THE ANALYSIS OF INTERCONNECTIONS OF SEQUENTIAL MACHINES BY POLYNOMIAL FUNCTIONS

by

Bobby R. Hunt

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 6 7

THE UNIVERSITY OF ARIZONA

GRADUATE COLLEGE

I hereby recommend that this dissertation prepared under my direction by **Bobby Ray Hunt**

entitled **The Analysis of Interconnections of Sequential Machines by Polynomial Functions**

be accepted as fulfilling the dissertation requirement of the degree of **Doctor of Philosophy**

A. Wayne Wymore — 6/27/67

Dissertation Director — Date

After inspection of the dissertation, the following members of the Final Examination Committee concur in its approval and recommend its acceptance:*

A. Wayne Wymore	6/27/67
S. R. Browning	6/28/67
Sidney J. Gabrini	6/28/67
D. E. Thelen	6/28/67
G. R. Peters	7/3/67

*This approval and acceptance is contingent on the candidate's adequate performance and defense of this dissertation at the final oral examination. The inclusion of this sheet bound into the library copy of the dissertation is evidence of satisfactory performance at the final examination.

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: Bobby R. Hurst

ACKNOWLEDGEMENTS

The author wishes to thank the faculty and staff of the Department of Systems Engineering, the University of Arizona, for their aid and understanding. Special thanks are expressed to Dr. A. Wayne Wymore, whose patience and personal counsel in the writing of this dissertation were without limit.

The author also gratefully acknowledges the support of the National Aeronautics and Space Administration.

The author lovingly dedicates this document to his wife, Susie. The Department of Systems Engineering and NASA made everything possible, but she made it enjoyable.

TABLE OF CONTENTS

TABLE OF CONTENTS--Continued

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS--Continued

ABSTRACT

The algebraic structure of a finite state system is classically defined as a quintuple. A new representation of this algebraic structure is introduced. This representation, called the real number representation, is based upon the use of finite sets of real numbers for the state, input, and output sets of a finite state system and the description of the state-transition and output functions by polynomial functions. Multiple input systems are similarly defined.

The interconnection of a set of finite state systems is formally defined and this definition is utilized to formally define the structure of the system which results from the interconnection of a set of finite state systems, called the resultant. Using the polynomial functions, a scheme is demonstrated to derive the real number representation of the resultant of an interconnection of finite state systems given the real number representations of each system in the interconnection. A synthesis procedure is also derived to illustrate the use of real number representations to synthesize a set of systems whose interconnected behavior possesses a resultant which realizes the given system. The procedure utilizes the

polynomial functions to specify both the structure of the systems and the interconnection of them.

Four illustrative examples are worked to show the type of computations involved in using the polynomial functions. Conclusions are presented and a set of suggested extensions is set forth.

CHAPTER 1

INTRODUCTORY COMMENTS

1.1. The "Systems Age" and System Theory.

Ellis and Ludwig, in the first chapter of their book Systems Philosophy (Ref. 5) observe that even though the journalistic media refer to the modern era as the "Jet Age", the "Nuclear Age", or the "Space Age", the most accurate characterization that can be given to the world since 1945 is the "Systems Age". Experience in everyday life bears out this assertion. The word "system" appears everywhere, both in technical and non-technical usage.

The concept of a system is not new. Even an older edition of Webster's dictionary (Ref. 23) reveals a definition that does not seem at odds with current thought: "System: n. A combination of parts into a whole; an orderly arrangement according to some common law." As acceptable as this definition may be to the intuition, it is lacking in specificity. For engineering purposes, it is necessary to give quantitative content and mathematical concreteness to the concept of system. The mathematical discipline which results from such considerations has become known as "system theory".

System theory, as a technical discipline in its own right, is a very new field. At least two researchers, Linvill (Ref. 19) and Zadeh (Ref. 28), suggest that system theory has grown up from a large class of closely related problems appearing in the field of circuit theory and associated branches of electrical engineering. More recent approaches, however, have been to discard the circuit or control theory aspects of system theory and employ mathematics as the starting point. The latest development along these lines is in the book by Wymore, A Mathematical Theory of Systems Engineering (Ref. 24). The difference in the two approaches is that Linvill and Zadeh look at a large body of electrical engineering and make generalizations by the process of induction. Wymore does just the opposite in a process of deduction. He begins by constructing a completely general mathematical definition of a system. Then by introducing more and more definitions and hypotheses he draws successively more specific conclusions about systems. In this way Wymore approaches such system-theoretic problems as the comparison of systems, interconnection of systems, decomposition of systems, and modeling of systems.

1.2. System Interaction and Finite State Systems.

Systems engineering is made up of two distinct halves: systems analysis and systems design. In analysis

a collection of systems which interact with one another is studied in order to derive the overall behavior of the collection. In design the goal is to specify a collection of systems which will interact with one another to produce a desired overall behavior. The important concept in either case is interaction. If the rules of interaction of a collection of systems are unknown then both analysis and design are impossible.

As a result, the problem of interactions between systems is a very important topic in system theory. There are many ramifications to this topic. Since system theory is mathematical in scope and intent, then the concept of interaction must be given mathematical content by defining precisely what is meant by two or more systems being interconnected. This mathematical definition should be consistent with some of the intuitive concepts as well. Once the concept of interaction is defined, then, hopefully, more system-theoretic argument can lead to a means for determining the behavior of an aggregate of systems in terms of the behavior of the individual systems involved.

Wymore has already studied this problem for general systems (Ref. 24). However, the tools which he develops are very general, so as to be applicable to all kinds of systems. The possibility exists that for one specific class of systems, not all the power in Wymore's methods

would be needed. The goal of this dissertation is to study a particular class of systems, finite state systems (which will be defined in Chapter 2) and seek a simple set of methods for describing the overall behavior of an interconnected set of these systems.

1.3. Scope and Organization of the Investigation.

The process of deriving the overall behavior of a collection of interacting systems is very sensitive to the specific mathematical structure used in describing each system in the collection. For example, if each element in an electrical circuit is described by a differential or integral equation, the application of Kirchoff's laws allow one set of equations to be written which completely describes the overall behavior of the circuit. If each element is described by a graphical voltage vs. current relation, though, the derivation of the overall behavior becomes much more tedious. As a result, there is merit in studying alternative mathematical structures in representing various sytems. One form may be more useful in visually comprehending the ongoing processes in the system, but not exceedingly useful in studying interactions of the system with other systems. Thus, once a mathematical structure has been decided on, its usefulness in studying interactions must be investigated.

The work of this dissertation will pursue the same lines. In Chapter 2 a formal mathematical definition of a finite state system will be presented and several mathematical representations of the structure of this definition will be reviewed. A new mathematical representation, utilizing polynomial functions, will be introduced and considered in detail. Then Chapter 3 will introduce the concept of interconnections of finite state systems. Using the polynomial functions as a representation for finite state systems, methods will be developed to facilitate the analysis of an arbitrary interconnection of finite state systems. In addition, a method for synthesizing a finite state system will also be presented. The remainder of the work will consist of examples illustrating the computational techniques developed in Chapters 2 and 3, as well as an example illustrating the applicability of finite state systems, and the techniques herein derived, in a systems engineering problem.

The manner of development in Chapters 2 and 3 will be mathematical in scope. The familiar structure of Definition, Theorem, Proof will be used frequently. This is in keeping with the basically mathematical nature of system theory. In Chapter 4, though, the emphasis will be on informal discussion of examples and concepts to illustrate the precise ideas being propounded in the earlier chapters.

In a totally theoretical document no examples would be included. But the ultimate goal of system theory is to serve systems engineering and systems engineers. To this end, the simple examples of Chapter 4 are presented to illustrate the methods introduced in the theoretical development.

CHAPTER 2

DEVELOPMENT OF THE REAL NUMBER REPRESENTATION.

2.1. Mathematical Structure of a Finite State System.

The mathematical formulation of the structure of finite state systems has been the subject of much discussion. As a result many synonymous terms have arisen to label these different formulations. The terms "finite state system" and "sequential machine" are the most widely used terms and in this work will be used interchangeably. But no matter what the name or the formulation, two important algebraic structures can be seen underlying these formulations. These are the so-called Moore machine and Mealy machine. The following definition gives Moore's version of the algebraic structure of a finite state system (Hartmannis & Stearns, Ref. 14).

Definition 2.1. A finite state system is a quintuple:

$$M = (S, I, O, f, g)$$

which possesses the following properties:

S is a finite set not empty,

I is a finite set not empty,

O is a finite set,

$f: S \times I \rightarrow S$,

$g: S \rightarrow O$.

The only distinction between the Moore structure and the Mealy structure is that the Mealy structure defines g as a function: $g:S \times I \rightarrow O$. The most extensive treatment of the Mealy structure available in book form is the work by Gill (Ref. 9).

It should be noted in this definition that no interpretation is assumed concerning the "nature" of the various sets in the quintuple. This places in evidence even more strongly the essential algebraic properties inherent in the structure of a sequential machine. Interpretations of the meaning of the various sets in the functioning of a sequential machine are direct and given immediately below. It is preferable, however, to state these definitions without interpretation for the purpose of emphasizing the algebraic structure (and, pragmatically, to shorten the definition for the sake of conciseness).

The interpretations attached to the elements of the quintuple M are as follows:

(1.) S is the set of <u>states</u>. The concept of state is certainly the most important single idea in the field of system theory and also does service in such diverse fields as Markov processes and more general stochastic processes (Feller, Ref. 6). As a result of playing such a fundamental role, many attempts have been made at either defining the concept of state or describing the property that

the states of a system possess. The latter approach is more appropriate because, as will be seen below, the concept of state is really an undefined term.

Conventional discussions of the concept of state, whether from the point of view of cybernetics (Ashby, Ref. 1; Pask, Ref. 20), control theory (Derusso, Roy, Close, Ref. 4; Bellman, Ref. 3), or system theory (Wymore, Ref. 24) condense to the same basic thought: state variables are entities complete knowledge of which, along with knowledge of the form of the stimuli presented to the system, is sufficient to predict completely the behavior of the system (in terms of its state) in responding to the stimuli. As such, this concept is quite old and dates back to the work of Lagrange in classical mechanics problems. States are the particular values taken on by the state variables of a system. Sometimes the difference between state variables and states is desired for purposes of clarity (Bellman, Ref. 3) and other times the distinction can be ignored and the term "state" used in a blanketing fashion. The latter practice will prevail in this work.

It is important to note the requirement that S be finite and not empty in Definition 2.1. When S is empty the system is "trivial" (Gill, Ref. 9) and uninteresting. When S is not finite the system is not approachable by the finite techniques to be studied in this work.

(2.) I is the set of inputs. The concept of input is more easily defined than that of state. Intuitively, input is the stimuli that impinge on the system from the external world, and to which the system responds.

Note that I is required to be non-empty and finite. The view adopted herein is that all finite state systems should have input, even if the system ignores the input or does not respond to it. This view is held for the purposes of convenience, and not from any overriding theoretical considerations.

(3.) O is the set of outputs. Again the concept of output is easy to grasp on an intuitive basis. Outputs are the quantitative results of the system's responding to its inputs. Of course, they may be of any nature and widely removed from the system's actual structure, being related to the system only by an arbitrary function (see below). Also the output can be different for different observers, even though the system is the same (Wymore, Ref. 24).

Note that, unlike the sets S and I, the set O is not required to be non-empty. This is to acknowledge two system-theoretic notions: (a.) The outputs of a system are not unique. Output usually depends on the observer. (b.) With outputs being arbitrary, there is good argument for studying systems without any output at all, using the

identity mapping as an output function, and concentrating on the state behavior.

(4.) f is the state-transition function. The concept of state involves with it the notion of a change in internal configuration due to the reception of an input. This change is defined completely whenever knowing the present state and the present applied input allows the observer to predict completely the state of the system that results from this change. This can be cast in terms of a function f on the set S X I with a range in S.

(5.) g is the output function. The output, viewed as in the paragraph above, is a measure of the system's "response". It is some function on the state set S with range in O. Intuitively, g tells an observer what he, in particular, wants to know about the system. Someone else's choice of output might be different, depending on his point of view and the measuring equipment available to him.

It must be noted, of course, that several undefined terms have been used in the interpretations given above, such terms as stimuli, response, etc. Not much benefit would be gained from trying to make these concepts more precise than the intuitive usage above. A set of undefined terms must always be used in any mathematical structure and awareness of their imprecise nature should be explicit.

2.2. Mathematical Representations of the Algebraic Structure.

With the mathematical archetype of the finite state system established through its algebraic structure, the next step concerns the choice of specific representations for the sets and functions of Definition 2.1. This step embodies the specification of the sets S, I, and O in terms of the kinds of elements they may contain and, contingent on this first step in some cases, the form of the system functions f and g.

In the past most effort has been devoted to the second step, representing the system functions, by means of two different techniques. The oldest and most widely used choice for representing the system functions has been the use of "flow tables" and "state flow-graphs". In these techniques the sets S, I, and O are allowed to be any finite sets of arbitrary symbols. The system functions f and g are specified by tabular listings with these symbols. The function f is presented as a table listing the next state for all possible combinations of present states and inputs. The function g becomes a table listing the output for each state (or for each state and input combination in the case of a Mealy machine definition). Gill's book (Ref. 9) is wholly devoted to this treatment. A variation of this approach can also be seen in the book by Hartmannis

and Stearns (Ref. 14), and also in papers by Hartmannis (Ref. 12 and Ref. 13). In this treatment (which is based on digital equipment considerations) the inputs and outputs are usually chosen as 1 or 0, and the states chosen as integers. The flow-table is used as before.

A second technique, and one which has been widely treated in the literature recently, involves the assumption of specific properties about the system functions and the elements of the sets S, I, and O. The assumed properties which have received so much recent treatment is that the elements of S, I, and O are elements of a Galois Field, a finite field in which all arithmetic is performed on a modular basis with respect to a prime number, p. The state, input, and output to the system is given by a vector formed from elements of these sets. The equations describing the next state and output are linear in these vectors and are of the form:

$$\begin{aligned} \underset{\sim}{s}' &= A\underset{\sim}{s} + B\underset{\sim}{I} \\ \underset{\sim}{O} &= C\underset{\sim}{s} + D\underset{\sim}{I} \end{aligned} \qquad (1.)$$

where: A, B, C, D are matrices from the same Galois field modulus p, $\underset{\sim}{s}$ is the current state vector, $\underset{\sim}{s}'$ is the next-state vector, $\underset{\sim}{O}$ is the current output, and $\underset{\sim}{I}$ is the current input. Machines with these assumed properties have been discussed under several titles, usually some form of the name "linear modular sequential machines" (Friedland, Ref.

7, 8; Kautz, Ref. 17). Yau and Wang (Ref. 25) present a good statement and summary of sequential machines and also the linearity properties and assumptions in general. Linear sequential machines have been found to possess useful properties in the study of error-correcting codes. This has spurred their study as much as research on their system properties.

Finally, one more specific representation for the sets and functions of a sequential machine should be mentioned, particularly because it bears a remote resemblance to the form to be developed later in this chapter. Haring (Ref. 11) uses a formulation in his work requiring the state set to be composed of n-component unit vectors. Only one component of each vector is non-zero, and that one is unity. Each vector has a different non-zero component. The input set has the same form. The state-transition function has the form:

$$\underset{\sim}{s}' = \sum_{j=1}^{m} I_j(A_j)\underset{\sim}{s} \qquad (2.)$$

Since only one value of I_j is non-zero for any $\underset{\sim}{I}$, the input acts to select a particular matrix A_j and yield the next-state vector $\underset{\sim}{s}'$. An interesting aspect of this formulation is the use of products of matrices and vectors to compute the next state. Usage of another kind of vector and matrix product will be made in the mathematical formulation in this work, but in association with polynomial functions.

2.3. The Real Number Representation.

The discussion in section 2.2 of some of the representations chosen for the algebraic structure of the sequential machine has paved the way for the introduction of the real number representation. This will be done by informally discussing the assumptions to be made and their meaning or usefulness. Then a fully rigorous development, through definitions and theorems, will be undertaken.

For the sake of discussion, assume that S, I, and O are disjoint subsets of real numbers. Thus S consists of n distinct real numbers, I consists of m distinct real numbers, O contains r distinct real numbers. Now inquiry is made as to a possible representation for the system functions f and g. By definition, f maps S X I into S. But since it is assumed that S and I are made up of real number elements, a geometric interpretation of this function can be obtained by plotting S versus values of the pairs (s, i) where $s \in S$, $i \in I$.

Viewing the geometry of Fig. 1, the problem of determining the function f is the same as specifying a surface which passes through the proper points in a 3-dimensional space. Since finite numbers of elements are involved in all sets, this can be easily done by polynomial interpolation, as will be shown below. By a similar argument, the function g can be viewed as passing a curve

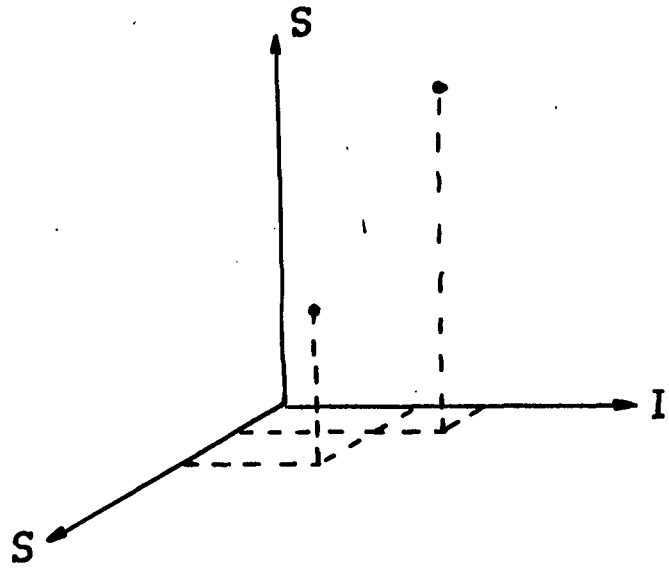


Fig. 1. Plot of S versus (S, I)

through a specified set of points, with S as the domain and O as the range. See Fig. 2.

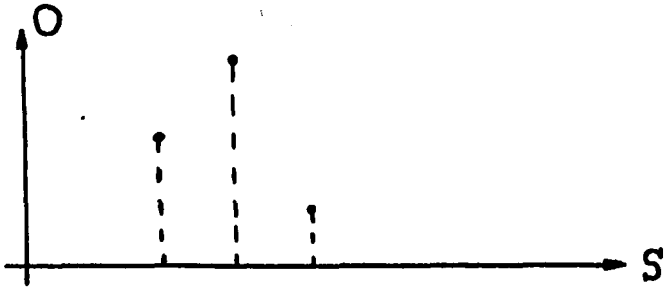


Fig. 2. Output versus States

The core of all the work to be done in the rest of this section rests on this simple geometric interpretation.

