453766+00	.7769937+00	.5132435+00	.3447829+00
.1136964+01	.1316515+01	.1198229+01	.1206753+01	.4241055+00
.4770019+00	.7123168+00	.1667331+01	.6542792+01	.1280409+01
.1934334+01				

.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				

ITERATION 40

STATE, OUTPUT-MEASURE, ACUM. STATE AND EPS

.1332522-02	.1365034-01	.3754130-01	.1894943-01	.1621867-01
.2061277-01	.2540588-01	.3831036-01	.3620434-01	.3008134-01
.3356862-01	.6479411-01	.1095495+00	.4591194+00	.4806601-01
.3689472-01				
.3759424+02	.1761205+02	.2418360+02	.2926233+02	.3126886+02
.8550596+02	.1437010+02	.3783236+02	.5000000	.0000000
.0000000	.0000000	.0000000	.0000000	.0000000
.0000000				
.3548429-02	.4728425-01	.8238060-01	.5854961-01	.3541636-01
.5364156-01	.7975395-01	.7848944-01	.8035703-01	.6372030-01

RAM

ED: CODE(1) 001472; DATA(0) 001434; BLANK COMMON(2) 000000

REFERENCES (BLOCK, NAME)

LR
ENTRS
ROUS
OZ\$
NDUS
OL\$
STOPS

OPTIMIZATION PROGRAM

ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

00154 122G	0001	000134 133G	0001	000141 137G	0001	000152 145
00171 157G	0001	000173 162G	0001	000175 166G	0001	000217 177
00241 215G	0001	000271 222G	0001	000272 225G	0001	000304 234
00404 252G	0001	000425 263G	0001	000427 266G	0001	000431 270
00471 313G	0001	000472 313G	0001	000503 321G	0001	000504 324
00533 34L	0001	000566 350G	0001	000567 353G	0001	000623 363
00552 377G	0001	000733 412G	0001	000740 416G	0001	000761 430
00777 441G	0001	001032 450G	0001	001073 472G	0001	000545 484
01147 431L	0001	001041 492L	0000	001046 500F	0000	001050 501
01101 502F	0000	001211 503F	0000	001167 504F	0000	001112 502
01122 503F	0000	001205 507F	0000	001130 503F	0000	001142 503
01156 511F	0000	001043 512F	0000	001044 513F	0001	001127 512
01147 525G	0001	001157 534G	0001	001172 540G	0001	001223 554
01247 570G	0001	001307 602G	0001	001314 606G	0001	001327 620
01362 627G	0001	001422 642G	0001	001427 646G	0001	001324 711
01446 78L	0001	001454 80L	0000 R	001311 B	0000 R	001215 D
01122 IJ	0000 I	001036 IU	0000 I	001040 IV	0000 I	001024 IX
01027 IXCS	0000 I	001033 J	0000 I	000776 JJ	0000 I	001042 JM
01125 JXS	0000 I	001035 K	0000 I	001037 KK	0000 I	001031 L
01041 N	0000 R	000000 P	0000 R	000632 Q	0000 R	001034 SU

* DIMENSION P(20,20),D(10,10),B(10),V(10),Q(10,10)
* EQUIVALENCE (D(1,7),B(1))
* DIMENSION JJ(20)
* ID=10
* JX=8
* IX=5
* JXS=JX+1
* IXS=IX+1
* IXSS=IXS+1
* 512 FORMAT(141)
* 513 FORMAT(////)
* READ(5,510) M
* 510 FORMAT(10I5)
* DO 150 L=1,M
* WRITE(6,512)
* WRITE(6,513)
* WRITE(6,501)

```

DO 10 I=1,IXS
  READ(5,500) (P(I,J),J=1,JX)
  WRITE(6,502) (P(I,J),J=1,JX)
10 CONTINUE
500 FORMAT(8F10.5)
501 FORMAT(/13X,63H THE FIRST FIVE SETS AND THE LAST SET ARE CHANNEL A
*CTIVITIES IN, /17X, 66HSTEADY STATE FOR EACH JOB TYPE AND IN REG. 0
*PERATION, RESPECTIVELY )
502 FORMAT(18X,1X,E12.6,1X,E12.6,1X,E12.6,1X,E12.6,1X,E12.6)
DO 20 I=1,IXS
DO 18 J=1,IX
SUM=0.
DO 16 K=1,JX
16 SUM=SUM+P(I,K)*P(J,K)
D(I,J)=SUM
18 CONTINUE
20 WRITE(6,502) (D(I,K),K=1,IX)
DO 31 I=1,IX
D(I,IXS)=1.
B(I)=D(IXS,I)
30 D(IXS,I)=1.
D(IXS,IXS)=0.
B(IXS)=1.
V(1)=5.
WRITE(6,505)
505 FORMAT(1H0,13X,33H THE SIMULTANEOUS EQ. TO SOLVE )
DO 32 I=1,IXS
DO 31 J=1,IXS
31 Q(I,J)=D(I,J)
Q(I,IXSS)=B(I)
32 WRITE(6,502) (Q(I,J),J=1,IXSS)
CALL GJR(9,ID,ID,IXS,IXSS,$200,JJ,V)
34 CONTINUE
WRITE(6,509)
DO 40 I=1,IXS
40 WRITE(6,502) (D(I,J),J=1,IXSS)
WRITE(6,506)
506 FORMAT(1H0,13X,23H UNIT MATRIX FOR CHECK )
DO 48 I=1,IXS
DO 46 J=1,IXS
SUM=0.
DO 44 K=1,IXS
44 SUM=SUM+D(I,K)*Q(K,J)
46 P(I,J)=SUM
48 WRITE(6,502) (P(I,J),J=1,IXS)
DO 480 I=1,ID
DO 480 J=1,ID
480 P(I,J)=0.
DO 482 I=1,IXS
DO 482 J=1,IXS
482 P(I,J)=Q(I,J)
J=J
DO 484 I=1,IX
IF(B(I).GE.0.) GO TO 484
J=J+1
K=IXS+J
P(K,I)=-1.
P(I,K)=-1.
B(K)=Q.
434 CONTINUE
IU=0
DO 485 I=1,K

```



```

DO 485 J=1,K
485 D(I,J)=P(I,J)
487 CONTINUE
KK=K+1
DO 486 I=1,K
486 P(I,KK)=Q(I,IXSS)
WRITE(6,503)
508 FORMAT(1H0,18X,45HIMPOSED NEG. PHI=ZERO COND. MATRIX TO SOLVE )
DO 488 I=1,K
488 WRITE(6,502) (P(I,J),J=1,KK)
V(1)=5.
CALL GJR(P,20,20,K,KK,$490,JJ,V)
WRITE(6,509)
509 FORMAT(1H0,18X,56HTHE INV. MATPIX AND SOLUTION(IN THE RIGHT MOST
*COLMN.) )
DO 489 I=1,K
489 WRITE(6,502) (P(I,J),J=1,KK)
IF(IU.EQ.1) GO TO 490
IV=J
IU=1
DO 491 I=1,K
DO 491 J=1,K
491 P(I,J)=D(I,J)
DO 492 I=1,IX
IF(P(I,KK).GE.-1.0E-6) GO TO 492
IV=IV+1
K=K+1
DO 493 N=1,ID
P(K,N)=D.
P(N,K)=D.
493 CONTINUE
P(K,IV)=-1.
P(IV,K)=-1.
492 CONTINUE
IF(IV.EQ.0) GO TO 490
GO TO 487
490 CONTINUE
WRITE(6,511)
511 FORMAT(1H0,18X,35HSTEADY STATE DIST. OF EACH JOB TYPE )
DO 49 I=1,K
49 B(I)=P(I,KK)
CONTINUE
JMAX=16
DO 50 I=1,IXS
READ(5,500) (P(I,J),J=1,JMAX)
50 WRITE(6,502) (P(I,J),J=1,JMAX)
DO 54 I=1,IXS
SUM=0.
DO 52 J=1,JMAX
52 SUM=SUM+P(I,J)
54 P(7,I)=SUM
DO 60 J=1,JMAX
SUM=0.
DO 58 I=1,IX
58 SUM=SUM+B(I)*P(I,J)
P(JX,J)=P(IXS,J)-SUM
60 V(J)=SUM
WRITE(6,504)
504 FORMAT(1H0,18X,35HMOST LIKELY STATE DISTRIB.,DIFF. AND/18X,
*24HCHECK SUM OF STATE DIST. )
WRITE(6,502) (V(J),J=1,JMAX)
WRITE(6,502) (P(JX,J),J=1,JMAX)

```

```

WRITE(6,502) (P(7,J),J=1,IXS)
V(1)=7.
C RE-INVERSE
CALL GJR(D,ID,ID,IXS,IXSS,$71,JJ,V)
WRITE(6,507)
507 FORMAT(1HJ,18X,11HRE-INVERSE )
DO 70 I=1,IXS
70 WRITE(6,502) (D(I,J),J=1,IXSS)
71 CONTINUE
WRITE(6,502) V(2)
DO 80 K=2,IXS
KK=K+1
DO 74 I=1,K
DO 72 J=1,K
72 D(I,J)=Q(I,J)
D(I,KK)=B(I)
74 CONTINUE
V(1)=7.
CALL GJR(D,ID,ID,K,KK,$78,JJ,V)
75 CONTINUE
DO 76 I=1,K
76 WRITE(6,502) (D(I,J),J=1,KK)
WRITE(6,502) V(2)
GO TO 80
78 WRITE(6,503)
GO TO 75
80 CONTINUE
GO TO 150
200 WRITE(6,503)
GO TO 34
150 CONTINUE
STOP
503 FORMAT(1H0,18X,10HOVERFLOW )
END

```

F COMPILATION: NO DIAGNOSTICS.

09-02:13

• PROG SIZE(I/D)=5566/3157

LEVEL 70-1

CTION - TIME 1.247 SECONDS

Example 1

THE FIRST FIVE SETS AND THE LAST SET ARE CHANNEL ACTIVITIES IN STEADY STATE FOR EACH JOB TYPE AND IN REG. OPERATION, RESPECTIVELY

.756134+02	.272408+02	.866407+02	.378083+02	.130941+02
.254157+02	.413912+02	.131621+02		
.304091+01	.223587+02	.738658+02	.543060+02	.213845+02
.324617+02	.608349+01	.185176+02		
.181061+02	.328651+02	.104139+02	.319533+02	.439975+02
.817112+02	.357778+01	.491109+02		
.169435+02	.172420+02	.499045+01	.307012+02	.337858+02
.726093+02	.331726+01	.387337+02		
.105067+02	.120804+02	.246345+02	.369883+02	.257948+02
.587673+02	.301147+01	.284840+02		
.303606+02	.116893+02	.288126+02	.352766+02	.228946+02
.511629+02	.133560+02	.252829+02		
.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.108923+05	.108055+05	.781891+04	.628938+04	.713546+04
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.627889+04	.628938+04	.112400+05	.947673+04	.739566+04
.698725+04	.713546+04	.937220+04	.789566+04	.717069+04
.392939+04	.709759+04	.883847+04	.745480+04	.683245+04

THE SIMULTANEOUS EQ. TO SOLVE

.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.100000+01	.892929+04			
.108923+05	.108055+05	.781891+04	.628938+04	.713546+04
.100000+01	.709759+04			
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.100000+01	.883847+04			
.627889+04	.628938+04	.112400+05	.947673+04	.739566+04
.100000+01	.745480+04			
.698725+04	.713546+04	.937220+04	.789566+04	.717069+04
.100000+01	.683245+04			
.100000+01	.100000+01	.100000+01	.100000+01	.100000+01
.000000	.100000+01			

THE INV. MATRIX AND SOLUTION (IN THE RIGHT MOST COLUMN.)