In the definition which follows a bit of notation will be used that bears explaining. If a function a is defined on a set A with values in a set B and is one-to-one and onto, then it is said that A is isomorphic (a) to the set B, and B is isomorphic (a^{-1}) to the set A. This is notation due to Wymore, who uses it in his book. In the following, all symbols for polynomial functions begin with a small letter "p". Thus, pf is a polynomial function. The set of the real numbers is R, and R^2 is the notation for R X R, and R^n is the symbol for R X R X ··· X R, n Cartesian products of R. The following is introduced as a definition (Wymore, Ref. 24):

Definition 2.2. If f is a function defined on the set A with values in B and C is a subset of A then the restriction to C of f is res(f,C) and defined:

$$\text{res}(f,C) = ((x,y): x \in C \text{ and } (x,y) \in f),$$

$$\text{i.e., } (\text{res}(f,C))(x) = f(x) \text{ for every } x \in C.$$

The necessary symbolism has been established to allow the statement of the following definition.

Definition 2.3. A real number representation of a finite state system M = (S, I, O, f, g) is a quintuple of sets M' = (S', I', O', f', g') such that there exists a set of isomophisms (α, β, γ) with the properties:

a.) S is isomorphic (α) to S', a subset of R,
I is isomorphic (β) to I', a subset of R,
O is isomorphic (γ) to O', a subset of R,

b.) $f' = res(pf', S' \times I')$, where pf' is a polynomial defined on R^2, such that:

$$\alpha(f(s,i)) = pf'(\alpha(s), \beta(i)), \qquad (3.)$$

c.) $g' = res(pg', S')$, where pg' is a polynomial defined on R, such that:

$$\gamma(g(s)) = pg'(\alpha(s)). \qquad (4.)$$

Definition 2.3 appears quite formidable at first. It can be considered a section at a time, though, and be easily understood. Condition (a.) requires that the sets S', I', O' be related to the sets of the original system by one-one mappings (this is important for the purposes of conditions (b.) and (c.)). In condition (b.) the function f' is defined as the restriction to S' X I' of a polynomial pf'. This polynomial must be such that f' satisfies equation (3.). The form of equation (3.) is that of an operation preserving mapping. Condition (c.) is much the same as (b.) The function g' is the restriction to S' of a polynomial pg' that also preserves operations.

Definition 2.4. A finite state system is a real number machine if it is a real number representation of itself.

The conditions of Definition 2.3 specify the requirements for a real number representation of a given sequential

machine. The desired end, though, is to represent any machine in a real number form, which leads to the question whether or not this is possible in any case. Also, if such a representation is always possible, then some procedure for finding the representation would also be desired. The following theorem provides the proof and also indicates a scheme for constructing a real number representation of the given machine.

Theorem 2.1. Let $M = (S, I, 0, f, g)$ be a finite state system. Then there exists a real number representation $M' = (S', I', 0', f', g')$ of the system M.

Proof. The proof will be made by constructing the system M'.

Let the number of elements in S, I, and 0 be, respectively, n, m, and r. Then define three one-to-one functions, α, β, γ that map S, I, 0 respectively into subsets of the reals. Thus:

$$\alpha: \quad S \rightarrowtail S' \qquad S' \subset R$$

$$\beta: \quad I \rightarrowtail I' \qquad I' \subset R$$

$$\gamma: \quad 0 \rightarrowtail 0' \qquad 0' \subset R$$

Condition (a.) of Definition 2.3 is seen to be satisfied by this construction.

Let $S = (s_1, s_2, \cdots, s_n)$, $I = (i_1, i_2, \cdots, i_m)$. Since $\alpha(s) \in R$ and $\beta(i) \in R$ for every $s \in S$, $i \in I$, then the following Lagrange interpolation polynomial may be formed:

$$pf'(\alpha(s),\ \beta(i)) = \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} \frac{(\alpha(s) - \alpha(s_0))..}{(\alpha(s_j)-\alpha(s_0))..}$$

$$\frac{..(\alpha(s) - \alpha(s_{j-1}))(\alpha(s) - \alpha(s_{j+1}))...(\alpha(s) - \alpha(s_{n-1}))}{..(\alpha(s_j)-\alpha(s_{j-1}))(\alpha(s_j)-\alpha(s_{j+1}))...(\alpha(s_j)-\alpha(s_{n-1}))} \quad X$$

$$\frac{(\beta(i) - \beta(i_0))...(\beta(i) - \beta(i_{k-1}))(\beta(i) - \beta(i_{k+1}))..}{(\beta(i_k)-\beta(i_0))...(\beta(i_k)-\beta(i_{k-1}))(\beta(i_k)-\beta(i_{k+1}))..}$$

$$\frac{..(\beta(i) - \beta(i_{m-1}))}{..(\beta(i_k)-\beta(i_{m-1}))} \left[\ \alpha(f(s_j,\ i_k))\ \right] \qquad (5.)$$

for every $s \in S$, $i \in I$.

Then, for every $s \in S$, $i \in I$:

$$pf'(\alpha(s),\ \beta(i)) = \alpha(f(s,\ i)).$$

By comparison with equation (3.) the condition (b.) of Definition 2.3 is satisfied.

Finally, form the Lagrange interpolation polynomial:

$$pg'(\alpha(s)) = \sum_{j=0}^{n-1} \frac{(\alpha(s) - \alpha(s_0))..}{(\alpha(s_j)-\alpha(s_0))..}$$

$$\frac{..(\alpha(s) - \alpha(s_{j-1}))(\alpha(s) - \alpha(s_{j+1}))..}{..(\alpha(s_j)-\alpha(s_{j-1}))(\alpha(s_j)-\alpha(s_{j+1}))..}$$

$$\frac{..(\alpha(s) - \alpha(s_{n-1}))}{..(\alpha(s_j)-\alpha(s_{n-1}))} \left[\ \alpha(g(s_j))\ \right] \qquad (6.)$$

Then for every $s \in S$,

$$pg'(\alpha(s)) = \gamma(g(s)).$$

Thus condition (c.) of Definition (2.3) is satisfied. So the construction of α, β, γ has produced the quintuple (S', I', O', f', g') that satisfies the required conditions. This quintuple is a real number representation M' of the machine M. The proof is completed.

Since any finite state system has a real number representation, it is natural to inquire about properties of a real number representation. The following theorem makes a fundamental assertion.

Theorem 2.2. Given a finite state system M whose real number representation is M', then M' is also a finite state system.

Proof. Let $M = (S, I, O, f, g)$ and $M' = (S', I', O', f', g')$. Since M' is a real number representation of M, then equations (3.) and (4.) are satisfied and:

$$f'(\alpha(s), \beta(i)) = \alpha(f(s, i)),$$

$$g'(\alpha(s)) = \gamma(g(s)).$$

But, by the definitions of α and β, $\alpha(s) \in S'$ and $\beta(i) \in I'$ for every $s \in S$ and $i \in I$. But, since M is a finite state system $f(s, i) \in S$ for every $s \in S$, $i \in I$. Thus, the function f' is such that f': $S' \times I' \rightarrow S'$.

By the definitions of α and γ, $\alpha(s) \in S'$ and $\gamma(o) \in O'$ for every $s \in S$ and $o \in O$. But, since M is a finite state system, $g(s) \in O$ for every $s \in S$. Thus, the function g' is such that g': $S' \rightarrow O'$.

Now this defines the quintuple (S', I', O', f', g') such that: S', I', O' are finite sets

f': $S' \times I' \rightarrow S'$

g': $S' \rightarrow O'$

By definition 2.1, M' is a finite state system. The proof is completed.

Theorem 2.1 shows how to construct the polynomials pf' and pg' for a real number representation, but in general they are very tedious constructions. The Lagrangian format is not very compact. But the use of standard interpolation theory allows the polynomial to be obtained in other ways. Notice first of all that the Lagrange interpolation polynomial used in equations (5.) and (6.) can be rewritten. By expanding each term in the sum and then collecting like powers of s and i the two polynomials can be written as follows: Let $\alpha(s) = s' \in S'$, $\beta(i) = i' \in I'$, then: $pf'(\alpha(s), \beta(i)) = pf'(s', i')$. Also $pg'(\alpha(s)) = pg'(s')$. Then after expansion and collecting:

$$pf'(s',i') = \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk}(s')^j(i')^k \qquad (7.)$$

$$pg'(s') = \sum_{j=0}^{n-1} b_j(s')^j \qquad (8.)$$

Now a means of finding the coefficients a_{jk} and b_j is desired. Since the desired polynomials are polynomials that pass through given points (see figures 1 and 2), the

methods applicable to finding a polynomial which passes through certain points can be used directly, and the same theorems on uniqueness and existence of the solutions will also be applicable. The results are standard in algebraic studies of interpolation (Hoffman and Kunze, Ref. 15). The proofs of the following two pertinent theorems will not be given. Instead in section 2.4 a more general theorem will be proved which encompasses these two.

Theorem 2.3. The coefficients (b_j) of the polynomial $y = \sum_{j=0}^{n-1} b_j x^j$ which takes on the values $(y_1, y_2, \cdots, y_n)$ at the points $(x_1, x_2, \cdots, x_n)$, all distinct, can always be uniquely found as solutions of the n simultaneous equations:

$$\begin{aligned}
b_0 + b_1 x_1 + b_2 x_1^2 + \cdots + b_{n-1} x_1^{n-1} &= y_1 \\
b_0 + b_1 x_2 + b_2 x_2^2 + \cdots + b_{n-1} x_2^{n-1} &= y_2 \\
\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots & \\
b_0 + b_1 x_n + b_2 x_n^2 + \cdots + b_{n-1} x_n^{n-1} &= y_n,
\end{aligned} \tag{9.}$$

i.e., the solution exists and is unique.

Equations (9.) can also be written as:

$$\begin{aligned}
\sum_{j=0}^{n-1} b_j x_1^j &= y_1 \\
&\vdots \\
\sum_{j=0}^{n-1} b_j x_n^j &= y_n
\end{aligned} \tag{10.}$$

Theorem 2.4. The coefficients (a_{jk}) of the polynomial $z = \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x^j y^k$ which takes on the values $(z_{11}, z_{12}, \ldots z_{1m}, z_{21}, z_{22}, \ldots, z_{2m}, \ldots, z_{n1}, \ldots, z_{nm})$ at the points $((x_1, y_1), \ldots, (x_1, y_m), (x_2, y_1), \ldots, (x_2, y_m), \ldots, (x_n, y_1), \ldots, (x_n, y_m))$, all distinct, can be uniquely found as solutions of nm simultaneous equations:

$$\begin{aligned}
&\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x_1^j y_1^k = z_{11} \\
&\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x_1^j y_2^k = z_{12} \\
&\quad \vdots \qquad\qquad \vdots \\
&\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x_1^j y_m^k = z_{1m} \\
&\quad \vdots \qquad\qquad \vdots \\
&\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x_n^j y_1^k = z_{n1} \\
&\quad \vdots \qquad\qquad \vdots \\
&\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk} x_n^j y_m^k = z_{nm},
\end{aligned} \tag{11.}$$

i.e., the solution exists and is unique.

With Theorems 2.3 and 2.4, the following theorem demonstrates the computational technique for finding real

number representations of finite state systems, and at the same time demonstrates a uniqueness property of the real number representation.

Theorem 2.5. Given a finite state system M = (S, I, 0, f, g) then for fixed values of the real numbers in the sets S', I', 0' which are isomorphic (α, β, γ) to S, I, 0, respectively, there exists one unique real number representation of M.

Proof. By hypothesis, the sets S', I', 0' are fixed in advance and are isomorphic to S, I, 0. By Theorem 2.1 there exists a real number representation for M using S', I', 0', since there is no restriction in the theorem if the sets S', I', 0' are chosen in advance. Let this representation be M' = (S', I', 0', f', g'). Also the functions f', g' are given by the restrictions of the polynomials of equations (5.) and (6.) to S' X I' and S'.

By taking equation (6.), expanding all the products, collecting like powers of $\alpha(s)$, it may be written as:

$$pg'(\alpha(s)) = \sum_{j=0}^{n-1} b_j(\alpha(s))^j \qquad (12.)$$

Where the b_j are unknown coefficients. However, as was shown in Theorem 2.1, it must be true for any $s \in S$ that:

$$pg'(\alpha(s)) = \gamma(g(s)) \qquad (13.)$$

This property is still true for equation (12.), which is the same equation (6.) after rearranging. Thus it must be true that for any $s_k \in S$, that

$$\sum_{j=0}^{n-1} b_j(\alpha(s_k))^j = \gamma(g(s_k)) \quad k = 1, 2, \cdots, n \qquad (14.)$$

But this is in the form of equations (10.) and Theorem 2.3 applies. Hence the system (14.) may be solved for the coefficients b_j, and they will be unique.

By a similar argument, it was proved in Theorem 2.1 that:

$$pf'(\alpha(s),\beta(i)) = \alpha(f(s,i)). \qquad (15.)$$

If equation (5.) is now expanded of all indicated products and the like powers of $\alpha(s)$ and $\beta(i)$ are collected then the result is:

$$pf'(\alpha(s),\beta(i)) = \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk}(\alpha(s))^j(\beta(i))^k \qquad (16.)$$

But the property expressed in equation (15.) must still be valid for any $s_r \in S$ and $i_p \in I$, because (16.) is only a rearrangement of equation (6.). Thus it must be true that for any $s_r \in S$ and $i_p \in I$, that

$$\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk}(\alpha(s_r))^j(\beta(i_p))^k = \alpha(f(s_r,i_p)) \quad \begin{matrix} r = 1,2,\cdots, n \\ p = 1,2,\cdots, m \end{matrix} \qquad (17.)$$

But this is in the form of equations (11.) and Theorem 2.4 applies. Hence the system (17.) may be solved for unique coefficients a_{jk}.

The sets S', I', 0' are fixed, and the functions f' and g' which may be obtained from the restrictions to S' X I' and S' of the polynomials of equations (14.) and (17.) are unique, by Theorems 2.3 and 2.4. Hence the system M' = (S', I', 0', f', g') is a unique real number representation for M. The proof is complete.

The importance of this theorem lies in the computational procedure it exhibits for finding a real number representation of a given finite state system. The procedure may be summarized as follows:

(1.) Given a finite state system M = (S, I, 0, f, g), choose sets of real numbers S', I', 0' isomorphic (α, β, γ) to S, I, 0. This choice may be arbitrary or motivated by considerations in a particular model.

(2.) For every $s_r \in S$ and $i_p \in I$ form the system of nm simultaneous linear equations in the nm unknowns a_{jk}:

$$\sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk}(\alpha(s_r))^j(\beta(i_p))^k = \alpha(f(s_r,i_p)) \quad \begin{matrix} r = 1,2,\cdots, n \\ p = 1,2,\cdots, m \end{matrix} \qquad (17.)$$

This system is uniquely solvable for the a_{jk}.

(3.) For every $s_r \in S$ form the system of n simultaneous linear equations in the n unknowns b_j:

$$\sum_{j=0}^{n-1} b_j(\alpha(s_r))^j = \gamma(g(s_r)) \text{ for } r = 1, 2, \cdots, n. \quad (14.)$$

This system is uniquely solvable for b_j.

(4.) The polynomials

$$pf' = \sum_{k=0}^{m-1} \sum_{j=0}^{n-1} a_{jk}(\alpha(s))^j(\beta(i))^k,$$

and

$$pg' = \sum_{j=0}^{n-1} b_j(\alpha(s))^j,$$

are used to form the restrictions:

$$f' = res(pf', S' \times I')$$

$$g' = res(pg', S')$$

(5.) The quintuple $M' = (S', I', 0', f', g')$ is the unique real number representation of M corresponding to the isomorphism (α, β, γ) between S', I', 0' and S, I, 0.

With Theorem 2.5 the real number representation has been brought down to the level necessary for carrying out computations. Further, it demonstrates that it is not necessary to use the unwieldy Lagrange polynomials of equations (5.) and (6.) to construct a real number representation. A simple procedure, the solution of linear algebraic equations, is all that is required. Further the unique solution is always guaranteed.

Concerning the polynomials, an observation should be made about the distinction between f' and g', in a real number representation, and the polynomials pf' and pg'. It is clear from Definition 2.2 that f' and g' are functions, just as pf' and pg' are functions. But the use of the concept of restriction eliminates the need for arguing about what the transition function and output function "do" for values not in S' X I' or S'. As far as the system M' is concerned the question is meaningless; f' and g' have no values in the domain other than S' X I' and S'. The polynomials pf' and pg' do have meaning for other values, of course, but they are not the functions used in the machines. The polynomials are functions which "generate", so to speak, the desired functions f' and g' in the real number representation. When working with a real number represesentation in practice the polynomials themselves will be dealt with, but it must be realized that the values of pf' and pg' outside S' X I' and S', respectively, may not have any system theoretic meaning.

To close this section, a final word should be said on a point that reflects the different approaches that could have been chosen in the development. It may be observed in Theorems 2.1 and 2.5 that even though the elements in the sets S', I', O' were chosen as any real numbers, there is no specific reason, in the proof of the

theorem, to consider such a broad class for the elements of these sets. For example a special subset of the reals could have been chosen, such as the integers. The functions α, β, γ could then map each set S, I, 0 into a sequence of integers for the sets S', I', 0', respectively. Since the images of α, β, γ would be integers then the equations (5.) and (6.) would be simpler to write, the elements $\alpha(s)$, $\beta(i)$, $\gamma(o)$ being replaced by an appropriate integer. This is a valid argument, and development of the theorems could be made on this basis. Later development, however, may encounter a situation when the usage of actual real numbers, and not integers, may be advantageous. To provide for such a circumstance the theorems have been developed on the basis of arbitrary real numbers. When integers are of specific advantage, such as the examples in Chapter 4, they will be used. But to give the theory a scope broad enough for all imaginable problems, the reals were chosen. The extra complication of the theorems, in comparison to the gain in generality thus obtained, is slight. Had the choice of reals introduced any more difficulties in the proofs than already shown, the pragmatism of a simpler structure versus the elegance in a more general structure would have to be resolved in favor of either integers or reals. In this case the choice was not of such consequence or scope.

2.4. Multiple Input Systems.

From definition 2.1 a finite state system is a quintuple $M = (S, I, O, f, g)$. No restriction was made in that definition as to the structure of the state, input, or output sets. Thus the sets S, I, or O could be Cartesian products of several sets. This proves to be a useful special case for the description of interconnected systems. If the input set I is a Cartesian product of several other sets, each input is a vector quantity. Intuitively, this describes a system with several input ports. Similarly, if the state set S is a Cartesian product, each state is a vector quantity. If a collection of systems was interconnected and one system was desired that behaved the same way as the interconnection, intuitively, again, a vector state would be the simplest representation of the state transitions in the single system. Each component of the vector state would be a state from a system in the interconnected collection. These concepts will be discussed in depth in Chapter 3. They serve as motiviation for the following definition:

Definition 2.5. A multiport finite state system is a finite state system $M = (S, I, O, \underline{f}, \underline{g})$ which satisfies the following:

$S = S_1 \times S_2 \times \cdots \times S_n$, where each S_r is a finite set, not empty,

$I = I_1 \times I_2 \times \cdots \times I_m$, where each I_p is a finite set, not empty,

$0 = S$,

$\underset{\sim}{g} = \underset{\sim}{w}$, where w is the identity mapping.