.236433-03	-.152084-02	.159450-02	-.453302-02	.422294-02
-.229657+03	.240471+00			
-.152085-02	.200814-01	-.229130-01	.650287-01	-.606762-01
.366271+01	.303864+00			
.159450-02	-.229130-01	.284156-01	-.774718-01	.703747-01
-.682965+01	-.772747+00			
-.453303-02	.650287-01	-.774718-01	.216785+00	-.199809+00
.146344+02	.187472+01			
.422294-02	-.606762-01	.703747-01	-.199809+00	.185887+00
-.102378+02	-.646307+00			
-.229658+00	.366272+01	-.682966+01	.146344+02	-.102378+02
-.257240+04	.567913+02			

UNIT MATRIX FOR CHECK

.100000+01	-.286847-06	.139990-06	-.763635-06	-.763635-06
-.116415-09				
.163913-05	.100001+01	.163913-05	.545382-05	.926852-05
.186265-08				
-.774860-05	-.393391-05	.399936+00	-.393391-05	-.113209-06

-.186265-08				
.178814-04	.331402-04	.483990-04	.100005+01	.331402-04
.745058-08				
-.104904-04	-.257492-04	-.562668-04	-.410080-04	.999974+00
-.745053-08				
-.305176-04	-.100708-02	-.296021-02	-.198364-02	-.305176-04
.100000+01				

IMPOSED NEG. PHI=ZERO COND. MATRIX TO SOLVE

.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.100000+01	.000000	.000000	.892989+04	
.108928+05	.108055+05	.781891+04	.628938+04	.713546+04
.100000+01	.000000	.000000	.709759+04	
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.100000+01	-.100000+01	.000000	.883847+04	
.527839+04	.628988+04	.112400+05	.947673+04	.789666+04
.100000+01	.000000	.000000	.745480+04	
.698725+04	.713546+04	.937220+04	.789666+04	.717069+04
.100000+01	.000000	-.100000+01	.683245+04	
.100000+01	.100000+01	.100000+01	.100000+01	.100000+01
.000000	.000000	.000000	.100000+01	
.000000	.000000	-.100000+01	.000000	.000000
.000000	.000000	.000000	.000000	
.000000	.000000	.000000	.000000	-.100000+01
.000000	.000000	.000000	.000000	

THE INV. MATRIX AND SOLUTION (IN THE RIGHT MOST COLUMN.)

.140488-03	-.142280-03	.000000	.179279-05	.000000
-.416979-02	.240098-02	-.236267-01	.253886+00	
-.142280-03	.273925-03	.000000	-.131644-03	.000000
.417969+00	-.328442-01	.338848+00	-.110244+00	
-.181018-10	.750424-10	.000000	.612448-09	.000000
.269110-06	-.100000+01	.327059-07	.471727-07	
.179284-05	-.131644-03	.000000	.129852-03	.000000
.585200+00	.103044+01	.684779+00	.635870+00	
.164356-10	-.330757-09	.000000	-.149012-08	.000000
.479415-06	-.127577-06	-.100000+01	.863941-07	
-.416965-02	.417967+00	.000000	.586199+00	.000000
-.815805+04	.166629+04	-.575763+03	-.858713+03	
.240093-02	-.328436-01	-.100000+01	.103044+01	.000000
.166629+04	-.564128+03	.213572+03	.297895+03	
-.236267-01	.338848+00	.000000	.684779+00	-.100000+01
-.575763+03	.213572+03	-.862353+02	-.109303+03	

STEADY STATE DIST. OF EACH JOB TYPE

.123320-02	.896333-01	.788959-01	.527624-01	.431910-02
.993446-01	.130079+00	.109387+00	.104376+00	.292177-01
.163037-01	.115376-01	.146032+00	.514521-01	.619545-01
.134574-01				
.121330-02	.166783-01	.248650-01	.245823-01	.193151-01
.485819-02	.709440-02	.102043-01	.172421-01	.228897-01
.481537-01	.480595-01	.121347+00	.119382+00	.170390+00
.343209+00				
.708100-03	.115243-01	.288509-01	.148763-01	.141407-01
.000000	.000000	.000000	.000000	.187170-01
.208993-01	.396441-01	.101274+00	.651501+00	.824743-01
.153885-01				
.317030-02	.304333-01	.799132-01	.531532-01	.512953-01
.000000	.000000	.000000	.000000	.359977-01
.396153-01	.395386-01	.562302-01	.517653+00	.512886-01
.416508-01				
.361780-02	.324768-01	.844760-01	.615063-01	.672844-01

.000000	.000000	.000000	.000000	.439690-01	227
.468289-01	.608702-01	.655945-01	.337952+00	.682630-01	
.127161+00					
.318493-02	.285907-01	.743678-01	.541466-01	.592331-01	
.200791-01	.320998-01	.329372-01	.345414-01	.387077-01	
.412254-01	.535866-01	.577456-01	.297513+00	.600948-01	
.111945+00					

MOST LIKELY STATE DISTRIBUTION DIFF. AND
CHECK SUM OF STATE DIST.

.246276-02	.439469-01	.735862-01	.499042-01	.358431-01
.257573-01	.338072-01	.283958-01	.284004-01	.328313-01
.346400-01	.333700-01	.862400-01	.355391+00	.671844-01
.677373-01				
.722169-03	-.153563-01	.781587-03	.424238-02	.233900-01
-.567853-02	-.170746-02	.404032-02	.614106-02	.587547-02
.658543-02	.202166-01	-.284944-01	-.578774-01	-.708960-02
.442071-01				
.100000+01	.100000+01	.999999+00	.100000+01	.100000+01
.999999+00				

RE-INVERSE

.236433-03	-.152084-02	.159450-02	-.453302-02	.422294-02
-.229657+00	.197207+03			
-.152035-02	.200814-01	-.229130-01	.650287-01	-.606762-01
.366271+01	-.314518+04			
.159450-02	-.229130-01	.284156-01	-.774718-01	.703747-01
-.682965+01	.586466+04			
-.453303-02	.650287-01	-.774718-01	.216785+00	-.199809+00
.146344+02	-.125666+05			
.422294-02	-.606762-01	.703747-01	-.199809+00	.185887+00
-.102378+02	.879120+04			
-.229653+00	.366272+01	-.682966+01	.146344+02	-.102378+02
-.267240+04	.229483+07			
.258376+02				
.999684+03	-.100776+04	.336673+07		
-.308728-02	.320477-02	-.106834+02		
.238037+01				
.679531+03	-.688353+03	.515602+01	.233083+07	
.175832-02	-.162350-02	-.780621-04	.499513+01	
-.533051-02	.531144-02	.853741-04	-.172530+02	
.117430+02				
.486577+02	-.492222+02	-.239757+01	.327474+01	.109195+06
-.232849-02	.251939-02	-.126987-03	.212104-04	-.939439+01
.705115-01	-.715714-01	.993803-03	-.393619-03	.249785+03
-.839106-01	.850620-01	-.100452-02	.435494-03	-.295446+03
.194820+02				
-.162198+00	.116224+01	-.138119+01	.366642+01	-.323086+01
-.862652+05				
.192904-02	-.187458-02	-.215624-03	-.129479-04	.231760-03
.765145+01				
-.290897-02	.420214-02	.252236-02	.195438-03	-.485891-02
-.441690+02				
.735794-02	-.913143-02	-.290465-02	-.296757-03	.604008-02
.599649+02				
-.621884-02	.641613-02	.129470-03	.498940-04	-.411558-03
-.248933+02				
.272776+02				
-.140335-01	.905394-01	-.949757-01	.270009+00	-.251539+00
.136791+02	.313649+08			
.222207-02	-.399475-02	.232905-02	-.673250-02	.617613-02
.270632-01	.622710+05			

-.521721-02	.209029-01	-.175222-01	.531258-01	-.512892-01	228
-.213178+00	-.490496+06				
.102575-01	-.301104-01	.222747-01	-.667863-01	.643645-01	
.267788+00	.616162+06				
-.630840-02	.706612-02	-.648258-03	.210359-02	-.221305-02	
-.827132-02	-.190544+05				
-.272211+01	.196952+02	-.236386+02	.624206+02	-.547554+02	
-.251401+03	-.578391+09				
.217505+02					

Example 2

THE FIRST FIVE SETS AND THE LAST SET ARE CHANNEL ACTIVITIES IN STEADY STATE FOR EACH JOB TYPE AND IN REG. OPERATION. RESPECTIVELY

.766134+02	.272408+02	.866407+02	.378083+02	.130941+02
.254154+02	.413912+02	.131621+02		
.304091+01	.223587+02	.733658+02	.543060+02	.213845+02
.324617+02	.608849+01	.185176+02		
.181051+02	.323551+02	.104139+02	.319533+02	.439975+02
.817112+02	.357778+01	.491109+02		
.169435+02	.172420+02	.499045+01	.307012+02	.337858+02
.726093+02	.331726+01	.387337+02		
.105057+02	.120804+02	.246345+02	.369883+02	.257948+02
.587673+02	.301147+01	.284840+02		
.373016+02	.299509+02	.165559+02	.308219+02	.388504+02
.712555+02	.139850+02	.433638+02		
.180994+05	.108928+05	.782202+04	.627389+04	.698725+04
.108928+05	.108055+05	.781891+04	.628986+04	.713546+04
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.627889+04	.628988+04	.112400+05	.947673+04	.789666+04
.698725+04	.713546+04	.937220+04	.789666+04	.717069+04
.970544+04	.771183+04	.125284+05	.103898+05	.876855+04

THE SIMULTANEOUS EQ. TO SOLVE

.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.100000+01	.970544+04			
.108928+05	.108055+05	.781891+04	.628988+04	.713546+04
.100000+01	.771183+04			
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.100000+01	.125284+05			
.627889+04	.628988+04	.112400+05	.947673+04	.789666+04
.100000+01	.103898+05			
.698725+04	.713546+04	.937220+04	.789666+04	.717069+04
.100000+01	.876355+04			
.100000+01	.100000+01	.100000+01	.100000+01	.100000+01
.000000	.100000+01			

THE INV. MATRIX AND SOLUTION (IN THE RIGHT MOST COLUMN.)