A multiport finite state system is allowed to have vector valued inputs and states but a rather trivial output structure, i.e., the output is the current state. This seems like an unrealistic restriction at first, but is actually an important simplification. The simplified output structure will be discussed in Chapter 3 and justified in the context of interconnected systems. Note that $\underset{\sim}{g}$ and $\underset{\sim}{w}$ are identified as vector quantities by the "$\sim$" underneath their symbols. This common convention will be employed throughout the remainder of this work.

Definition 2.6. A multiport real number representation of a multiport finite state system $M = (S, I, S, \underset{\sim}{f}, \underset{\sim}{w})$ is a quintuple $M' = (S', I', S', \underset{\sim}{f}', \underset{\sim}{w}')$ such that there exists isomorphisms $\underset{\sim}{\alpha}$ and $\underset{\sim}{\beta}$ with the properties that:

S is isomorphic ($\underset{\sim}{\alpha}$) to $S' \subset R^n$, i.e., $S' = S_1' \times S_2' \times \cdots \times S_n'$, where for each $r = 1, 2, \cdots, n$, S_r is isomorphic (α_r) to $S_r' \subset R$,

I is isomorphic ($\underset{\sim}{\beta}$) to $I' \subset R^m$, i.e., $I' = I_1' \times I_2' \times \cdots \times I_m'$, where for each $p = 1, 2, \cdots, m$, I_p is isomorphic (β_p) to $I_p' \subset R$,

$\underset{\sim}{f}' = \text{res}(p\underset{\sim}{f}', S' \times I')$, where pf' is a polynomial defined on R^{m+n} such that

$\underset{\sim}{\alpha}(\underset{\sim}{f}(\underset{\sim}{s}, \underset{\sim}{i})) = p\underset{\sim}{f}'(\underset{\sim}{\alpha}(\underset{\sim}{s}), \underset{\sim}{\beta}(\underset{\sim}{i}))$, for every $\underset{\sim}{s} \in S$ and $\underset{\sim}{i} \in I$,

$\underset{\sim}{w}' = \text{res}(p\underset{\sim}{w}', S')$, where $p\underset{\sim}{w}(\underset{\sim}{s}') = \underset{\sim}{s}'$, for every $\underset{\sim}{s}' \in S'$, is the identity function.

Note that in the quintuple M (or M') of this definition the sets S and O being identical is indicated by repeating S in the third position of the quintuple. Since sets are being dealt with, the temptation exists to eliminate the repetition of S and use a four-tuple, thus trusting to memory the identical nature of the sets. This is not done above for the sake of consistency with the initial definition of a finite state system as a quintuple. Similarly $\underset{\sim}{w}$ is understood to always be the identity function for multiport systems. Again, the $\underset{\sim}{w}$ will not be eliminated. The sacrifice of occasionally writing down redundant information will be made in order to retain a consistent quintuple structure for finite state systems.

Pursuing the same path as in the previous section, it is now desired to show that every multiport system has a real number representation and then exhibit a scheme for constructing that representation. This could be done in a fashion analogous to Theorems 2.1 and 2.3. But the Lagrange polynomials of Theorem 2.1 become extremely

difficult to write for the systems with more states and inputs. But drawing on the experience gained in the previous section with polynomial functions for representation of state transitions allows a simpler proof to be presented for the method of deriving the coefficients of the polynomial pf'.

To aid in the notation, the following convention is adopted after Wymore. If $\underset{\sim}{y}$ is a vector, then $\pi_j(\underset{\sim}{y})$ is understood to be the j^{th} component of $\underset{\sim}{y}$. Thus $\pi_j(\underset{\sim}{y}) = y_j$. With this notation the following theorem can be proved:

Theorem 2.6. Let $M = (S, I, S, \underset{\sim}{f}, \underset{\sim}{w})$ be a multiport finite state system and let $S = S_1 \times S_2 \times \cdots \times S_n$ and $I = I_1 \times I_2 \times \cdots \times I_m$. Let $S'_r \subset R$, $r = 1, 2, \cdots, n$, and $I'_p \subset R$, $p = 1, 2, \cdots, m$, be such that S_r isomorphic (α_r) to S'_r, $r = 1, 2, \cdots, n$, and I_p isomorphic (β_p) to I'_p, $p = 1, 2, \cdots, m$. Then there exists a multiport real number representation of M.

Proof. The proof will be made by constructing the real number representation.

By hypothesis, the isomorphisms α_r, $r = 1,2,\cdots, n$, and β_p, $p = 1, 2, \cdots, m$, form the two isomorphisms:

$$\underset{\sim}{\alpha}: \quad S \rightarrow S' \subset R^n,$$

$$\underset{\sim}{\beta}: \quad I \rightarrow I' \subset R^m.$$

These isomorphisms qualify for the isomorphism requirements of Definition 2.6. It is now necessary to construct the proper polynomials.

From the arguments given in the previous section it is apparent that a Lagrange interpolation polynomial could be constructed for each component of the vector-valued function $p\underset{\sim}{f}'$. Once the polynomials were constructed, each component could be represented, after multiplication of the Lagrange products and collecting like variables, in the form:

$$\pi_t \left[p\underset{\sim}{f}'(\underset{\sim}{\alpha}(\underset{\sim}{s}), \underset{\sim}{\beta}(\underset{\sim}{i})) \right] =$$

$$\sum_{q=0}^{m_m-1} \cdots \sum_{b=0}^{m_1-1} \sum_{k=0}^{n_n-1} \cdots \sum_{j=0}^{n_1-1} a^t_{j\ldots kb\ldots q} (\alpha_1(s_1))^j \cdot\cdot$$

$$\cdot\cdot(\alpha_n(s_n))^k (\beta_1(i_1))^b \cdots (\beta_m(i_m))^q, \qquad (18.)$$

where (18.) is true for every $i_p \in I_p$, $p = 1, 2, \cdots, m$ for every $s_r \in S_r$, $r = 1, 2, \cdots, n$, and for every $t=1,2,\cdots,n$. But from Definition 2.6 it is required that:

$$\pi_t \left[p\underset{\sim}{f}'(\underset{\sim}{\alpha}(\underset{\sim}{s}), \underset{\sim}{\beta}(\underset{\sim}{i})) \right] = \pi_t \left[\underset{\sim}{\alpha}(\underset{\sim}{f}(\underset{\sim}{s}, \underset{\sim}{i})) \right]. \qquad (19.)$$

Examination of equations (19.) and (18.) shows they both have the same left-hand-side, and hence their right-hand-sides are equal. Thus:

$$\sum_{q=0}^{m_n-1} \cdots \sum_{b=0}^{m_1-1} \sum_{k=0}^{n_n-1} \cdots \sum_{j=0}^{n_1-1} a^t_{j\ldots kb\ldots q} (\alpha_1(s_1))^j \cdot\cdot$$

$$\cdot\cdot(\alpha_n(s_n))^k (\beta_1(i_1))^b \ldots (\beta_m(i_m))^q =$$

$$\pi_t \left[\underset{\sim}{\alpha}(\underset{\sim}{f}(s_1, \cdots, s_n, i_1, \cdots, i_m)) \right], \qquad (20.)$$

which is true for every $s_r \varepsilon S_r$, $r = 1, 2, \cdots, n$, for every $i_p \varepsilon I_p$, $p = 1, 2, \cdots, m$, and for every $t = 1, 2, \cdots, n$. By the substitution of every $s_r \varepsilon S_r$, $r = 1, 2, \cdots, n$, and every $i_p \varepsilon I_p$, $p = 1, 2, \cdots, m$, into (20.) a set of $n_1 \cdots n_n m_1 \cdots m_m$ equations in the $n_1 \cdots n_n m_1 \cdots m_n$ unknowns

$a^t_{j\ldots kb\ldots q}$ is generated, since $\underset{\sim}{\alpha}$, $\underset{\sim}{\beta}$, and $\underset{\sim}{f}$ are known. If a solution exists to these equations, the proof is complete.

The elements of the coefficient matrix of the unknowns are given by: $(\alpha_1(s_1))^j \cdots (\alpha_n(s_n))^k (\beta_1(i_1)^b \cdots (\beta_m(i_m))^p$. From the manner in which the equations are generated (direct substitution), all elements in the <u>same row</u> correspond to only one value of $s_r \varepsilon S_r$, $r = 1,2,\cdots,n$, and $i_p \varepsilon I_p$, $p = 1, 2, \cdots, m$. The unknowns can also be arbitrarily ordered and, by ordering the coefficients can be brought into an order so that all elements in the <u>same column</u> have the same values of $j, \cdots, k, b, \cdots, p$ in the powers of $\alpha_1(s_1), \cdots, \alpha_n(s_n), \beta_1(i_1), \cdots, \beta_m(i_m)$.

Now examine whether the columns and rows of this matrix are linearly independent. If the columns were linearly dependent, there would exist a sequence of elementary linear operations on the columns which would make every element in at least two columns of the matrix identical. This is impossible, because any two columns differ in the same two positions by differing <u>powers</u> of $\alpha_1(s_1)$, $\cdots$, $\alpha_n(s_n)$, $\beta_1(i_1)$, $\cdots$, $\beta_m(i_m)$. No sequence of linear operations can reduce this difference of powers. Hence, the columns are linearly independent.

If the rows of the matrix were linearly dependent, there would exist another sequence of linear operations that would reduce two rows to be identical. But any two rows differ in the same column position by different powers of $\alpha_1(s_1)$, $\cdots$, $\alpha_n(s_n)$, $\beta_1(i_1)$, $\cdots$, $\beta_m(i_m)$. The two rows themselves differ by different values of $s_1 \in S_1$, $\cdots$, $s_n \in S_n$, $I_1 \in I_1$, $\cdots$, $i_m \in I_m$. Linear operations consist of addition of rows and multiplication of rows by scalars. Any attempt to reduce two elements in the same two column positions of two rows to identical elements will not produce identical column elements in all the other column positions, because the other column positions differ by powers of the variables from the two elements being operated on. This argument fails only if two points $(s_1, \cdots, s_n, i_1, \cdots, i_m)$ and $(s_1', \cdots, s_n', i_1', \cdots, i_m')$ are the

same. This can't happen, since only differing combinations are being generated in the substitution. The rows are linearly independent.

The matrix is invertible, and hence there does exist a solution for the unknown $a^t_{j\ldots kb\ldots p}$. The complete solution for t = 1, 2, $\cdots$, n completely specifies a polynomial pf' whose restriction to S' X I' satisfies Definition 2.6. The proof is complete.

This theorem thus establishes a computational procedure for multiport systems that has the same form as the simple procedure for single port systems established in the previous section. Now the usage of multiport systems in studying the interconnections of finite state systems will be undertaken in the next chapter.

CHAPTER 3

THE INTERCONNECTION OF FINITE STATE SYSTEMS

3.1. A Brief Review of Some Interconnection Schemes.

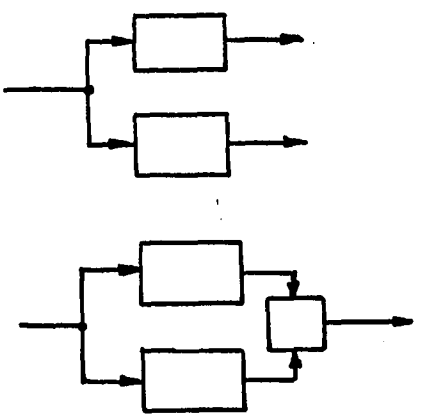
With the real number representation now developed, in a preliminary fashion at least, in the previous chapter, this chapter will be devoted to the second major task of analysis -- the consideration of interconnections.

Little work has been done on the analysis problem for interconnection of finite state systems per se (although much has been done for aspects of synthesis). The first work to appear on this specific topic was a paper in 1958 by Simon (Ref. 21). In that paper Simon chose the tabular form as the basic representation for a finite state system. The tabular form was described briefly in Chapter 2. Simon viewed these tables as matrices and gave them the name of the "operation matrices" of a system. Simon then discussed three different kinds of interconnections of finite state systems and described a method for deriving the "operation matrices" of interconnected systems from the matrices of the components.

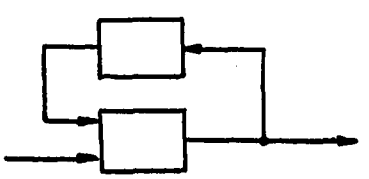
The three basic interconnections discussed by Simon were cascade, parallel, and feedback. In a cascade system,

the output of one system becomes the input to another system. In parallel systems, both systems receive the input. The outputs may then be either combined together by a trivial (one-state) system or allowed to proceed independently. In feedback systems the output of one machine is used as an input to itself. The feedback output can be modified by passage through another system. See Fig. 3 below.

Cascade Systems



Parallel Systems



Feedback Systems

Fig. 3. Interconnections of Finite State Systems.

The next research to deal with the interconnection of finite state systems as an analysis problem appeared in 1961, in a paper by Gill (Ref. 10). In this work Gill employed a totally different representation for finite state systems than did Simon, describing the systems by what he referred to as "transition matrices", which were quite different from the tables called "operation matrices" by Simon. Gill's paper was concerned only with cascaded finite state systems, though, except for a short reference to a feedback connection as a special kind of cascade connection. About one-half of the paper was devoted to problems in cascade synthesis, the specification of cascade systems to perform the same operations as a given system.

Neither of these two papers took what would be considered a system-theoretic approach. No attempt was made to define what precisely was meant by an interconnection, other than drawing sketches to illustrate a particular kind of interconnection, such as cascade or parallel. Neither paper defined more general interconnections, either, i.e., interconnections which were not simple cascade, parallel, or feedback interconnections.

Almost exclusively since 1961, the thrust of research involving interconnections of finite state systems has been in response to concern over synthesis problems, and has received most attention from researchers studying

better methods for the synthesis of digital and other sequential systems. Principally, two problems have been treated: the resolution of a given finite state system into an interconnection of simpler systems, and problems involving the implementation of finite state systems by sequential circuitry. Of course, the resolution of a system into simpler interconnected systems certainly must consider and define certain aspects of interconnection, and these interconnections have been treated as a part of the effort necessary to preface the resolution theory. See, for example, the work of Yoeli (Ref. 26 and Ref. 27). However, the concentration of research in these areas as part of synthesis problems has had two effects. First, many approaches reflect the practical aspects of the synthesis of sequential circuits. For example in the book by Hartmannis and Stearns (Ref. 14) most of the examples have outputs limited to the symbols "1" and "0", the logical outputs in a digital or sequential circuit. Second, the methods involved in synthesis have been developed from sophisticated algebraic treatments, hence they work quite well for a general system specified in its tabular form, the transition table. So, not a great deal of effort in exploring other schemes of representation for a finite state system has been necessary.

It is curious that the interconnection of finite state systems has not been taken up as an analysis problem exclusively since Simon's initial paper in 1958. Besides the impact of synthesis on the field, this may be also due partially to the results of his paper. The technique developed in that paper showed that, basically, the whole of analysis for finite state systems could be boiled down to brute-force enumeration of all possibilities. This is not very exciting and also a task that is ideally suited for the digital computer. The usage of the real number representation for a machine will reflect a sizable "streamlining" in methods as well as rigorously fixing the concepts involved in a suitable framework.

3.2. The Role of Outputs in System Interconnections.

Before treating the topic of interconnections in a rigorous fashion through the use of the real number representation a closer discussion of system structure is desirable, specifically, the affect that outputs of a system can have on system interconnections. As was developed in Chapter 2, the structure of a finite state system is embodied in three sets and two functions. The next-state function f(s, i) and the output function are independent entities. This, in turn, suggests the following interpretation of a finite state system: Given a finite state system, A, in a

"box" with input i and output o, the internal machinery in the box can be separated into two parts. One box f(s,i) is a device that determines the next state, and another box g(s) determines the current output on the basis of the current state. See Fig. 4 below.

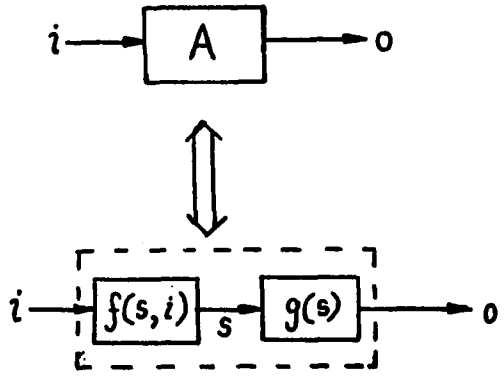


Fig. 4. Subdivision of a Finite State System.

In Fig. 4 the total device enclosed by the dotted lines is the system A, viewed as a pair of input-output terminals, broken down into the two component functions f(s,i) and g(s).

When a system is viewed in this manner an interesting property is made explicit. The output is a rather arbitrary definition on the set of states of the system. The real "heart" of the finite state system is the function f(s,i) that determines the manner in which states change in response to inputs. This property is true for systems of all natures and is strongly emphasized in Wymore's book. To quote Wymore (Ref. 24):

> The viewpoint held here then, is that output is not intrinsic to the system but depends on the observer. This viewpoint is consistent with experience: an engineer sometimes uses the reading on a voltmeter connected across a component or an oscilloscope display as the output of the circuit or component, a psychologist uses his electroencephalogram recordings as the brain's output, an accountant uses the net financial position as the output of the corporation. In all these cases, the so-called "output" is dependent on the observer and the instrumentation available to him.

The common sense power and appeal of this observation should not be allowed to bely its fundamental significance, especially when the question of interconnection of systems arises. In the classical engineering fields, output functions usually are generated by an identity mapping on the set of states. Only seldom are mappings of a more complicated nature used. Thus, an electrical engineer usually chooses a directly measurable variable, such as voltage or current, to coincide with the system state variables and concurrently characterize the system's behavior as an output. The usage of conventional instruments, such as voltmeters and ammeters, can then be employed to read the "output" directly to the engineer. The same could be said of many problems in other fields of engineering, such as mechanical or chemical engineering. When the system state and system output are chosen to be the same, interconnection of the output of one system into the input of another system becomes simple. It is usually no different than connecting wires together, or pipes, or linkages.

No complicated machinery is required as an intermediate step to produce the output as a computation on the state set.

The point being made here is not that the traditional fields of engineering are cowardly in choosing simple output structures. The nature of analysis and synthesis in these areas is such that the relation of system state to an output usually has some simple natural basis. Thus a voltage or a current is a natural variable to use as an output from most electrical systems. In fact more general schemes for choosing outputs may constitute an undue complication of the analysis. This is because the mathematics utilized has some relation to a physical system and the modelling process which led to the mathematics involved was chosen for simplicity, not for completeness through the inclusion of more complex outputs. Kirchoff's laws are a valid mathematical tool in a circuit problem, so the choice of outputs to conform with circuit variables, such as voltages or currents, is wise from the overall viewpoint of a simple system analysis. To take a circuit problem and then define complicated output functions on some or all of the state variables, and then use these as input functions to another circuit would be burdensome for analysis' sake, and would be avoided by the practicing engineer if he has a chance to avoid it.