.236433-03	-.152084-02	.159450-02	-.453302-02	.422294-02
-.229657+00	.245005+00			
-.152035-02	.200814-01	-.229130-01	.650287-01	-.606762-01
.366271+01	.293651+00			
.159450-02	-.229130-01	.284156-01	-.774718-01	.703747-01
-.682965+01	.116819+00			
-.453303-02	.650287-01	-.774718-01	.216785+00	-.199809+00
.146344+02	.184404+01			
.422294-02	-.606762-01	.703747-01	-.199809+00	.135837+00
-.102378+02	-.149931+01			
-.229653+00	.366272+01	-.682965+01	.146344+02	-.102378+02
-.267240+04	.576922+02			

UNIT MATRIX FOR CHECK

.100000+01	-.286847-06	.189990-06	-.763685-06	-.763685-06
-.116415-09				
.163913-05	.100001+01	.163913-05	.545382-05	.926852-05
.136265-03				
-.774860-05	-.393391-05	.999996+00	-.293391-05	-.119209-06

- .126265-08					
.178814-04	.331402-04	.483990-04	.100005+01	.331402-04	
.745058-08					
-.104904-04	-.257492-04	-.562668-04	-.410080-04	.999974+00	
-.745058-08					
-.305176-04	-.100708-02	-.296021-02	-.198364-02	-.305176-04	
.100000+01					

IMPOSED NEG. PHI=ZERO COND. MATRIX TO SOLVE

.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.100000+01	.000000	.970544+04		
.108928+05	.108055+05	.781891+04	.628988+04	.713546+04
.100000+01	.000000	.771183+04		
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.100000+01	.000000	.125284+05		
.627889+04	.628988+04	.112400+05	.947673+04	.789666+04
.100000+01	.000000	.103893+05		
.698725+04	.713546+04	.937220+04	.789666+04	.717069+04
.100000+01	-.100000+01	.876855+04		
.100000+01	.100000+01	.100000+01	.100000+01	.100000+01
.000000	.000000	.100000+01		
.000000	.000000	.000000	.000000	-.100000+01
.000000	.000000	.000000		

THE INV. MATRIX AND SOLUTION (IN THE RIGHT MOST COLUMN.)

.140498-03	-.142420-03	-.425608-05	.617845-05	.000000
.292209-02	-.227177-01	.279065+00		
-.142420-03	.275837-03	.582213-04	-.191638-03	.000000
.320956+00	.326414+00	-.195745+00		
-.425608-05	.582200-04	.177265-02	-.182661-02	.000000
-.295375+01	-.378508+00	.684240+00		
.617842-05	-.191637-03	-.182661-02	.201207-02	.000000
.362987+01	.107469+01	.232439+00		
.158923-10	-.323330-09	.226148-09	-.172315-03	.000000
.102586-06	-.100000+01	.122361-09		
.292207-02	.320956+00	-.295375+01	.362987+01	.000000
-.323625+04	.550754+02	-.246830+02		
-.227177-01	.326414+00	-.378598+00	.107489+01	-.100000+01
.550753+02	-.537960+01	.806569+01		

IMPOSED NEG. PHI=ZERO COND. MATRIX TO SOLVE

.180994+05	.108928+05	.782202+04	.627889+04	.698725+04
.100000+01	.000000	.000000	.970544+04	
.108928+05	.108055+05	.781891+04	.628988+04	.713546+04
.100000+01	.000000	-.100000+01	.771183+04	
.782202+04	.781891+04	.135746+05	.112400+05	.937220+04
.100000+01	.000000	.000000	.125284+05	
.627889+04	.628988+04	.112400+05	.947673+04	.789666+04
.100000+01	.000000	.000000	.103893+05	
.698725+04	.713546+04	.937220+04	.789666+04	.717069+04
.100000+01	-.100000+01	.000000	.876855+04	
.100000+01	.100000+01	.100000+01	.100000+01	.100000+01
.000000	.000000	.000000	.100000+01	
.000000	.000000	.000000	.000000	-.100000+01
.000000	.000000	.000000	.000000	
.000000	-.100000+01	.000000	.000000	.000000
.000000	.000000	.000000	.000000	

THE INV. MATRIX AND SOLUTION (IN THE RIGHT MOST COLUMN.)

.669634-04	.000000	.253047-04	-.927681-04	.000000
.168638+00	.145816+00	.516320+00	.177999+00	
-.181899-11	.000000	.454747-12	-.181899-11	.000000

.372529-08	.372529-08	-.100000+01	-.186265-03	
.256041-04	.000000	.176036-02	-.178616-02	.000000
-.302143+01	-.447483+00	-.211067+00	.725555+00	
-.927674-04	.000000	-.178616-02	.187893-02	.000000
.385285+01	.130167+01	.694746+00	.964454-01	
-.151049-09	.000000	.294394-09	-.194778-08	.000000
.473803-06	-.100000+01	.117218-05	-.229325-06	
.168638+00	.000000	-.302149+01	.385286+01	.000000
-.360970+04	-.324730+03	-.116357+04	.202379+03	
.145816+00	.000000	-.447484+00	.130167+01	-.100000+01
-.324730+03	-.391543+03	-.118336+04	.239701+03	
.516320+00	-.100000+01	-.211071+00	.694751+00	.000000
-.116357+04	-.118336+04	-.362533+04	.709539+03	

STEADY STATE DIST. OF EACH JOB TYPE

.123320-02	.896333-01	.788959-01	.527524-01	.431910-02
.993446-01	.130079+00	.109387+00	.104376+00	.292177-01
.163037-01	.115376-01	.146032+00	.514521-01	.619545-01
.134574-04				
.121340-02	.166783-01	.248550-01	.245823-01	.193151-01
.485819-02	.709440-02	.102043-01	.172421-01	.228897-01
.481587-01	.480695-01	.121347+00	.119382+00	.170890+00
.243209+00				
.708100-03	.115243-01	.288509-01	.148763-01	.141407-01
.000000	.000000	.000000	.000000	.187170-01
.206993-01	.396441-01	.101274+00	.651501+00	.824743-01
.153885-01				
.317030-02	.304333-01	.799132-01	.531532-01	.512953-01
.000000	.000000	.000000	.000000	.359977-01
.396153-01	.395386-01	.562802-01	.517663+00	.512886-01
.416500-01				
.351780-02	.324768-01	.844760-01	.615063-01	.672844-01
.000000	.000000	.000000	.000000	.439690-01
.468289-01	.608702-01	.655945-01	.337952+00	.682630-01
.127161+00				
.622210-03	.101267-01	.253518-01	.130721-01	.124257-01
.196580-01	.336498-01	.341016-01	.338500-01	.164470-01
.183646-01	.348360-01	.889922-01	.572490+00	.724726-01
.135222-01				

MOST LIKELY STATE DISTRIB., DIFF. AND
CHECK SUM OF STATE DIST.

.103904-02	.272513-01	.426836-01	.253116-01	.159759-01
.176832-01	.231538-01	.194708-01	.185788-01	.222528-01
.218873-01	.346310-01	.104902+00	.531785+00	.758162-01
.175777-01				
-.416827-03	-.171246-01	-.173318-01	-.122395-01	-.355014-02
.197480-02	.104959-01	.146308-01	.152712-01	-.580580-02
-.352265-02	.205074-03	-.159094-01	.407050-01	-.334356-02
-.405541-02				
.100000+01	.100000+01	.999999+00	.100000+01	.100000+01
.999983+00				

RE-INVERSE

.236433-03	-.152084-02	.159450-02	-.453302-02	.422294-02
-.229657+00	-.465920+02			
-.152085-02	.200814-01	-.229130-01	.650287-01	-.606762-01
.366271+01	.743079+03			
.159450-02	-.229130-01	.284156-01	-.774718-01	.703747-01
-.682965+01	-.138558+04			
-.453303-02	.650287-01	-.774718-01	.216725+00	-.199809+00
.146344+02	.296898+04			

.422294-02	-.605762-01	.703747-01	-.199809+00	.185837+00	232
-.102378+02	-.207701+04				
-.223653+00	.366272+01	-.682966+01	.146344+02	-.102378+02	
-.267240+04	-.542178+06				
.258375+02					
-.767842+02	.774046+02	.610952+05			
.246083-03	-.155532-03	-.127038+00			
.494681+01					
-.605297+02	.816224+02	-.610939+00	.652504+05		
-.329590-03	.492747-03	-.939021-04	.511615+00		
.812153-03	-.914581-03	.132475-03	-.901001+00		
.138759+02					
.572227+02	-.578365+02	-.281361+01	.385118+01	-.303394+05	
-.428795-02	.450158-02	-.304352-04	-.110665-03	.325842+01	
.130043+00	-.131793+00	-.193956-02	.361294-02	-.905774+02	
-.154604+00	.156576+00	.247887-02	-.432231-02	.107284+03	
.193199+02					
-.101588+00	.727941+00	-.865077+00	.229638+01	-.202358+01	
.127652+05					
.192591-02	-.185213-02	-.242303-03	.578725-04	.219353-03	
-.141405+01					
-.290064-02	.414244-02	.259329-02	.713407-05	-.469298-02	
.938862+01					
.734171-02	-.901513-02	-.304286-02	.701194-04	.571678-02	
-.144905+02					
-.620919-02	.634893-02	.211709-03	-.168411-03	-.219188-03	
.466895+01					
.277455+02					
-.393571-01	.253161+00	-.265421+00	.754571+00	-.702954+00	
.382282+02	-.207223+08				
.212440-02	-.336648-02	.167035-02	.485988-02	.443161-02	
.121932+00	-.661377+05				
-.445415-02	.159947-01	-.123763-01	.384964-01	-.376606-01	
-.954316+00	.517631+06				
.929672-02	-.239305-01	.157956-01	-.483665-01	.472047-01	
.170095+01	-.651413+06				
-.627513-02	.635243-02	-.424213-03	.146665-02	-.161963-02	
-.405402-01	.219937+05				
-.182604+01	.139314+02	-.175957+02	.452414+02	-.387511+02	
-.112173+04	.608427+09				
.207228+02					