The question which must now be confronted is the following: Is it legitimate to consider the interconnection of finite state systems based on the simplest possible output, an identity mapping? The answer is yes, and for the following two reasons. First, if a systems engineer is faced with deriving models for individual components, there is nothing to prevent him from specifying a state set which carries as much of the structure of an output set as may be needed, i.e., left to his own devices, an engineer can construct models that make the identity mapping a natural output, just as the identity function is a natural output for other engineering systems. Second, even if he cannot do this he can modify the input structure of the system to which an output is applied. Thus in figure 5 the system B can be changed to a system B', where the allowed input set of B' has been changed to be the state set of A. The system B' responds to the state of A' just as B responded to the output of A.

Thus, in all the following work it will be assumed that $O = S$ and $g = w$. This will be an important simplification. It also is now clear why multiport systems were defined in Chapter 2 to have the structure of identity outputs. When interconnections are considered, the identity output is both simpler and reasonable.

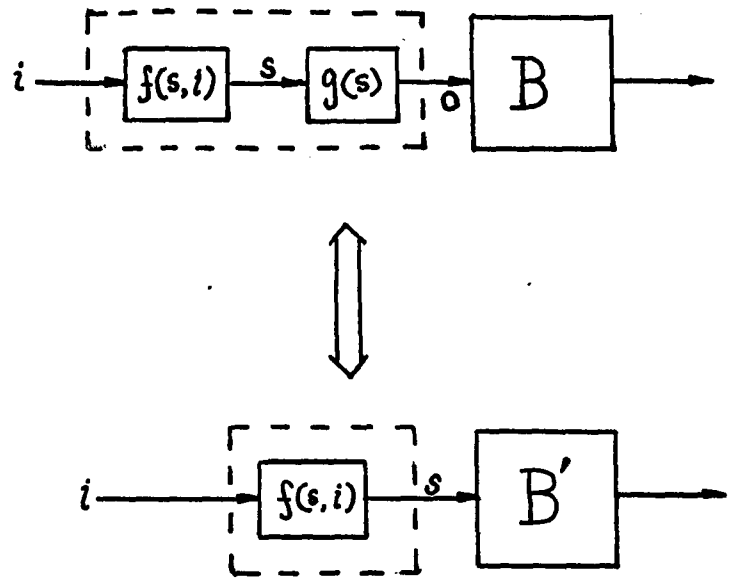


Fig. 5. Simplifying Outputs.

3.3. Describing the Next-State Function of Interconnections.

When two systems are interconnected in some arbitrary fashion, the state behavior of each individual system must be included in any description of the state behavior of the interconnection. Such a statement is true for all systems, but in the particular case of finite state systems this is usually recognized by defining the state set of the interconnection to be the Cartesian product of the state sets of the individual systems. Thus if S_1 is the state set of system M_1 and S_2 is the state set of system M_2, then $S_1 \times S_2$ is the state set of any interconnection of the two systems (the formal concept of an interconnection is

not defined yet, but the intuitive level of discussion is sufficient for this short section). For more discussion see Wymore (Ref. 24) and the book by Hartmannis and Stearns (Ref. 14).

The state set of an interconnection being described by a Cartesian product means that the state of an interconnection is a vector-valued quantity. The attempt to describe the next-state function of the interconnection must utilize vector-valued functions, as a result. In this light, the definition of a multiport system to be valid for vector-valued states and vector-valued next-state functions is logical. To put it on an intuitive level: the process of interconnecting finite state systems can be described. Several multiport finite state systems, with scalar state sets, are arranged in some interconnection. The state set of the interconnection is a vector-valued quantity defined as the Cartesian product of each state set in the individual systems. The problem of analysis is to find one system, with this vector-valued state set, which behaves in the same fashion as the interconnection.

All preliminaries have now been completed. The earlier studies of interconnection schemes have been briefly reviewed, and a new convention for output structures of interconnected systems has been adopted in the previous sections. This section has described the

interconnection problem in an intuitive way. With this foundation, plus the work of Chapter 2, the interconnection of finite state systems will be analyzed rigorously in the next section.

3.4. The Interconnection of Finite State Systems.

The first problem that manifests itself in discussing interconnections of finite state systems is the problem of notation. Given several multiport systems there is a very large number of distinct possible interconnections. What is desired is a compact notation for describing in as explicit a manner as possible just which specific interconnection is to be analyzed out of the set of possible interconnections.

This problem has already been studied, in general, by Wymore for general systems. In his book, A Mathematical Theory of Systems Engineering, Wymore develops notation that is sufficiently powerful to be extended to infinite numbers of interconnections of infinite numbers of arbitrary systems. This work is concerned only with the interconnection of a finite number of finite state systems, hence the complete power of Wymore's method is not needed. A different notation will be developed and employed instead. Particularly, the notation will be consistent with the symbolism already established for finite state systems and real number representations.

First, the concept of input terminals must be established. The multiport finite state systems has been treated in section 2.4 as possessing a Cartesian product structure on the input and/or state set. Intuitively, each possible input set in the Cartesian product is a different "terminal" into which inputs may be sent. This concept can be formalized by the following definition:

Definition 3.1. Given a multiport finite state system with the input set $I = I_1 \times I_2 \times \cdots \times I_m$, an input terminal is any one of the sets I_r, $r = 1, 2, \cdots, m$. The set of input terminals of a multiport finite state system is $U = (I_1, I_2, \cdots, I_m)$. An input terminal for a real number representation is defined as any of the sets I'_r, $r = 1, 2, \cdots, m$, where $I' = I'_1 \times I'_2 \times \cdots \times I'_m$ is the input set of the real number representation. The set $U' = (I'_1, I'_2, \cdots, I'_m)$ is set of input terminals of the representation.

Now consider what happens when several systems are interconnected. On an intuitive basis (an intuition derived from comparable processes in other disciplines, such as electrical engineering) the output is fed from one system to the input of another. Since identity outputs are being used, this means that the system structure is arranged so that the current state which a particular system is in becomes the current input symbol applied to an

input terminal of some other system or systems. Of course, no interconnection is allowed unless the range of an output (the state) from one system is the same set or a subset of the input set at the input terminal to which it is connected. Also it is expected, after all the interconnections are completed, that there be some input terminals which are "left over", i.e., some input terminals are not receiving input from other systems in the interconnection, but are free instead to receive inputs from the universe that exists outside the interconnected systems.

It is necessary to formalize all these concepts in mathematics. First, in a set of systems it is necessary to adopt double subscript notation to distinguish the input terminals. The following definition creates a structure to use in unambigously specifying interconnections of systems.

Definition 3.2. An interconnection of n multiport finite state systems is a set:

$K = (M_1, M_2, \cdots, M_n, I, C)$ where:

a.) $M_j = (S_j, I_j, S_j, f_j, w)$ for each $j = 1,2,\cdots,$ n is a multiport finite state system, where

$$I_j = I_{j1} \times I_{j2} \times \cdots \times I_{jm_j},$$

$$S_j = S_{j1} \times S_{j2} \times \cdots \times S_{jn_j},$$

b.) I is a non-empty subset of $(I_{11}, I_{12}, \cdots, I_{1m_1}, \cdots, I_{n1}, \cdots, I_{nm_n})$,

c.) C is a set whose elements are ordered pairs: (M_j, I_{br}), where $S_j \subseteq I_{br}$ (the state set of system M_j is a subset of the input terminal I_{br}), and $I_{br} \notin I$,

d.) $I \cup \{I_{br}: I_{br} \in x \text{ for some } x \in C\} = (I_{11}, I_{12}, \cdots, I_{1m_1}, \cdots, I_{n1}, \cdots, I_{nm_n})$.

Informally, the set I will be referred to as the free input terminals of the interconnection K. The ordered pairs in C actually determine the structure of the interconnection. Each pair is understood to be an association of an input terminal for some machine with the output generated as the state of another system. Note that: $I_{er} \in I$ and condition d.) means any given terminal must be in either the set of terminals bound up in the interconnection or in the set of terminals free to receive input. The requirement that $I \neq \emptyset$ is in keeping with the convention argued in Chapter 2 that every system must have an input. An interconnection of systems (which defines another system) must also have an input.

As an example of the use of this notion, consider the example of the description of the following interconnection of two finite state systems. Given system

$M_1 = (S_1, I_{11} \times I_{12} \times I_{13}, S_1, f_1, w)$ and the system $M_2 = S_2, I_{21} \times I_{22}, S_2, f_2, w)$ assume that $S_2 \subseteq I_{12}$, $S_2 \subseteq I_{22}$, $S_1 \subseteq I_{13}$, $S_1 \subseteq I_{21}$. Then consider the interconnection: $K = (M_1, M_2, (I_{11}), ((M_1, I_{13}), (M_1, I_{21}), (M_2, I_{12}), (M_2, I_{22}))$. This interconnection is diagrammed in Fig. 6 below.

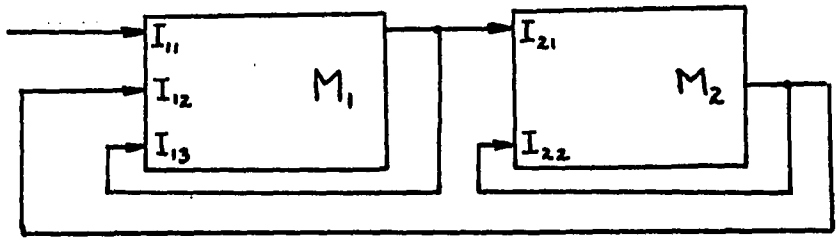


Fig. 6. Interconnection of Two Systems.

It is simple to construct the diagram from the set K. An element in the set of free input terminals is left with its input unconnected. All other inputs in the ordered pairs denoting the actual interconnections are connected by inputs from the appropriate machine.

Because the notational scheme of 3.2 consists of enumerating the interconnections it is obviously of use only when the number of systems and/or interconnections is finite. When interconnections of more general systems are studied, a more powerful notation is required, as is the case in Wymore's book (Ref. 24). For finite state systems the present notation is sufficient and possesses an advantage in being equally as concise..

With the means at hand to unambiguously describe the interconnections of finite state systems, attention can now be turned to the theoretical properties of such interconnections. Ideally, the goal of an analysis procedure would be to take a complex interconnection of systems and replace it with one system whose behavior, from the viewpoint of the free input terminals and state behavior, is the same as the complex interconnection. Before this can be considered, though, an important question must be answered. Intuitively, interconnecting systems should yield another structure which is also a system. Thus it is necessary to inquire as to how well-defined an interconnection of systems can be. It would be discomforting, to say the least, to find that for an arbitrary set of finite state systems there existed an interconnection of them that no longer behaved as a finite state system. This unfortunate situation does not occur, though, as the following definition and corollary will demonstrate.

Definition 3.3. The resultant of the interconnection $K = (M_1, \cdots, M_n, I, C)$ of n multiport finite state systems $M_j = (S_j, I_{j1} \times \cdots \times I_{jm_j}, S_j, f_j, w)$, for $j = 1,2,\cdots, n$, is the set M*, denoted $M^* = RES(K)$, such that $M^* = (S_1 \times \cdots \times S_n, I, S_1 \times \cdots \times S_n, \underset{\sim}{f}, \underset{\sim}{w})$ where $\underset{\sim}{f}$: $S_1 \times \cdots \times S_n \times I \rightarrow S_1 \times \cdots \times S_n$ such that:

$$\pi_j\ (\underset{\sim}{f}(s_1, \cdots, s_n, \underset{\sim}{i})) = f_j(s_j, x_1, \cdots, x_{m_j}), \quad (1.)$$

where for $b = 1, 2, \cdots, m_j$

$$x_b = \begin{cases} \pi_{jb}(\underset{\sim}{i}) \text{ if } I_{jb} \varepsilon I \\ \\ s_r \text{ if } (M_r, I_{jb}) \varepsilon C. \end{cases}$$

Corollary 3.1. M^* is a multiport finite state system.

Proof. Since in definition 3.3 the function f has the property that $f: S_1 \times \cdots \times S_n \times I \longrightarrow S_1 \times \cdots \times S_n$, and $\underset{\sim}{w}$ is the identity function, then M^* satisfies all the requirements of definition 2.5 and is a finite state system. The proof is complete.

The corollary is obvious, of course, from the manner in which the resultant, $M^* = RES(K)$, is defined.

The resultant of an interconnection is a multiport finite state system, hence by Theorem 2.6 possesses a multiport real number representation. The object now is to construct a real number representation for a resultant given the real number representations of the component systems in an interconnection. Consideration of the next-state function in definition 3.3 shows one possible way. Since the variables i_{jk} in the function f_j on the right-hand side of (1.) will be receiving the state of a system if that particular terminal I_{jk} is bound up in an interconnection, then $i_{jk} = s_u$, if the input comes from M_u.

Thus f_j would be a function of s_j, plus the states of the other machines providing input to the j^{th} system, and any inputs from the external world into the free input terminals of the j^{th} system. If each of the systems was described by a real number representation, then a function $\underset{\sim}{f}'$ could be constructed by taking each polynomial pf' and substituting an appropriate system state for the input variable of each terminal which receives an input from another system. This simple procedure would be sufficient to derive a next-state function $\underset{\sim}{f}'$ of the resultant. The following theorem is a rigorous demonstration of this conjecture and the method for constructing the real number representation of the resultant of an interconnection.

Theorem 3.1. Let $M_1, M_2, \cdots, M_n$ be a set of multiport finite state systems, and let $K = (M_1, M_2, \cdots, M_n, I, C)$ be an interconnection of these systems whose resultant $RES(K) = (S_1 \times \cdots \times S_n, I, S_1 \times \cdots \times S_n, \underset{\sim}{f}, \underset{\sim}{w})$. Let each system M_j have a multiport real number representation $M_j' = (S_j', I_j', S_j', f_j', w')$. Then the real number representation of the resultant is: $(S_1' \times \cdots \times S_n', I', S_1' \times \cdots \times S_n', \underset{\sim}{f}', \underset{\sim}{w})$, where $U \in I'$ if and only if $I_{rs} \in I$ and $U = I_{rs}'$, and $\underset{\sim}{f}' = res(p\underset{\sim}{f}', S_1' \times \cdots \times S_n' \times I)$, where $p\underset{\sim}{f}'$ is such that:

$$\pi_j\,(pf'(s'_1, \cdots, s'_n, i')) =$$

$$\sum_{k_{n_j}=0}^{m_{jn_j}-1} \cdots \sum_{k_1=0}^{m_{j1}-1} \sum_{k_0=0}^{n_j-1} a^j_{k_0 k_1 \cdots k_{n_j}} (s'_j)^{k_0} (x_1)^{k_1} \cdots (x_{n_j})^{k_{n_j}}, \tag{2.}$$

where $x_b = \begin{cases} \pi_{jb}(i'),\ i' \in I', \text{ if } I_{jb} \in I, \\ s'_r \text{ if } (M_r, I_{jb}) \in C, \end{cases}$

and where

$$pf'_j\,(s'_j, i'_j) =$$

$$\sum_{k_{n_j}=0}^{m_{jn_j}-1} \cdots \sum_{k_1=0}^{m_{j1}-1} \sum_{k_0=0}^{n_j-1} a^j_{k_0 k_1 \cdots k_{n_j}} (s'_j)^{k_0} \pi_1(i'_j)^{k_1} \cdots \pi_{n_j}(i'_j)^{k_{n_j}}, \tag{3.}$$

for every $s'_j \in S'_j$, $i'_j \in I'_{j1} X \cdots X I'_{jn_j}$, $j=1, 2, \cdots, n$.

Proof. It is necessary to show that the conditions of definition 2.6 are satisfied in order to prove the theorem. This means it is necessary to exhibit isomorphisms α and β such that:

$$\alpha(f(s, i)) = pf'\,(\alpha(s), \beta(i))$$

is true.

Begin with equation (1.) in defintion 3.3. That equation states that:

$$\pi_j\,(\underset{\sim}{f}(s_1, \cdots, s_n, \underset{\sim}{i})) = f_j(s_j, x_1, \cdots, x_{m_j}),$$

where for $b = 1, \cdots, m_j$

$$x_b = \begin{cases} \pi_{jb}\,(\underset{\sim}{i}) \text{ if } I_{jb} \varepsilon\ I \\ \\ s_r \text{ if } (M_r, I_{jb}) \varepsilon\ C \end{cases} \qquad (4.)$$

By the hypotheses of the theorem, each system has a real number representation. From the basic definition in Chapter 2, this means that there exists isomorphisms α_j and β_{jb} such that

$$\alpha_j[f_j(s_j, i_{j1}, \cdots, i_{jm_j})] =$$

$$pf'_j(\alpha_j(s_j), \beta_{j1}(i_{j1}), \cdots \beta_{jm_j}(i_{jm_j})), \qquad (5.)$$

for $j = 1, \cdots, n$, $b = 1, \cdots, m_j$.

But the f_j in square brackets on the left-hand side of equation (5.) is the same f_j on the right-hand side of equation (4.) before the interconnection K takes place to form the resultant, the formation of the resultant being governed by equation (4.). When the resultant is formed, each of the i_{jb} in f_j on the left-hand side of equation (5.) is replaced by x_b in accordance with equation (4.). This, however, must also cause changes on the left-hand side so that:

$$\alpha_j(f_j(s_j, x_1, \cdots, x_{m_j})) =$$

$$pf'_j(\alpha_j(s_j), \gamma_1(x_1), \cdots, \gamma_{m_j}(x_{m_j})), \qquad (6.)$$

where: $\gamma_b = \alpha_r(s_r)$ if $(M_r, I_{jb}) \in C$, and $\gamma_b = \beta_{jb}(i_{jb})$ if $I_{jb} \in I$, for $j = 1, \cdots, n$, and x_b satisfies the same conditions stated in equation (4.).

Equation (6.) is the direct result of acting on equation (5.) by constructing the resultant, RES(K), as required in definition 3.3. For each $j = 1, \cdots, n$, the left hand side of equation (6.) is changed by substituting either i_{jb} or s_r into the variables x_b, depending on the conditions in equation (4.). Similar action takes place on the right-hand side of (6.) only for the substitution of $\alpha_r(s_r)$ and $\beta_{jb}(i_{jb})$. When these substitutions are carried out for every $j = 1, \cdots, n$, the following is the result: The collection of all the f_j, for $j = 1, \cdots, n$, is a function of $s_1, \cdots, s_n$ and only those i_{jb} such that $I_{jb} \in I$. The right-hand side of (6.) is a function of $\alpha_1(s_1), \cdots, \alpha_n(s_n)$ and only those $\beta_{jb}(i_{jb})$ such that $I_{jb} \in I$. Let the collection of α_j's be an isomorphism on $S_1 \times \cdots \times S_n$, and let the collection of β_{jb}'s be an isomorphism on the Cartesian product of the input sets in I, X(I). Then equation (6.) can be seen as a component-by-component form of the general vector equation:

$$\underset{\sim}{\alpha}(\underset{\sim}{f}(s_1,\cdots,s_n,\underset{\sim}{i})) = p\underset{\sim}{f}'(\alpha_1(s_1),\cdots,\alpha_n(s_n),\underset{\sim}{\beta}(\underset{\sim}{i})), \qquad (7.)$$

for every $(s_1, \cdots, s_n) \in S_1 \times \cdots \times S_n$, and $\underset{\sim}{i} \in X(I)$. This is the operation preserving mapping required in definition 2.6.