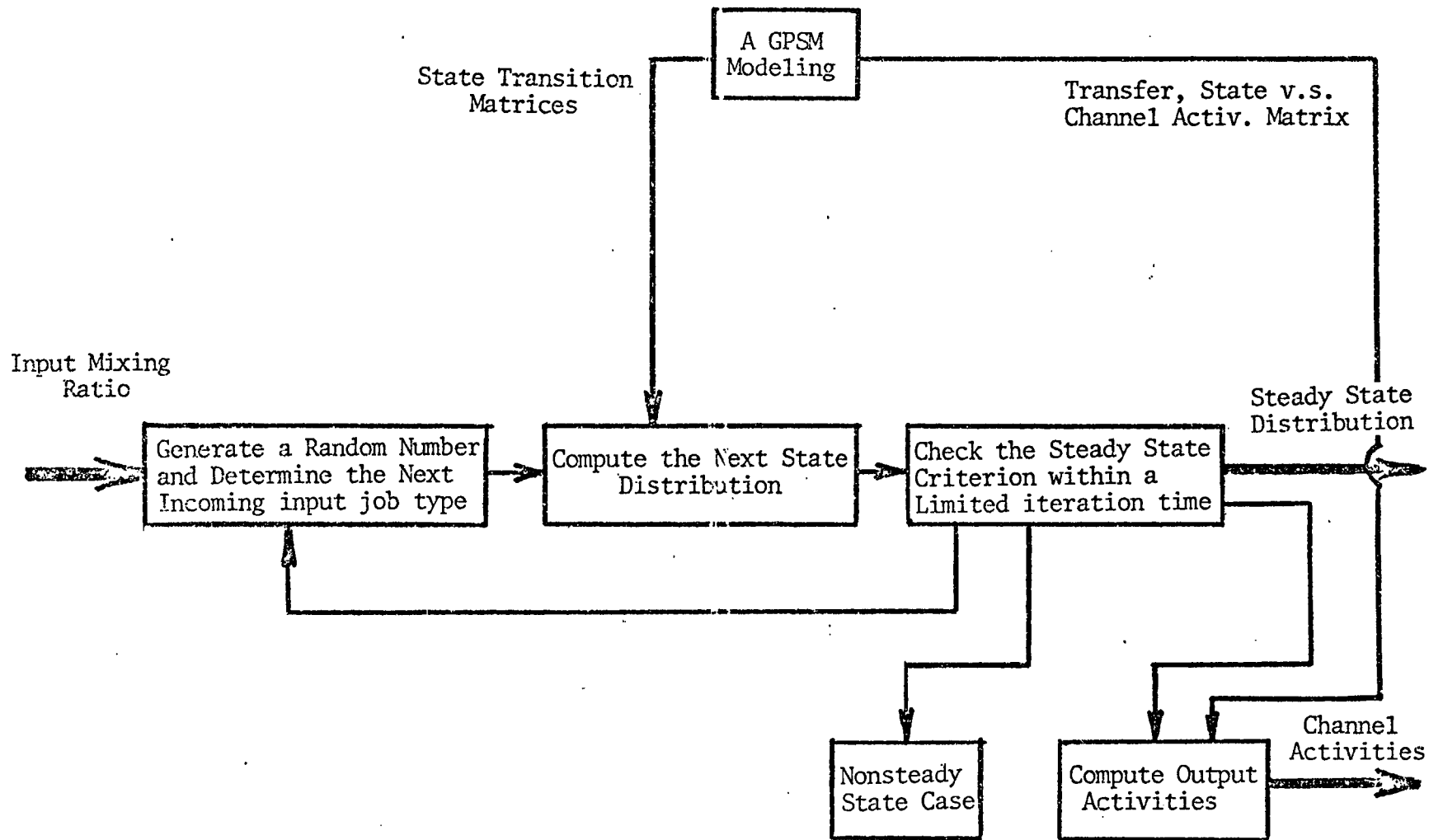


Figure A.1 A Block Diagram of Simulation

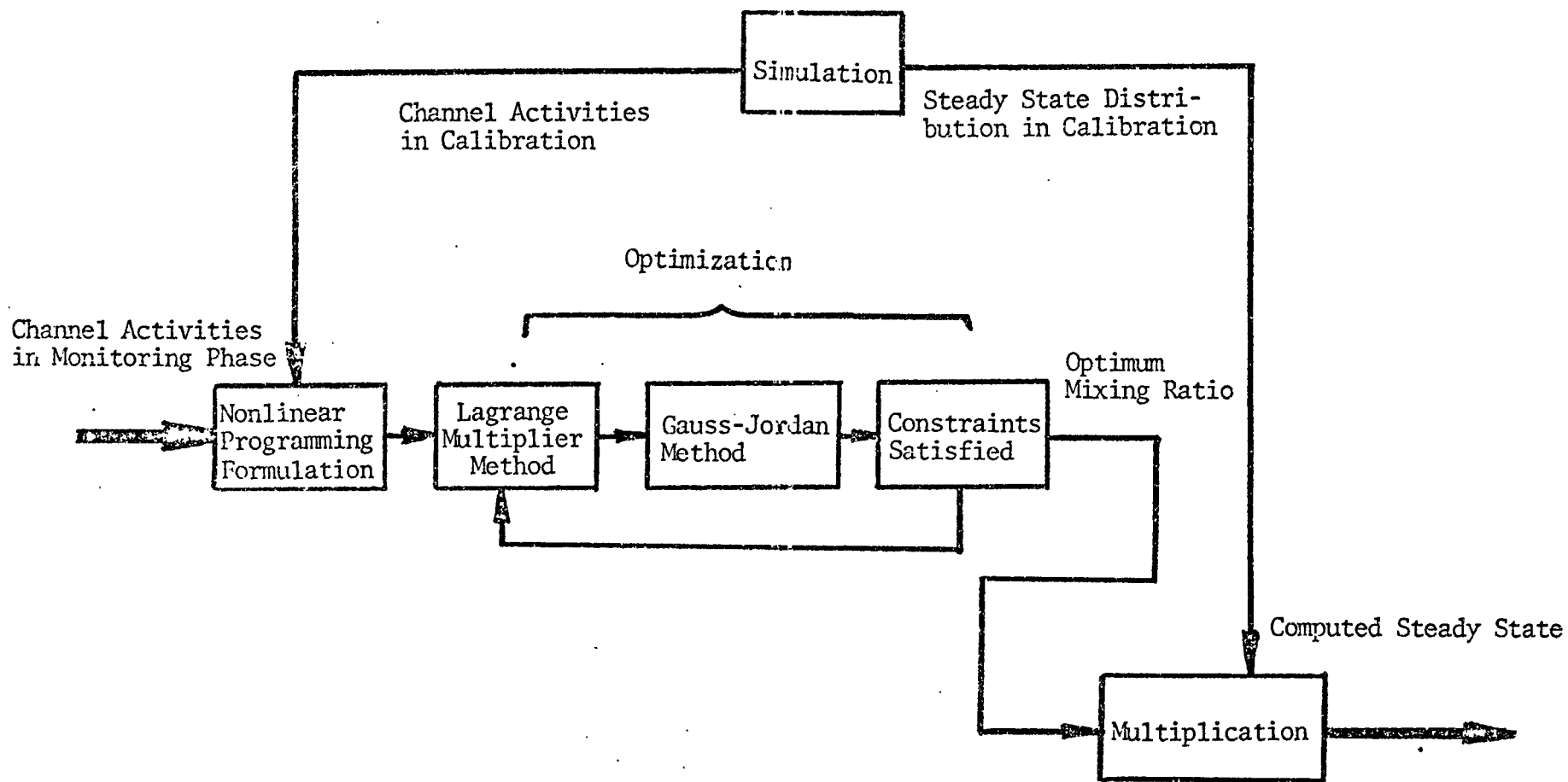