However, the construction of the right-hand of equation (6.) is precisely the replacement of variables demonstrated in equations (2.) and (3.) in the statement of the theorem. Since equation (6.) is derived from equation (5.) by the action of constructing the resultant, RES(K), and exhibits the operation preserving property then equation (2.) gives the real number representation of the resultant based on equation (3.) for each system. The proof is complete.

Theorem 3.1 details the procedure by which a real number representation may be obtained for an interconnection of systems. It proves to be quite simple. Given a set of systems, each with a real number representation, then the real number representation of the resultant can be derived in two basic steps. First, each polynomial next-state function is examined and each input variable in the polynomial which is associated with input from another system is replaced by the state variable of the system from which it receives input. Second, the new set of polynomials are collected together to form a vector-valued next-state function. This is the next-state function of the real number representation of the resultant and gives, on a component-by-component basis, the state transitions of the resultant in response to the input variables which were

unreplaced. In Chapter 4 an example will be worked to illustrate the method and the manipulations necessary to apply it.

The next section illustrates the derivation of a synthesis procedure based on polynomial functions.

3.5. Synthesis by the Use of Polynomial Functions.

Synthesis is the very core of engineering design. Much of modern engineering practice is involved in writing proposals and specifications for systems. But once the proposal is accepted and the specifications agreed upon, then there must be a practicing engineer to translate the overall features of a system into a choice of components and interactions. Typically this is a process without unique solutions. Many different physical systems, all with different components and structures, can be formulated to perform a given function. It is up to the engineer to reach into this large class of systems and extract one, and thereby provide complete details on what the components are and how the components are interconnected.

The synthesis of sequential machines possesses these same features. The structure of a certain machine is completely known. It is desired then to decide upon a set of component machines and a set of interconnections of these machines so that the resultant of the interconnection

can "do the same things" as the given system. The quotes in the previous sentence emphasize the use of undefined terms. It is necessary to introduce two definitions in order to provide meaning and specific content to the concept of a machine being able to "do the same things" as another machine.

Definition 3.4. Given two finite state systems M_1 and M_2, then M_1 and M_2 are isomorphic if and only if there exist two isomorphisms between S_1 and S_2, and between I_1 and I_2 such that:

S_1 is isomorphic (α) to S_2,

I_1 is isomorphic (β) to I_2,

$f_2(\alpha(s_1), \beta(i_1)) = \alpha(f_1(s_1, i_1))$, for every $s_1 \in S_1$, $i_1 \in I_1$.

A similar definition can be made for isomorphism between two real number representations by simply inserting primes on the symbols M_1, M_2, I_1, I_2, f_1, f_2, S_1, S_2 in Definition 3.7. Note that Definition 3.7 casts isomorphism only in terms of the input and state set and the state transition function. This reflects two considerations: First, the ambiguity and arbitrariness of outputs from a system and secondly, the decision made in the early part of this chapter to be concerned with identity mappings as output functions for interconnected systems.

Definition 3.5. Given a finite state system $M = (S, I, O, f, g)$, then a finite state system $M_G = (S_G, I_G, O_G, f_G, g_G)$ is a subsystem if and only if it satisfies the properties:

$$S_G \subseteq S,$$

$$I_G \subseteq I,$$

$$O_G \subseteq O,$$

$$f_G: \; S_G \times I_G \rightarrow S_G, \; f_g = \text{res}\,(f, S_G \times I_G)$$

$$g_G: \; S_G \rightarrow O_G, \; g_G = \text{res}\,(g, S_G)$$

With these two definitions a dilemma can be resolved. Suppose it is desired to find the components necessary to synthesize a given machine. The work conducted in the last section shows that the resultant of the interconnection specified by the synthesis will have a vector-valued state set, hence as many distinct state pairs as the product of the number of distinct elements in the state set of each system. Suppose the given system has a prime number of states, though, such as thirteen. This state set cannot be "factored" so as to allow an assignment of the states in the given machine to ordered tuples of states in the components on a one-one basis. However, a collection of component systems could be used such that the product of the number of elements in the state set of each was greater than the prime number. Then the structure and interconnections of the components could be specified so that the

given system was isomorphic to a subsystem of the resultant of the interconnection. Such will be the strategy in this section.

The preceding paragraph outlines the structure of synthesis. The crucial point is in the development of a new state set. But this is viewed in a particular way in terms of the results of the previous section. After an analysis of an interconnection of sequential machines is completed, the resultant is described by a vector-valued state set and a next-state polynomial with a component for each state in the vector used to identify the state of the resultant. But the important aspect is that each polynomial was constructed by taking the polynomial of a given machine and substituting state variables for input variables involved in an interconnection. It would be desirable if the process could be reversed: take the state set of an original system and then develop a new vector state set. Then using the state transitions in the original system, derive a set of polynomials describing the mappings of this vector-valued state set. Whether or not a given system's polynomial is a function of the state of a particular system should tell whether or not that particular system provides input to the given system. Thus the structure of the component systems (from the polynomials for the next-state function) and the interconnections (from the

variables appearing in the polynomial) can be determined in one step. The following theorem will demonstrate, by construction, the synthesis procedure. In the statement of the hypothesis the notation #(s) is used to denote the cardinality of a set. Thus if there are n distinct elements in S, then #(S) = n.

Theorem 3.2. Given a multiport finite state system M = (S, I, S, f, w), if a finite set of finite sets of real numbers $(N_1, N_2, \cdots, N_r)$ is such that $\prod_{j=1}^{r} \#(N_j) \geq \#(S)$, then there exists a set of machines $(M_1, M_2, \cdots M_r)$ such that $S_j = N_j$ and an interconnection K of the (M_j) such that M is isomorphic to a subsystem of RES(K).

Proof. Assume that #(S) = n. It is necessary to represent the states in S by vector-valued states. Let the sets N_j, $j = 1, 2, \cdots, r$, be the state sets of a set of machines $(M_1, M_2, \cdots, M_r)$. It is necessary now to specify the inputs and transition functions of the machines, where $S_j = N_j$, $j = 1, 2, \cdots, r$.

Now define a function $\underset{\sim}{\alpha}$ with the following properties:

$\underset{\sim}{\alpha}$ is one-one,

$\underset{\sim}{\alpha}$: $S \rightarrow S_1 \times S_2 \times \cdots \times S_r$.

Since $\underset{\sim}{\alpha}$ is one-one, then every state in the original system M has only one image in the set of all r-tuples defined by the Cartesian product of the sets S_j. There may be

r-tuples which are not the image under α of an element in S, since $\#(S) \leq \prod_{j=1}^{r} \#(S_j)$ was allowed as a possibility.

It is now desired to choose the input sets for the systems. Knowing nothing about the system structure, it may be necessary for any component system to receive some or all of the input to the given system M. Also a component system may be a function of some or all of the states of other systems as well as its own state. Thus the input set of every system will be defined as:

$$I_j = X (I, I_{j1}, I_{j2}, \cdots, I_{jj-1}, I_{jj+1}, \cdots, I_{jr}),$$

where I is the input set of the given system and $I_{jk} = S_k$ for $k = 1, 2, \cdots, r, k \neq j$. This structure for I_j means every system will be able to receive an input from every other system except itself. There is no "feedback" from the output of a system to itself. Each machine is assumed to be a function of its current state and this state is available without direct feedback so that the state appears as an input variable.

Now consider the following interconnection of this set of systems. Let every system receive the original input I to M plus every system be interconnected with every other system. This is the most complex interconnection possible, the "worst case", and is given by:

$$K' = (M_1, M_2, \cdots, M_r, (I), ((M_k, I_{jk}) : k = 1, 2, \cdots, r, j = 1, 2, \cdots, r, k \neq j)).$$

Note in this interconnection that the free inputs are the set I, which is an input to every system. This means that each of the system M_j is receiving the same input I. Otherwise the free terminals would have been designated as I X I X ··· X I, signifying that the inputs to different systems could be different, provided it came from the set I. This is not allowed. All systems receive the same input, operating in "parallel" with respect to input, to use a term from electrical engineering.

It is desired now to find a next-state function f_j for each system M_j and concurrently to reduce the complexity of the K' to a simpler interconnection K.

Assume M is its own real number representation (See Definition 2.2). Then the function f comes from:

$f = res(pf, S \times I)$, where pf is a polynomial defined on R^2 with values in R.

Let the next-state function of each system M_j be given as the following:

$f_j = res(S_1 \times S_2 \times \cdots \times S_r \times I, pf_j)$ where pf_j is a polynomial defined on R^{r+1} with values in R.

By this structure, the j^{th} function evaluates the current state of the j^{th} system, and the current states (received as inputs) of all other systems, plus the input to the whole interconnection and computes the next state of the j^{th} system.

To determine each pf_j the following restriction can be utilized: for any transition in the original system, there must be a state transition of the image under $\underset{\sim}{\alpha}$ of the state in M that is the same. Thus each polynomial pf_j must be such that the operations of state transition will be preserved under the image of $\underset{\sim}{\alpha}$. This is equivalent to the equation:

$$pf_j(\underset{\sim}{\alpha}(s),i) = \pi_j(\underset{\sim}{\alpha}(f(s,i)) \qquad (8.)$$

for every $s \in S$ and $i \in I$.

Since $\underset{\sim}{\alpha}$ has been chosen already and the next state function f of M is known, then the right-hand side of equation (8.) is known. Similarly the left-hand side of (8.) is known except for the coefficients of the polynomial pf_j, which is a polynomial defined on $S_1 \times S_2 \times \cdots \times S_r \times I$. Equation (8.) is a familiar form: it represents the problem of finding a polynomial which passes through a given set of points, and this has already been studied in detail in the theorems of Chapter 2. The only difficulty is that $\#(S) \leq \prod_{j=1}^{r} \#(S_j)$ is allowed as a possibility, hence there may be fewer equations generated by substitution than unknowns. This can be resolved in two ways: either arbitrarily set some unknown coefficients to zero or add a sufficient number of equations.

Assume that one of the methods is used to reach a solution for these equations (they have a solution, being

the interpolation form used in previous theorems). Find the coefficients of the pf_j for each $j = 1, 2, \cdots, r$ by this means. The coefficients of the polynomial will define a new interconnection in the following fashion: if a particular state variable, say s_u, is chosen, then pf_j is made up of terms of s_u to the first and higher powers and other terms with s_u not included (s_u to the zero power). If all coefficients in terms involving s_u are zero, then pf_j is not a function of s_u and hence does not need to be interconnected to machine M_u. Thus by inspection of the pf_j an interconnection, K, can be derived. Let the resultant of this interconnection be $RES(K) = (S_1 \times \cdots \times S_r, I, S_1 \times \cdots \times S_r, \underset{\sim}{f}, \underset{\sim}{w})$, where $f = res(p\underset{\sim}{f}, S_1 \times \cdots \times S_r \times I)$, and $p\underset{\sim}{f}$ is the set of polynomials pf_j whose coefficients have just been found.

The mapping $\underset{\sim}{\alpha}$ is constructed so that:

$$\underset{\sim}{\alpha}(S) \subseteq S_1 \times S_2 \times \cdots \times S_r.$$

The polynomials pf_j were constructed from equation (8.) and hence they must satisfy:

$$pf_j: \ \underset{\sim}{\alpha}(S) \times I \rightarrow \underset{\sim}{\alpha}(S)$$

and since pf_j is a scalar function of vector variables the mapping is <u>into</u> $\underset{\sim}{\alpha}(S)$ (into the set S_j in $\underset{\sim}{\alpha}(S)$, to be exact).

Now let $\underset{\sim}{\alpha}(S) = S_G$, $\underset{\sim}{w}(I) = I_G$. Then:

$$S_G \subseteq S_1 \times S_2 \times \cdots \times S_r$$

$$I_G = I$$

$$\underset{\sim}{pf}: \quad S_G \times I_G \rightarrow S_G$$

Neglecting the identity outputs, it is definite that the image of S under $\underset{\sim}{\alpha}$ forms a submachine in the resultant, RES(K), by Definition 3.8. But using the mapping $\underset{\sim}{\alpha}$ and the identity w, it can be said that:

S is isomorphic ($\underset{\sim}{\alpha}$) to $S_G = \underset{\sim}{\alpha}(S)$

I is isomorphic ($\underset{\sim}{w}$) to $I_G = I$

And since the polynomials were constructed to satisfy equation (8.) then it must be that

$$\underset{\sim}{pf}\ (\underset{\sim}{\alpha}(s), \underset{\sim}{i}) = \underset{\sim}{\alpha}(f(s, \underset{\sim}{i}))$$

which is the necessary isomorphic property stated in Definition 3.7. The proof is complete.

This theorem is important in two ways, as several others have been. It demonstrates the possibility of conducting a synthesis and also presents a computational technique for actually carrying out the computational steps. The most noteworthy aspect of the method is that two results are achieved in one step. When the equations are solved for the coefficients of each pf_j, not only is the necessary structure of M_j found in the next-state function pf_j, but the presence or absence of state variables in pf_j tells what interconnections are required to provide input to the j^{th} system from other systems. Thus the structure of the components and the structure of the interconnections

are both found at the same time. But this is perfectly natural in view of the way that the analysis of interconnections was constructed in the previous section, input variables being replaced by state variables in the proper polynomials. Thus, the two methods have a rather symmetric appearance which appeals to mathematical aesthetics and is also quite useful in a practical way.

Note that in this section only multiport systems with scalar state sets were treated. If a multiport system has a vector-valued state set, then one possible synthesis is trivial to derive from the components of the next-state function. Each component is implemented. So only the scalar-valued state set presents appreciable problems in the synthesis.

The next chapter will present examples to illustrate how the computational techniques developed in Chapter 2 and Chapter 3 can be employed.

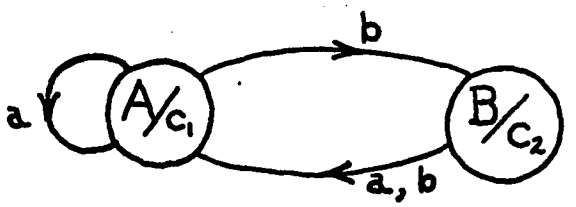
CHAPTER 4

ILLUSTRATIVE EXAMPLES

4.1. Construction of a Real Number Representation.

The purpose of this chapter is to demonstrate how the techniques developed in the previous chapters can be applied. To accomplish this task some simple examples will be presented to show the use of real number representations. The first example will show the mechanics of constructing a real number representation.

Example 1. The following finite state system is given.



$S = (A, B)$

$I = (a, b)$

$O = (c_1, c_2)$

Fig. 7. Machine of Example 1.

By specifying isomorphisms to the sets, S, I, and O a real number representation can be derived, as was proved in Chapter 2. Choosing three isomorphisms α, β, γ, the graphs of these functions, presented by ordered pairs, are:

$$\alpha = ((A, 0), (B, 1)), \alpha(S) = S' = (0, 1),$$

$$\beta = ((a, 0), (b, 1)), \beta(I) = I' = (0, 1),$$

$$\gamma = ((c_1, 1), (c_2, 0)), \gamma(O) = O' = (0, 1).$$

With the sets S', I', O' fixed it is now necessary to specify polynomials such that:

$$pf'(\alpha(s), \beta(i)) = \alpha(f(s,i)), \qquad (1.)$$

for every $s \in S$ and $i \in I$,

$$pg'(\alpha(s)) = \gamma(g(s)), \qquad (2.)$$

for every $s \in S$. The functions g and f are the graphical functions in the flow-graph of Fig. 7. Following Chapter 2, these polynomials take the form:

$$pf'(s',i') =$$

$$a_{11}(i')^0(s')^0+a_{12}(i')^1(s')^0+a_{21}(i')^0(s')^1+a_{21}(i')^1(s')^1$$

$$pf'(s',i') = a_{11}+a_{12}i'+a_{21}s'+a_{22}i's', \qquad (3.)$$

for every $s' \in S'$ and $i' \in I'$,

$$pg'(s') = b_1(s')^0+b_2(s')^1$$

$$= b_1 + b_2S', \qquad (4.)$$

for every $s' \in S'$.

Utilizing equations (1.) and (3.) for every $s \in S$ and $i \in I$ produces the following set of 4 simultaneous linear equations in 4 unknowns:

$$a_{11}+a_{12}(0)+a_{21}(0)+a_{22}(0) = 0$$

$$a_{11}+a_{12}(1)+a_{21}(0)+a_{22}(0) = 1$$

$$a_{11}+a_{12}(0)+a_{21}(1)+a_{22}(0) = 0$$

$$a_{11}+a_{12}(1)+a_{21}(1)+a_{22}(1) = 0 \qquad (5.)$$

This set is equivalent to the following set of 4 equations in 4 unknowns:

$$a_{11} = 0,$$

$$a_{11} + a_{12} = 1,$$

$$a_{11} + a_{21} = 0,$$

$$a_{11} + a_{12} + a_{21} + a_{22} = 0. \qquad (6.)$$

Solving this set of equations, the result is:

$$a_{11} = 0,$$

$$a_{12} = 1,$$

$$a_{21} = 0,$$

$$a_{22} = -1. \qquad (7.)$$

Hence the polynomial for the next state function is:

$$pf(s', i') = i' - i's', \qquad (8.)$$

for every $s' \in S'$ and $i' \in I'$.

Now it is necessary to compute the coefficients of the output polynomial. Again from Chapter 2, by using every value of $s \in S$ and equations (2.) and (4.), the following equations are generated:

$$b_1 + b_2(0) = 1,$$
$$b_1 + b_2(1) = 0. \qquad (9.)$$

This set has the solution:

$$b_1 = 1,$$
$$b_2 = 1. \qquad (10.)$$

Thus the output polynomial is:

$$pg'(s') = 1 - s', \qquad (11.)$$

for every $s' \in S'$.

The fact that these two polynomials pg' and pf' do indeed yield the tabulated function in Fig. 7 can be verified by direct substitution. The real number representation of machine A can be given as follows: Given machine A in Fig. 7, then under the mappings α, β, γ chosen, there exists a real number representation A' which is:

$$A' = (S', I', 0', f', g'),$$

where:

$S' = (0, 1)$, $S' = \alpha(S)$,

$I' = (0, 1)$, $I' = \beta(I)$,

$0' = (0, 1)$, $0' = \gamma(0)$,

$f' = res(pf', S' \times I')$, where pf' is a polynomial, $pf'(s', i') = i' - i' s'$,

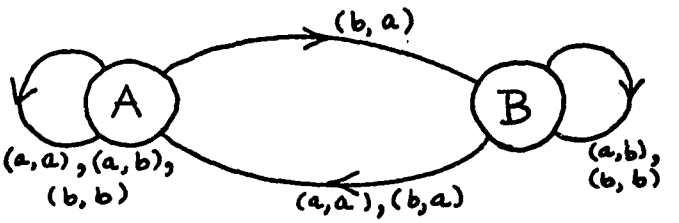
$g' = res(pg', S')$, where pg' is a polynomial, $pg'(s') = 1 - s'$.

This example illustrates the method to be used in generating real number representations of finite state systems.

4.2. Analysis of an Interconnection of Systems.

Now an example will be presented to illustrate the analysis techniques for finite state systems.

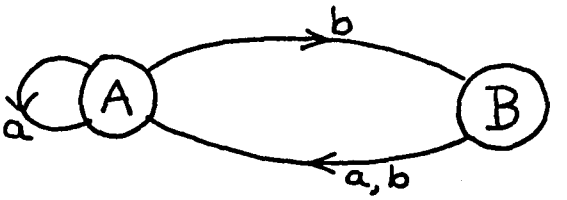
Example 2. Given the following two systems:



$S_1 = (A, B)$

$I_{11} = (a, b)$

$I_{12} = (a, b)$



$S_2 = (A, B)$

$I_{21} = (a, b)$

Fig. 8. Two Systems of Example 2.

The first machine is a multiport system and the second is the machine just examined in the first example. Output is not included with the second machine, though, in contrast to the first example.

Real number representations will be needed for both systems. Using the representation constructed in Example 1 for the second system, but with an identity output function:

$$M_2' = (S_2', I_{21}', S_2', f_2', w'),$$

where:

$$S_2' = (0, 1),$$

$$I_{21}' = (0, 1),$$

$$S_2' = (0, 1),$$

$$f_2' = res(pf_2', S_2' \times I_{21}'),\ pf_2'(s_2', i_{21}') =$$

$$i_{21}' - i_{21}'\ s_2',$$

$$w' = res(pw', S_2),\ pw'(s_2') = s'.$$

Now it is necessary to construct a multiport real number representation for the first system. Choose the following mappings, listed below by their graphs (sets of ordered pairs):

$\alpha = ((A, 0), (B, 1))$, thus $\alpha(S_1) = S_1' = (0, 1)$,

$\beta_1 = ((a, 0), (b, 1))$, thus $\beta_1(I_{11}) = I_{11}' = (0, 1)$,

$\beta_2 = ((a, 0), (b, 1))$, thus $\beta_2(I_{12}) = I_{12}' = (0, 1)$.

Following Chapter 2, the next-state polynomial must take the form:

$$pf'(i_{11}', i_{12}', s_1') = a_{111} + a_{112}s_1' + a_{121}i_{12}' + a_{122}i_{12}'s_1' + a_{211}i_{11}' + a_{212}i_{11}'s_1' + a_{221}i_{11}'i_{12}' + a_{222}i_{11}'i_{12}'s_1' \quad (12)$$

for every $i_{11}' \in I_{11}'$, $i_{12}' \in I_{12}'$, $s_1' \in S_1'$ it must satisfy the equation (from Chapter 2):

$$pf'(\alpha(s_1), \beta_1(i_{11}), \beta_2(i_{12})) = \alpha(f(s_1(i_{11}, i_{12}))) \quad (13.)$$

where f is the function defined by the graph of the system presented in Figure 8. Now all possible combinations of

the values in S and I are substituted into equation (13.) and this process, utilizing the polynomial in equation (12.), generates the following set of 8 equations in 8 unknowns:

$$a_{111}+a_{112}(0)+a_{121}(0)+a_{122}(0)+a_{211}(0)+a_{212}(0)+a_{221}(0)+a_{222}(0) = 0$$

$$a_{111}+a_{112}(0)+a_{121}(0)+a_{122}(0)+a_{211}(1)+a_{212}(0)+a_{221}(0)+a_{222}(0) = 1$$

$$a_{111}+a_{112}(0)+a_{121}(1)+a_{122}(0)+a_{211}(0)+a_{212}(0)+a_{221}(0)+a_{222}(0) = 0$$

$$a_{111}+a_{112}(0)+a_{121}(1)+a_{122}(0)+a_{211}(1)+a_{212}(0)+a_{221}(1)+a_{222}(0) = 0$$

$$a_{111}+a_{112}(1)+a_{121}(0)+a_{122}(0)+a_{211}(0)+a_{212}(0)+a_{221}(0)+a_{222}(0) = 0$$

$$a_{111}+a_{112}(1)+a_{121}(0)+a_{122}(0)+a_{211}(1)+a_{212}(1)+a_{221}(0)+a_{222}(0) = 0$$

$$a_{111}+a_{112}(1)+a_{121}(1)+a_{122}(1)+a_{211}(0)+a_{212}(0)+a_{221}(0)+a_{222}(0) = 1$$

$$a_{111}+a_{112}(1)+a_{121}(1)+a_{122}(1)+a_{211}(1)+a_{212}(1)+$$
$$a_{221}(1)+a_{222}(1) = 1$$

In spite of the appearance of 8 equations in 8 unknowns, many of the coefficients are zero and the system can be easily solved:

$$a_{111} = 0 \qquad a_{112} = 0$$
$$a_{121} = 0 \qquad a_{122} = 1$$
$$a_{211} = 1 \qquad a_{212} = -1$$
$$a_{221} = -1 \qquad a_{222} = 1 \qquad (14.)$$

By substituting these coefficients into the polynomial in equation (12.), the following is obtained for the next-state function:

$$pf'(i'_{11}, i'_{12}, s'_1) = i'_{12}s'_1+i'_{11}-i'_{11}s'_1-i'_{11}i'_{12}+i'_{11}i'_{12}s'_1. \qquad (15.)$$

This polynomial can be verified by tabulating the next-state function in the flow-graph of Figure 8 and making the comparisons according to the mappings α, β_1, and β_2. Thus the first machine has the following real number representation:

$$M'_1 = (S'_1, I'_1, S'_1, f'_1, w')$$
$$S'_1 = (0, 1),$$

$$I'_1 = I'_{11} \times I'_{12},\ I'_{11} = (0, 1),\ I'_{12} = (0, 1),$$

$$S'_1 = (0, 1),$$

$$f'_1 = res(pf'_1, S'_1 \times I'_1),\ pf'_1 (s'_1, i'_{11}, i'_{12}) =$$

$$i'_{11} s'_1 + i'_{12} s'_1 + i'_{11} - i'_{11} i'_{12} + i'_{11} i'_{12} s'_1,$$

$$w' = res(pw', S'_1),\ pw'(s'_1) = s'_1$$

Now that real number representations have been found for both systems, they will be interconnected and the resultant constructed. Consider the following interconnection:

$$K = (M_1, M_2, (I_{11}), (M_1, I_{21}), (M_2, I_{12})).$$

This is a relatively simple interconnection with one input terminal left free. The interconnection K can be diagrammatically presented, as in Figure 9. It is now desired to construct the real number representation of the resultant by using the real number representations of the components.

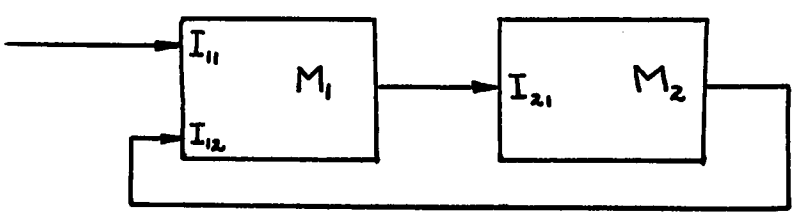


Fig. 9. Interconnection of Example 2.

By the technique developed in Section 3 of Chapter 2, the real number representation of the resultant can be written down directly. It is:

$$RES(K)' = (S_1' \times S_2' , I_{11}', S_1' \times S_2', \underset{\sim}{f}', \underset{\sim}{w}'),$$

where: $S_1' = S_2' = (0, 1)$,

$I_{11}' = (0, 1)$,

$\underset{\sim}{f}' = res(p\underset{\sim}{f}', S_1' \times S_2' \times I_{11}')$, $p\underset{\sim}{f}'(s_1', s_2', i_{11}') =$

$(i_{11}'s_1'+s_2' s_1' + i_{11}' - i_{11}' s_2' + i_{11}'s_2's', s_1' - s_1's_2')$,

$\underset{\sim}{w}' = res(p\underset{\sim}{w}', S_1' \times S_2')$, $p\underset{\sim}{w}'(s_1', s_2') = (s_1', s_2')$.

The polynomial pf' was constructed by replacing i_{12}' in equation (15.) by s_2', since the input to that terminal came from system M_2, and replacing i_{21}' by s_1' in equation (8.), since the input to that terminal came from system M_1. These two functions, after replacement of variables, became the first and second components, respectively, of the vector-valued next-state function $p\underset{\sim}{f}'$.

A complete table of the transitions of states in the resultant of the interconnection could now be constructed by substituting all possible values of s_1', s_2', and i_{11}' into the vector polynomial $p\underset{\sim}{f}'$. This in distinct contrast to Simon's method which would require first applying his method to reduce the cascade portion of the interconnection,

and then reapply on the resulting feedback connection. At each step, as well, the transition tables must be carried along and manipulated. However, construction of the resultant by using the polynomial functions eliminates the need for such processes. The polynomial state-transition of the resultant carries with it all the necessary information on the structure of each system and the structure of the interconnection. This is decidedly of advantage in constructing digital computer programs to simulate interconnections of finite state systems. Using the transition tables, such a program would have to be made of several fairly complex subroutines, and a great amount of information would have to be fed into each subroutine concerning the particular systems and the interconnection under study. In contrast, a program to simulate the interconnection of finite state systems by polynomial functions would need to contain only a subroutine to evaluate polynomials, given the structure of the polynomials. No interconnection information would need to be supplied, that being in the polynomials themselves. It may be argued that use of polynomials would require a simulation program to be changed everytime it was to be used. However, most advanced FORTRAN systems allow uncompiled subroutines to be included with a pre-compiled object deck. Thus, the user of a simulation program using polynomials would merely submit his pre-compiled

deck with a small FORTRAN subroutine containing the particular polynomials he was interested in using.

Further, new techniques in symbol manipulation languages may soon eliminate the need for the process of finding polynomials and then altering them for interconnection by substitution of variables. Recently, progress has been made with a programming language capable of performing symbol manipulations of equations, substitution of variables, etc. (Tobey, Ref. 23). This language, FORMAC, can be used to do substitution, expansion of products, etc., and then used to compute from the arithmetic expressions resulting from the manipulation. This creates the possibility, in the not too distant future, of removing all concern with the polynomial manipulations from the hands of a systems engineer who would care to use them. He would supply the transition tables and an interconnection. A FORMAC program would then derive a set of real number representations, manipulate them as specified by the interconnection, and do computations with the polynomials resulting from the manipulation. This would free the systems engineer from a great many wearying details. He could concentrate on using both the digital computer and the polynomial representations in the most efficient way possible.

4.3. Synthesis of a System by Polynomial Functions.

In this section an example will be presented to illustrate the synthesis procedure derived in Chapter 3.

Example 3. Given the following system:

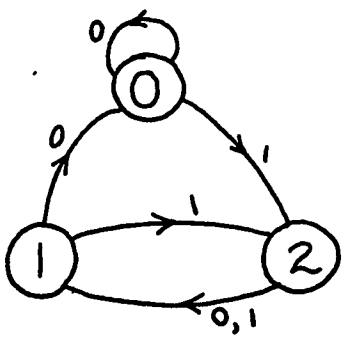


Fig. 10. System of Example 3.

It is desired to find a synthesis which realizes this system. There are three states in this system. Since this is a prime number, the smallest interconnection of systems which could contain this system as a subsystem would be two systems with two states. Let those two systems be designated:

$$M_1 = (S_1, I \times I_1, S_1, f_1, w),$$

$$M_2 = (S_2, I \times I_2, S_2, f_2, w),$$

where: $S_1 = S_2 = (0, 1)$

$$I_1 = I_2 = I = (0, 1).$$

Synthesis will specify both f_1 and f_2.

The two systems with two states have a Cartesian product of state sets, $S_1 \times S_2$. But $\#(S_1 \times S_2) > 3$. Hence it is necessary to add states to the original system in Figure 10 such that the system in Figure 10 is a subsystem of the new system. This is shown in Figure 11, where one state has been added.

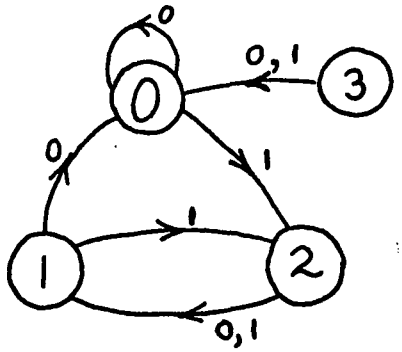


Fig. 11. Dummy State Added to System.

Assume now the worst possible (in terms of complexity) interconnection of the systems M_1 and M_2. Let this interconnection be $K' = (M_1, M_2, I, (M_1, I_2), (M_2, I_1))$. It is now necessary to use this K' to find the specific coefficients in the following two next-state functions (note the lack of primes. Since the machine in Figure 10 already had real number states, it is its own real number representation and the primes aren't necessary. See Definition 2.4).

$$pf_1(s_1, i, i_1) = \sum_{b=0}^{1} \sum_{k=0}^{1} \sum_{j=0}^{1} a^1_{jkb} (s_1)^j (i)^k (i_1)^b \qquad (16.)$$

$$pf_2(s_2, i, i_2) = \sum_{b=0}^{1} \sum_{k=0}^{1} \sum_{j=0}^{1} a^2_{jkb} (s_2)^j (i)^k (i_2)^b \qquad (17.)$$

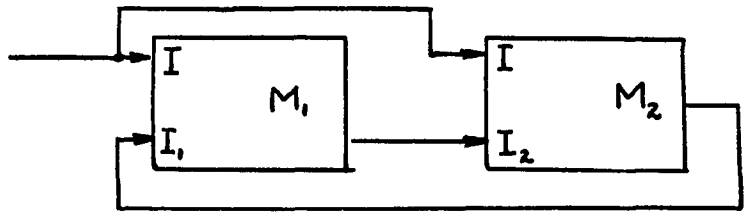


Fig. 12. Interconnection Assumed for Synthesis.

Note in these polynomials that the coefficients a^p_{jkb} have superscripts, p = 1, 2, to distinguish the two polynomials. Using the interconnection K', the input variables i_1 and i_2 are replaced by state variables. The polynomials then become:

$$pf_1(s_1, i, s_2) = \sum_{b=0}^{1} \sum_{k=0}^{1} \sum_{j=0}^{1} a^1_{jkb} (s_1)^j (i)^k (s_2)^b \qquad (18.)$$

and:

$$pf_2(s_2, i, s_1) = \sum_{b=0}^{1} \sum_{k=0}^{1} \sum_{j=0}^{1} a^2_{jkb} (s_2)^j (i)^k (s_1)^b \qquad (19.)$$

It is necessary now to make a mapping α from the states of the system (with a dummy state) in Figure 10 to ordered pairs of the states of M_1 and M_2. Let the ordered pairs be (s_1, s_2) where s_1 is the state of M_1, s_2 the state M_2. Define the mapping α as follows: $\underset{\sim}{\alpha}(s) = (\alpha_1(s), \alpha_2(s))$, and

$$\alpha_1 = ((0, 0), (1, 0), (2, 1), (3, 1)),$$

$$\alpha_2 = ((0,0), (1,1), (2,0), (3,1)),$$

where α_1 and α_2 are defined by the graph of their ordered pairs.

Given this mapping α it is now necessary to find f_1 and f_2 to satisfy the following equation:

$$pf_j(\underset{\sim}{\alpha}(s),\underset{\sim}{i}) = \pi_j(\underset{\sim}{\alpha}(f(s,\underset{\sim}{i})), \quad j = 1, 2,$$

which is from Chapter 3. By substituting every $s \in S$ and $i \in I$ in this equation sets of equations in the unknown coefficients will be generated. For $j = 1$ this is:

$$pf_1(\alpha_1(s),\alpha_2(s),i) = \alpha_1(f(s,i)).$$

Thus the substitution of every $s \in S$ and $i \in I$ generates the following:

$$a^1_{111}+a^1_{112}(0)+a^1_{121}(0)+a^1_{122}(0)+a^1_{211}(0)+a^1_{212}(0) + a^1_{221}(0)+a^1_{222}(0) = 0$$

$$a^1_{111}+a^1_{112}(0)+a^1_{121}(1)+a^1_{122}(0)+a^1_{211}(0)+a^1_{212}(0) + a^1_{221}(0)+a^1_{222}(0) = 0$$

$$a^1_{111}+a^1_{112}(0)+a^1_{121}(0)+a^1_{122}(0)+a^1_{211}(0)+a^1_{212}(0) + a^1_{221}(0)+a^1_{222}(0) = 1$$

$$a^1_{111}+a^1_{112}(0)+a^1_{121}(1)+a^1_{122}(0)+a^1_{211}(1)+a^1_{212}(0) + a^1_{221}(1)+a^1_{222}(0) = 1$$

$$a^1_{111}+a^1_{112}(1)+a^1_{121}(0)+a^1_{122}(0)+a^1_{211}(0)+a^1_{212}(0)+$$
$$a^1_{221}(0)+a^1_{222}(0) = 0$$

$$a^1_{111}+a^1_{112}(1)+a^1_{121}(1)+a^1_{122}(1)+a^1_{211}(0)+a^1_{212}(0)+$$
$$a^1_{221}(0)+a^1_{222}(0) = 0$$

$$a^1_{111}+a^1_{112}(1)+a^1_{121}(0)+a^1_{122}(0)+a^1_{211}(1)+a^1_{212}(1)+$$
$$a^1_{221}(0)+a^1_{222}(0) = 0$$

$$a^1_{111}+a^1_{112}(1)+a^1_{121}(1)+a^1_{122}(1)+a^1_{211}(1)+a^1_{212}(1)+$$
$$a^1_{221}(1)+a^1_{222}(1) = 0$$

The second polynomial whose coefficients must be evaluated can be also obtained by the same means:

$$pf_2(\alpha_1(s),\alpha_2(s),i) = \alpha_2(f(s,i))$$

where again this is to be evaluated for every $s \in S$ and $i \in I$. Also f(s,i) is the next-state function presented as a state flow-graph in Figure 11. Upon substitution the following set of equations is generated for a^2_{jkb}.

$$a^2_{111}+a^2_{112}(0)+a^2_{121}(0)+a^2_{122}(0)+a^2_{211}(0)+a^2_{212}(0)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 0$$

$$a^2_{111}+a^2_{112}(0)+a^2_{121}(1)+a^2_{122}(0)+a^2_{211}(0)+a^2_{212}(0)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 1$$

$$a^2_{111}+a^2_{112}(0)+a^2_{121}(0)+a^2_{122}(0)+a^2_{211}(1)+a^2_{212}(0)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 0$$

$$a^2_{111}+a^2_{112}(0)+a^2_{121}(1)+a^2_{122}(0)+a^2_{211}(1)+a^2_{212}(0)+$$
$$a^2_{221}(1)+a^2_{222}(0) = 1$$

$$a^2_{111}+a^2_{112}(1)+a^2_{121}(0)+a^2_{122}(0)+a^2_{211}(0)+a^2_{212}(0)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 0$$

$$a^2_{111}+a^2_{112}(1)+a^2_{121}(1)+a^2_{122}(1)+a^2_{211}(0)+a^2_{212}(0)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 0$$

$$a^2_{111}+a^2_{112}(1)+a^2_{121}(0)+a^2_{122}(0)+a^2_{211}(1)+a^2_{212}(1)+$$
$$a^2_{221}(0)+a^2_{222}(0) = 0$$

$$a^2_{111}+a^2_{112}(1)+a^2_{121}(1)+a^2_{122}(1)+a^2_{211}(1)+a^2_{212}(1)+$$
$$a^2_{221}(1)+a^2_{222}(1) = 0$$

These two systems, though fairly large, are not too difficult to solve by hand. The results are:

$a^1_{111} = 0$ $\qquad$ $a^2_{111} = 0$

$a^1_{112} = 0$ $\qquad$ $a^2_{112} = 0$

$a^1_{121} = 0$ $\qquad$ $a^2_{121} = 1$

$a^1_{122} = 0$ $\qquad$ $a^2_{122} = -1$

$a^1_{211} = 1$ $\qquad$ $a^2_{211} = 0$

$a^1_{212} = -1$ $\qquad$ $a^2_{212} = 0$

$a^1_{221} = 0$ $\qquad$ $a^2_{221} = 0$

$a^1_{222} = 0$ $\qquad$ $a^2_{222} = 0$

Thus the two polynomials are:

$$pf_1(s_1,s_2,i) = i - i\,s_1$$

$$pf_2(s_1,s_2,i) = s_1 - s_1 s_2$$

By examining these polynomials it can be seen that pf_1 is not a function of s_2, hence needs no connection from machine M_2. Similarly, the polynomial pf_2 is not a function of i, the input, but is a function of s_1, so a connection must run from M_1 to M_2 to provide the value of s_1 to machine M_2. The following figure shows the actual interconnection, K, which is required for the synthesis:

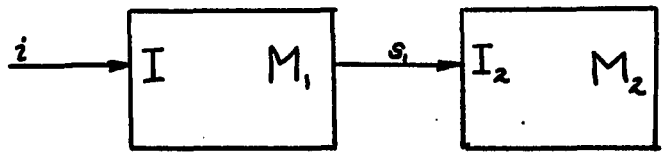


Fig. 13. Actual Synthesis of Systems.