Figure A.2 A Block Diagram of Estimation Method

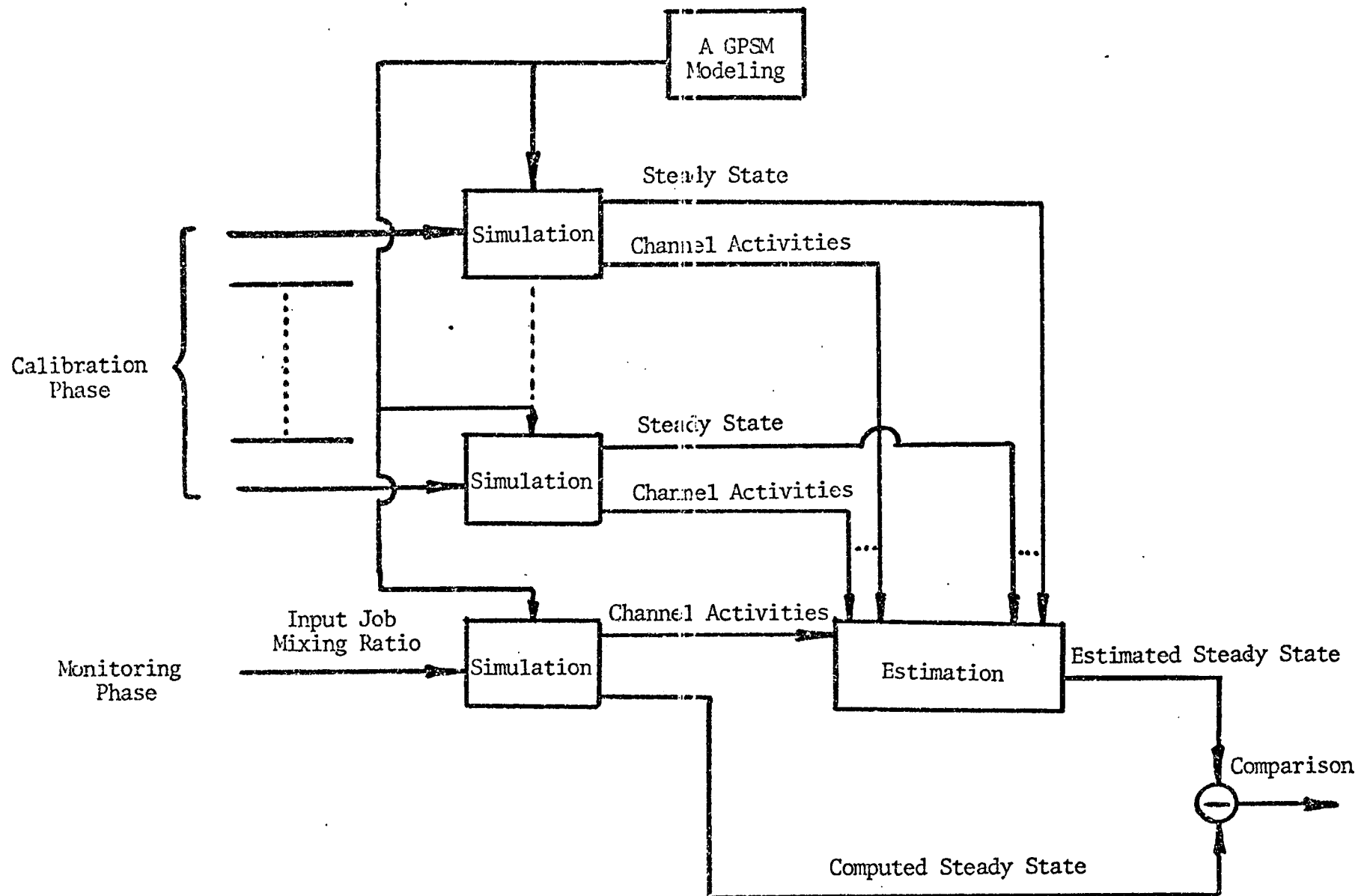


Figure A.3 An Overall Diagram of Simulation and Estimation