This interconnection is $K = (M_1, M_2, I, (M_1, I_2))$, where M_1 and M_2 are the systems:

$$M_1 = (S_1, I, S_1, f_1, w),$$

$$f_1 = res(pf_1, S_1 \times I),\ pf_1(s_1,i) = i - i\,s_1,$$

$$w = res(pw, S_1),\ pw(s_1) = s_1,$$

$$M_2 = (S_2, I_2, S_2, f_2, w),$$

$$f_2 = res(pf_2, S_2 \times I_2),\ pf_2(s_2,i_2) = i_2 - i_2 s_2,$$

$$w = res(pw, S_2),\ pw(s_2) = s_2.$$

It is easy to verify, using the method illustrated in Example 2, that RES(k) gives the desired system.

4.4. Analog Computation with Finite State Systems.

The last example to be presented will show a unique application of the polynomial functions. Using the real number representation it is possible to simulate a finite state system by analog computer. In this section an example will be presented to illustrate the combined simulation of continuous and finite state systems.

Example 4. Consider a process which can be described by a first order differential equation:

$$\frac{dy}{dt} + ay = F. \qquad (20.)$$

Assume that F is a constant. It is now desired to control this process in the following manner: It is of no concern if the magnitude of y(t) is greater than the steady state solution, F/a. However, whenever the value of y(t) falls below the value F/a - δ, then it is desired to exercise control and apply a positive input to force the process back above F/a - δ. Thus, if a negative input, p, is applied to the system, then it is necessary to apply a positive control, u, to force the steady state value back above F/a - δ.

$$\frac{dy}{dt} + ay = F - p + u \qquad (21.)$$

The controller which applies the control input is of a special type, though, a finite state controller. At specific intervals of time the controller samples the current value of y(t) and makes state changes. The controller is assumed to start in a state of no control applied, the zero control state. When it samples y(t) and finds the value below F/a - δ, then it makes a change of state to positive control. But the internal structure of the controller is such that, following a sample time in the positive control state the controller must return to zero control state, no matter whether or not the process is still too low.

This controller, which must return to zero state following control, to "rest" itself so-to-speak, can be modeled by a two-state finite state system. The following is the state flow-graph of the controller:

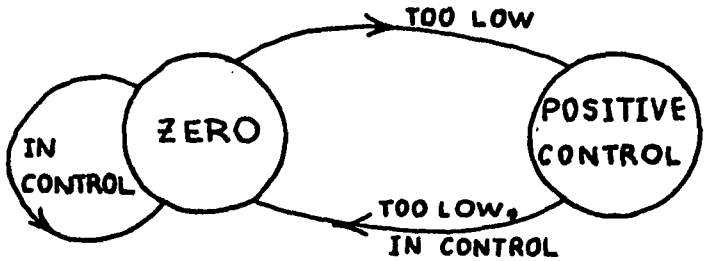


Fig. 14. Two State Controller.

Again, the desire is to find a real number representation of this system. Designating the functions α and β by the following graphs: α = ((ZERO STATE, 0),(POSITIVE CONTROL STATE, 1)), β = ((IN CONTROL, 0), (TOO LOW, 1)). The polynomial of state transitions takes the form:

$$pf(s,i) = \sum_{k=0}^{1} \sum_{j=0}^{1} a_{jk}(s)^j(i)^k. \qquad (22.)$$

Using the functions α and β, four equations in four unknowns may be generated. When solved the solution is:

$$a_{00} = 0, \qquad a_{01} = 1,$$

$$a_{10} = 0, \qquad a_{11} = -1. \qquad (23.)$$

Thus the state transition polynomial is

$$pf(s, i) = i - is = i(1-s). \tag{24.}$$

for the controller.

To formulate an equation for the system behavior, assume that the forcing function which tends to drive the process too low can be represented by a time function p and that, since only values of the process being too low are of concern, p is always subtracting from F. Let a maximum value of p exist and be known, p_{max}. Similarly, let the control input be a $+U_0$ or zero, depending on the state of the controller. Then the process can be described by the equation:

$$\frac{dy}{dt} + ay = F - p + U_0(s_k), \qquad p > 0, \tag{25.}$$

where s_k is the current state of the controller. Since the state of the controller is +1 or 0, then the control input is:

$$U_0(s_k) = U_0(i_{k-1} - i_{k-1}s_{k-1}) = U_0(i_{k-1}(i-s_{k-1})). \tag{26.}$$

But the function i_k is a function of the output y(t). It is the function described by:

$$i_k = 0 \quad \text{if } y(t) > F/a - \delta$$

$$i_k = +1 \quad \text{if } y(t) < F/a - \delta \tag{27.}$$

Let this function be g(y(t)). Then the complete process is described by the equation:

$$\frac{dy(t)}{dt} + ay(t) = F - p(t) + U_0(g(y(t))(1-s_{k-1})), \qquad (28.)$$

$$k = 1, 2, 3, \cdots.$$

This equation can be simulated on an analog computer. This is surprising, in that two fundamentally different kinds of systems are involved, continuous and finite state systems. The simulation of the differential equation by analog is a commonplace application. However, the polynomial functions make simulation of the finite state system simple as well. First, the state transitions of the finite state system are described by a polynomial function. Since the ordinary arithmetic operations of addition, subtraction, and multiplication can be performed by analog equipment, then an analog program can be constructed to compute state transitions from the polynomial.

The second aspect of simulation of finite state systems by analog is the sampled behavior. A finite state system operates, as has been discussed in Chapter 2, by computing the next state on the basis of the current input and current state. This suggests, for polynomial evaluation, that the current state and input must be saved for computation of the next state. The sampling and storage of a value is not difficult on the analog even though it is

unconventional. Bekely and Gerlough (Ref. 2) describe how to program a sample-and-hold circuit by using an integrator, and Korn (Ref. 18) shows how to connect two such circuits to obtain a complete memory or storage circuit.

The analog simulation can now be completely described. An integrator is used to solve the differential equation of the process. The output of the process is then digitized into either a 0 or +1. This is done by diode circuits and an operational amplifier. This discrete signal is then fed into a memory circuit. The output of the memory circuit goes to a program to compute the state transition polynomial. The output of this computation is the current state. It is fed into a memory circuit and back into the polynomial program. The state also goes to the integrator simulating the differential equation after being scaled by the magnitude of the control, U_0. See Appendix I for complete details and sample results of a simulation of this system.

With the simulation of a finite state system by analog computer, a new dimension is opened to the systems engineer. One of the chief advantages of the analog computer is in the ability to vary easily parameters in a direct simulation of system behavior. Simulation of continuous and finite state systems becomes approachable by the same techniques, using polynomial functions. Also in

Appendix I, a parameter study of the finite state controller was carried out on the analog to illustrate the use of the analog in parametric studies of finite state systems.

This example illustrates a truly unique usage of polynomial functions. It is not easy to see how simulation of finite and continuous state systems directly on an analog computer can be made without the polynomial representation. The application of this technique should make new efforts in simulation possible. However, only time and the study of a large number of examples will ultimately allow the judgment of the usefulness of this new technique. It does appear promising, though.

CHAPTER 5

CONCLUSIONS AND SUGGESTED EXTENSIONS

5.1. The Function of Extensions

One of the most fascinating aspects of any research activity is that usually a concentrated research effort will raise a great deal more questions than it will answer. To be sure, the end result of a piece of research is to gain a new understanding of some process or phenomenon. But this new understanding will carry with it a large number of hypotheses and differing viewpoints. These, in turn, make the researcher curious about how sensitive his bit of newly won knowledge is to changes in the hypotheses or viewpoints. Or they spur him to wondering what major new results could be achieved by using his new understanding as a foundation for further research. It is this action which makes research an infinite and unbounded process.

The open-ended nature of research requires that, as an integral part of every research effort, the researcher does more than just draw his conclusions. Acting under the philosophy that the questions that are posed are as important as the ones that have just been answered, it is

necessary to identify what may be important areas of work related to the research which has been previously completed. Sometimes the pursuit of a conjecture of suggestion has generated much valuable knowledge in a given area, but without resolving the conjecture. The classic example of this is in "Fermat's Last Theorem", a conjecture by a great French mathematician which has never been proved or disproved but nonetheless has been invaluable in the impetus it gave to certain fields of algebra. Thus, this chapter will contain not only the conclusions that can be drawn from the research undertaken, but also some suggested extensions of this research. The possible extensions are offered not in the hope of posing another conjecture like "Fermat's Last Theorem", but in the hope of illustrating that a great deal of very interesting questions have been raised in the process of pursuing the initial goals of the research.

Since extensions are closely related to the conclusions and final results, the conclusions from this research are presented first, in the folllowing section, then the possible areas of extension.

5.2. Conclusions

The following are the major conclusions that can be drawn from this work. In each case the conclusion is supported by reference to the work in the previous chapters.

(I.) The real number representation, as an alternate mathematical formulation of a finite state system, is a structure consistent with conventional formulations for finite state systems.

This conclusion is supported by the development, in Chapter 2, of theorems detailing the isomorphism that exists between conventional formulations of finite state systems and the real number representation. Thus all properties possessed by a finite state system are also possessed by its real number representation. It was also shown in Chapter 2 that every finite state system possesses a real number representation that is unique for a specified choice of elements in the state, input, and output sets. These same properties also held true when generalized to multiport finite state systems in the development of the last sections of Chapter 2. Again an isomorphism was established between multiport finite state systems and multiport real number representations. The real number representation is consistent with the common structures chosen for finite state systems because of these isomorphisms.

(II.) The real number representation is a useful tool in the analysis of interconnections of finite state systems.

This conclusion is justified by the contents of Chapter 3, except for the last section which deals with

synthesis. In that chapter the concept of interconnections of finite state systems was defined and given mathematical structure. Then it was shown that, for any conceivable interconnection of multiport finite state systems, the resultant was well-defined and also had to be a system possessing a real number representation. Hence, the problem of analysis of an interconnection can be treated as the equivalent problem of finding the real number representation of the resultant system. It was in turn shown that given the real number representation of each system in an interconnection of systems, then the real number representation of the resultant could be easily formed by simple modifications to the invididual polynomial functions. (III.) The real number representation is a useful tool in the synthesis of an interconnection of finite state systems to realize a given finite state system.

This conclusion is supported by the last section of Chapter 3. There it was shown that, by using a number of multiport finite state systems interconnected in the most complex manner, the real number representation could be employed to find simultaneously the specific form of the individual systems and the simplifications of the interconnection that might be possible. The given system was then shown to be isomorphic to a submachine of the system resulting from the interconnections. Thus the real number

representations of the component systems in a synthesis convey sufficient information to ascertain the structure of any individual system and also how the individual system is interconnected with the other components. In the case where the given system is synthesized as a submachine of an interconnection, the fact that there are more unknowns than linear equations in the coefficients of the individual real number representations means the designer has some freedom in specifying the form of individual components, or the form of the system determined by the interconnection of those components. This was illustrated in Chapter 3 and the example in Chapter 4.

(IV.) The computational techniques required for the evaluation of real number representations are simple.

The underlying technique in all the theorems and examples in this work has been the use of polynomial functions of real numbers and the specification of these polynomials by their coefficients. In all cases these coefficients have been shown to be obtainable as the solutions to sets of simultaneous linear equations. Furthermore, it has been shown that these solutions always exist. Given the assurance that a set of simultaneous linear equations always has a solution, the obtaining of that solution is trivial in theory. The emphasis on "theory" is important, for the obtaining of that solution by any of the well

known procedures can be a problem from the viewpoint of computational difficulty or accuracy. But theoretically, at least, the coefficients of the polynomials can always be found with any degree of accuracy. Speaking from a practical viewpoint, the ability to solve linear equations with a high degree of accuracy and speed has been resolved to quite a large degree by the digital computer, and most computing installations possess "canned" programs for such a task.

(V.) Finite state system models and the use of real number representations can be of great value in problems involving the analysis of general systems.

This last conclusion is supported by the developments in Chapter 3, showing new techniques for analysis and synthesis with finite state systems. It is also supported by the last example in Chapter 4, where a continuous and a finite state system were simulated on one analog computer. This example provided a graphic illustration of a general system with a finite state component. There could be many other possibilities for similar applications. The new techniques of analysis make easy the determination of the resultant of an interconnection of finite state systems. This, in turn, should allow more freedom for the systems engineer to model general systems, such as organizations of people or machines, by finite state components. The

analysis techniques could then be applied to study the organization's behavior. Similarly, an organization of people or machines could be synthesized by using the synthesis procedure. Applications to general systems would be particularly fruitful, since other classical techniques, such as differential equations, are not applicable.

5.3. Suggested Extensions.

The following are extensions suggested by the previous conclusions and various facets of the development in the previous chapters.

(I.) Extensions to the treatment of real number representations.

The foundation of all the work in Chapter 2 rests on interpolation. The polynomial functions chosen are merely an application of polynomial interpolation so that a function takes on a specified set of values on a set of values in the domain. Polynomial interpolation is not the only scheme involving such a procedure. In fact, for the simultaneous linear equations of the coefficients to be solvable it appears that the only requirement is that the set of interpolation functions be linearly independent. Any orthogonal set of interpolation functions should be usable as a set of functions for constructing the behavior of the state-transition and output functions in a real

number representation, as a result. The utility of any other such set in the machine representations is doubtful, but a thorough study might reveal interesting properties.

A much more interesting aspect of the real number representation lies in limiting behavior. The technique of using the restriction function guaranteed that the polynomials for next-state and output were limited properly to the finite sets that they were constructed from. But these polynomials could be freed from the restriction function and the geometrical behavior, plotting next state versus current state and input, could be studied for a system with an infinite amount of states, acting in discrete time. It is of some interest to conjecture how some given finite state system is related to such an infinite state system. Or, conversely, it is of interest to examine how an infinite state system is the "limit" of a sequence of finite state systems. The surface described by the interpolation functions can be described with more and more accuracy, corresponding to using more and more points in the surface. In the limit the polynomials for the finite state systems become series, and the convergence of the series defines a new system. Put another way, the finite state systems would appear to converge to some limit.

A rigorous treatment of these limiting behaviors would be quite complex, because concepts of topology and

convergence would have to be introduced into a work that is strictly algebraic thus far. Wymore has only just begun to consider the problems of topology in systems, and perhaps the completion of his work may provide the first foundation for approaching this last extension of the real number representation.

(II.) Extensions to Computational Techniques.

The foundation of the work, as it was stated above, is basically the solution of simultaneous linear equations so as to find the coefficients of the polynomials of a real number representation. The ability to derive a real number representation rests on the ability to generate the proper system of linear equations in the coefficients, as a result. In the work of the previous chapters methods were presented, and examples worked, to show how the proper equations may be derived in a given situation. There may be more efficient shemes of generating such equations than the ones proposed and illustrated, though. It is possible that bringing to bear a large amount of analytical power specifically on the problem of generating these equations would prove fruitful and result in simpler methods for constructing the proper equations to obtain a real number representation. The current work, being concerned with clarifying the strictly mathematical problems related to real number representations, has, out of necessity, overlooked the technical

problems of generating the right set of equations. Instead, the emphasis has been on the theoretical ease of solution of the equations for the required coefficients. Actually the situation may not be as bad as here stated. As was shown in previous chapters, the equations can be generated by brute-force enumeration using a transition-table or output table. This kind of work is easily relegated to a digital computer and perhaps eliminates to a great degree the need for more elegant schemes to create the proper equations in the unknown set of coefficients.

Once the equations have been solved and the coefficients are known, the questions of accuracy of the solution arise. If more computation is to be done with a polynomial, then errors in the coefficients tend to be reflected in errors in the next-state, which are then amplified when iterated further by resubstitution in the next-state function again. This can be eliminated, if the polynomial is being evaluated on a digitial computer, by always "rounding" the result of a computation off and then making certain that the real numbers in the set involved are "far enough apart" so that round off doesn't confuse the process. This question essentially becomes involved with numerical analysis aspects of using the polynomials on a digital computer. Some practical approaches are probably needed as well as some analytical work. It is possible that a good deal of

experimentation on a digital computer would be sufficient to indicate the magnitude of the problem as well as ways it could be alleviated.

(III.) Extensions in System Modelling.

If persuasion is to be given to the argument that finite state systems can be used as components in the analysis of very complex systems, then many different applications and examples must be modelled with finite state systems appearing in the analysis as a component or components. Not just one simple example, such as the differential equation and finite state controller, but literally hundreds of examples should be attempted. For example, there may be many situations where the human being could be modelled as a finite state system. Suppose that in the last example of Chapter 4 that the finite state controller was to be implemented by a human operator observing a process and then controlling it. Extensive analog simulation could then be used to study the properties of a system with a human element, the human element being viewed as a finite state system. In general, even though the human being is not a finite state system, there is probably an unbounded field of possible applications where, for specific situations, the human operator could be modelled as a finite state system. One large area of research could be undertaken by utilization of the fact that in many cases a

human being acts as a combination of an analog and digital system. Using real number representations a combined simulation for some particular task could reflect a modelling of both the analog and discrete aspects of a human operator. The overall operator simulation could then be combined with other models and programs to simulate total system dynamics.

There are many other possible applications for the real number representations and finite state machine models. A large number of applications will ultimately bear out -- or reject -- the usefulness of them. In the meantime those applications await to be tried and tested. Perhaps in this place, more than any other, a great deal of work needs to be placed in extension to that suggested here.

5.4. Closing Remarks.

A few final remarks may be made about this work in the more general framework of system theory, exclusive of specific considerations for finite state systems. One of the aims of system theory is to demonstrate relationships between various kinds of systems, or to demonstrate the fundamental similarities and differences between various different systems. Some of the work in this dissertation reflects such a goal. For example, the use of analog components to carry out the polynomial evaluation of a real number representation is an interesting point. Intuition

would demand that digital systems, with a finite (if extremely large) set of states, should be related to analog systems in some way. The real number representation gives the basis for just such a relationship. A digital computer is just a very large finite state machine and as such it must be describable, in theory at least, by a suitably large real number representation. Since such a real number representation can be evaluated by the usage of analog devices, as was illustrated in Chapter 4, then there is theoretical support for the concept of simulating a digital computer on an analog computer! There is no practical hope of doing so, of course, but the theoretical relationship between the two of them is interesting.

Also viewed from the context of general system theory, the question of limiting systems and convergence of a very large finite state system to a system with infinite states is very intriguing. Also the method of plotting the input versus the state space along with the space of the next state to form surfaces, as was done initially in Chapter 2, may be of interest for more general systems. The technique is somewhat reminiscent of a phase plane for a system governed by non-linear equations, but the input totally changes the nature of the plot. It would be interesting from the standpoint of general system theory to

study how surfaces in the space formed by the three axes of input, state, and next state characterize various systems.

Finally, it is interesting from the view of system theory to point out the consistency of results and procedures that the development has produced. For example, it has been shown that every finite state system has a real number representation. This is comforting in the sense of system theory, consistency and patterns being two of the aesthetic aspects of mathematics. In a similar vein it is proved that every interconnection of systems can also be described by a real number representation for the system that results from the interconnections. Equally as remarkable is the synthesis theorem that shows that a given system can be synthesized in terms of the real number representations of its components. In all these cases the polynomial functions have been always available to do service in solving particular problems. At the same time, a consistent procedure for constructing a given real number representation, the solution of a set of simultaneous linear equations, has also served as the principal scheme. Any branch of mathematics is, in some way or other, an attempt to establish a pattern or a degree of consistency over the elements or abstractions it describes. System theory, being principally founded on mathematics and utilizing it in many ways, is no different. The consistency of

results, as obtained herein, is rewarding from such a viewpoint. The disturbing thing would be the appearance of inconsistencies or lack of patterns. Such a happening would compel the researcher to wonder what possibly could be wrong in his initial approach, or become concerned with how his apparent inconsistencies could be encompassed within a more general framework.

The emphasis on system theory should not be allowed to hide the applicability of the work in this dissertation, though. It is felt that there is possibility for real applications in systems engineering of the techniques described herein. If they also have relevance in a system theory context, so much the better. Just as circuit theory serves the electrical engineer, so will system theory come to serve the systems engineer.

APPENDIX I

In this Appendix the detailed simulation of the differential equation and finite state controller discussed in Example 4 of Chapter 4 will be shown. First in order to have system dynamics slow enough to be easily studied, a large time constant was chosen for the system. The value of a = .04 in the equation

$$\frac{dy}{dt} + ay = F$$

was found to be suitable. A steady-state value was chosen next. Since y(t) = F/a in steady-state a value of F = 2 yielded a steady value of 50. Finally it is necessary to choose the limit δ defining the lower limit of control. For δ = 5, then the lower limit of control was 45. The maximum value allowed for the perturbation force, p, was set at 1. The differential equation to simulate was:

$$\frac{dy}{dt} + .04\ y = 2$$

The analog computer used in the simulation was an Applied Dynamics AD-64 owned by the Aerospace Department. Since this machine had a reference voltage of $\pm$100 volts it was not necessary to scale the problem, and voltages on the analog were directly the same as problem units (which are arbitrary and unspecified).

One integrator is sufficient to solve the differential equation. The output of that integrator must then be digitized to zero or one, in accordance with equation (27.) in Chapter 4. This was done in two steps. First 45 volts was subtracted from the integrator output. This voltage was then applied to an operational amplifier with diodes in the feedback path. Such an amplifier becomes a "bang-bang" type of system, showing almost a step response in its change of output when the input changes from negative to positive. Since the multiplier network, used to evaluate the next-state function, computed with greatest accuracy for full-scale voltages, the output of the diode-amplifier combination was adjusted to produce a full-scale $+100\frac{v}{-}$ output when the input became negative. When input was positive the output was desired to be zero, but the small forward drop in the diodes made it possible to reduce the output to a minimum of only about $\frac{1}{2}\frac{v}{-}$. This was very small compared to the full scale value of $+100\frac{v}{-}$. The output of the integrator was then applied to a track-hold or "memory" circuit. This is a circuit formed by two integrators without any inputs to the grid of the integrator. Instead the integrator input is applied to the initial conditions circuit of the integrator. The two integrators are connected with input into the I. C. circuit of the first and the output of the first going into the I. C. circuit of

the second. They are then operated in complementary modes. When the first is in reset mode it is tracking the input while the second is in hold mode, storing a previous value. Then the second goes into reset as the first goes in hold. The second then provides the output just stored into the first.

The output of the memory circuits is the value of the previous input. How often the memory circuits sample and store the input is determined by how rapidly the integrators are switched between modes. The output of the memory circuit is applied to one input of a multiplier. The other input to the multiplier is the quantity $(1-s_{k-1})$, scaled to 100 volts. The output of the multiplier is the current state, based on previous state and input. This state is then multiplied by the value U_0, by potentiometer, and fed back into the integrator simulating the dynamics of the differential equation. Also the value of the current state is used as input to another pair of integrators connected as a memory circuit. The output of this memory circuit is then fed into the multiplier after forming $(1-s_{k-1})$. The memory circuits storing the previous state are operated at the same switching rate as the memory storing the previous input. See Fig. 15 for the complete analog simulation diagram of the system, both differential equation and finite state system.

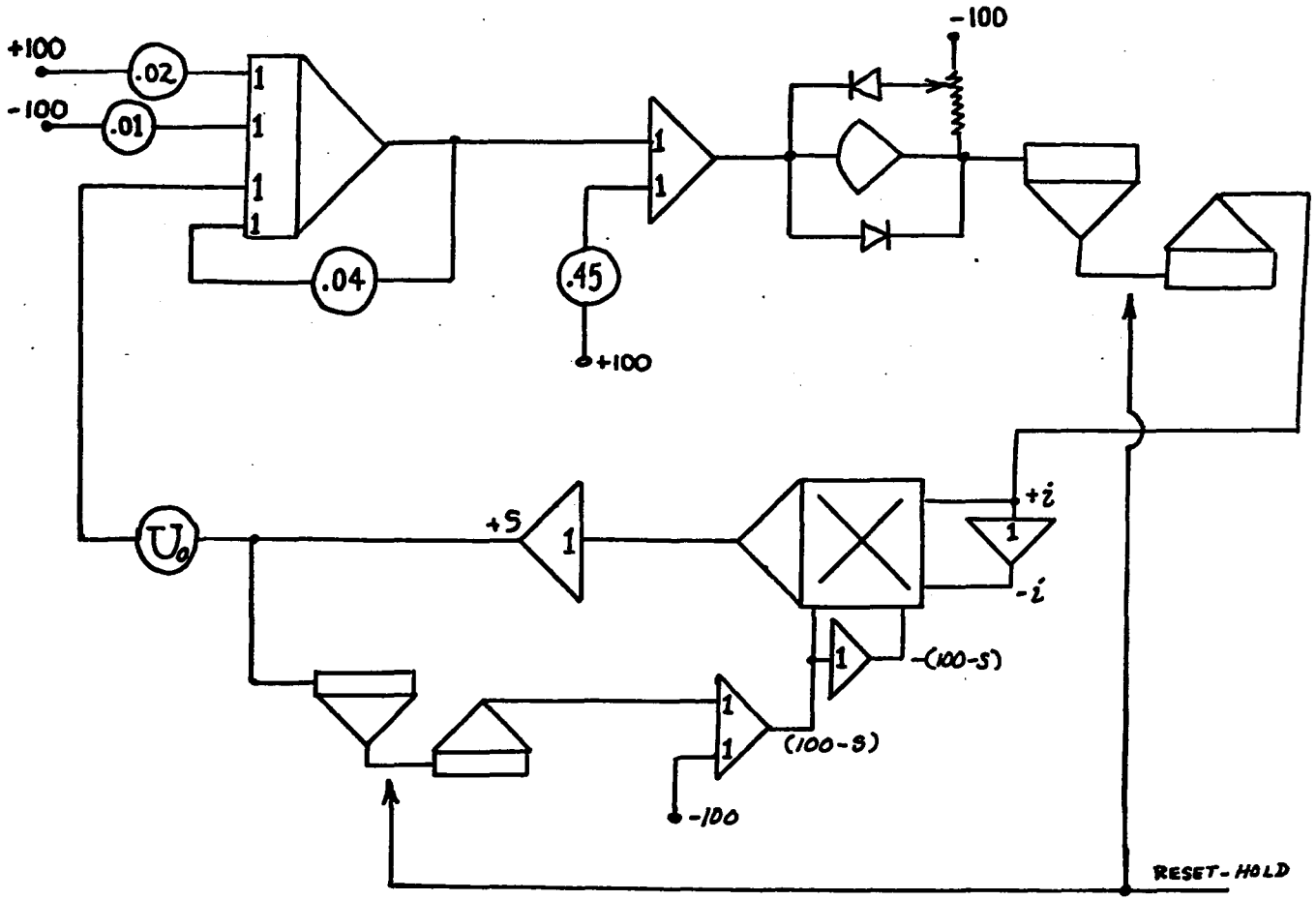


Fig. 15. Analog Simulation of Differential Equation and Finite State System.

The operation of the system can now be described. The two pairs of memory circuits store the current input and state when their integrators receive reset and hold pulses (which were supplied by a separate pulser unit on the AD-64 that allowed operation of these integrators independently of the other integrator). The values currently stored being the values of input and state previously sampled and stored by the previous reset and hold pulse cycle. An output is computed from the present state and applied to the input of the first integrator. If a perturbation is introduced and the output of y(t) falls too low then the output of the diode-amplifier combination rises to +100. This is stored and upon the next sampling cycle applied to the multiplier network, producing a change in state.

The system behavior can now be observed as a function of the magnitude of the control, U_0, and the time duration between samples. For example, see the following figures, 16 and 17. In the first figure the magnitude of U_0 is 1 volt and the perturbation is a step voltage of -1 volt. It appears that with the sample time shown, 3 sec. from one state transition to the next, that the $U_0 = 1$ control is insufficient to bring the output of the process back above $45^{\underline{v}}$. But in the next figure the control voltage has been increased to $U_0 = 2$, while sample time and

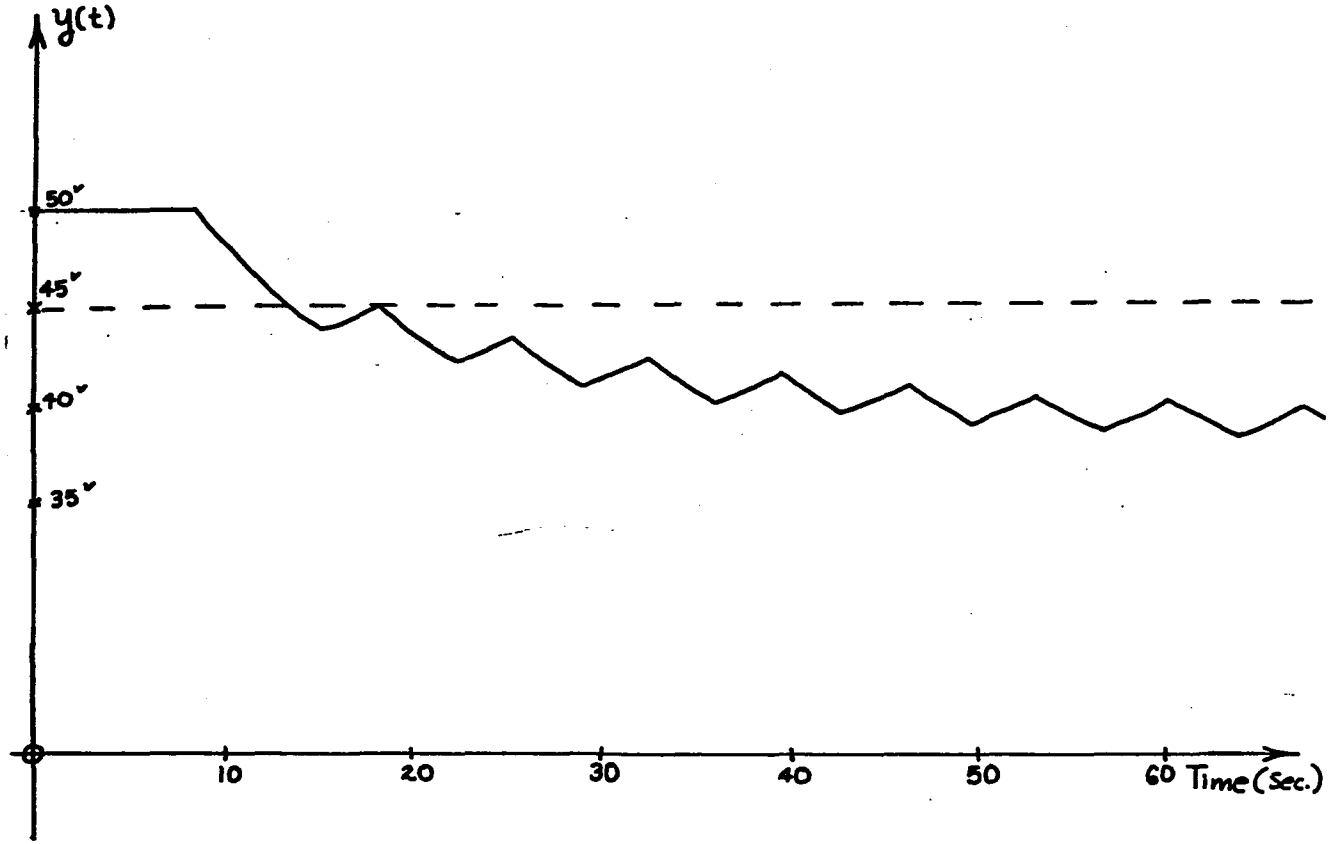


Fig. 16. System Behavior, Output Falling.

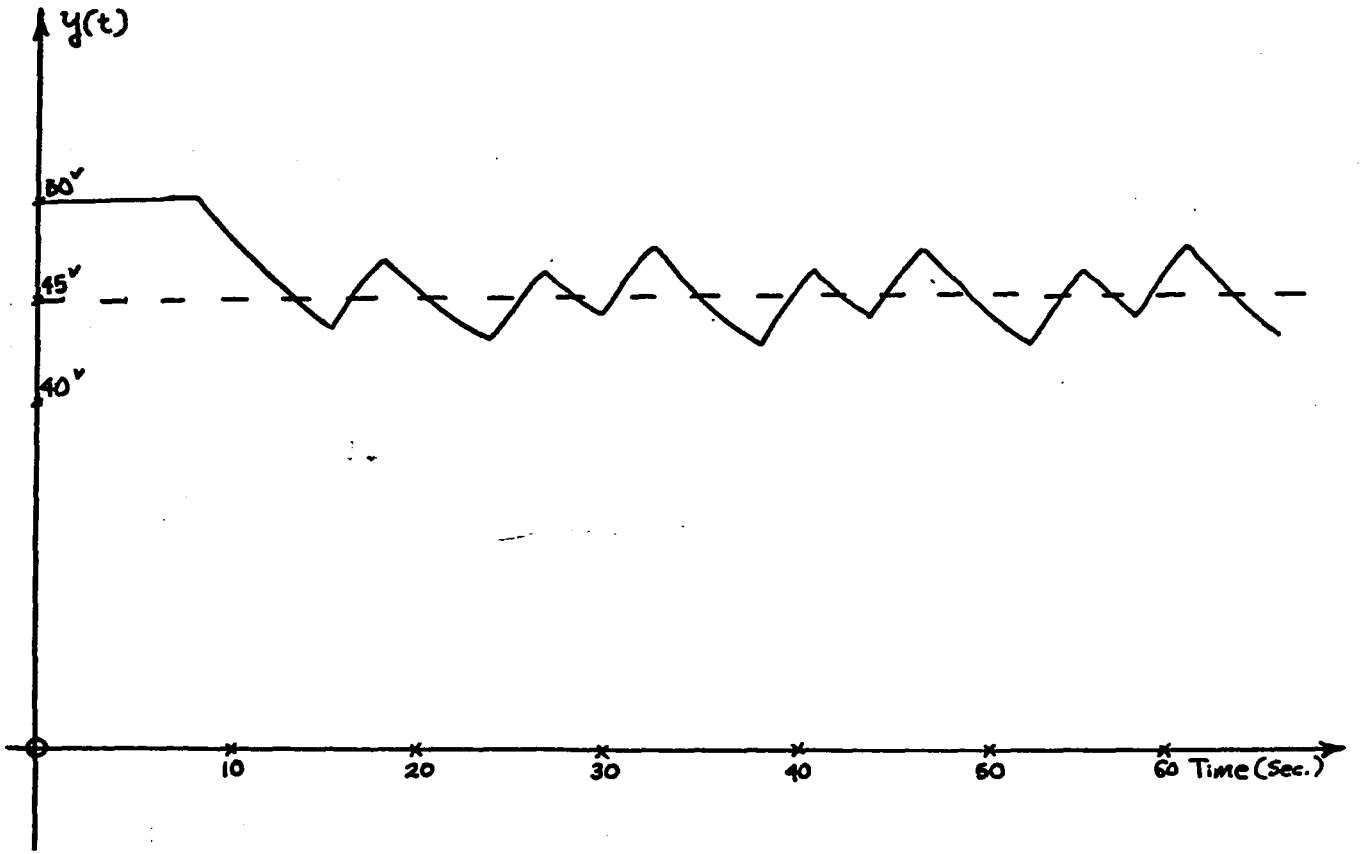


Fig. 17. System Behavior about 45$\frac{V}{-}$.

perturbation remain the same. Now the control is capable of raising the process output above 45^{V}_{-}. The system ranges back and forth about 45^{V}_{-}.

The next graph, Fig. 18, shows a comparison of open and closed loop response to a perturbation consisting again of a -1^{V}_{-} step. Without the loop closed and control being applied the system output falls steadily toward 25^{V}_{-}. With the loop closed and the finite state control being applied the system output ranges back and forth about 45^{V}_{-}. Also plotted on the same graph are the state transitions of the controller, but not on the same scales as the plots of open and closed loop system response. However, the behavior of the state transitions matches the overall behavior as seen in the closed loop response. There are two adjacent control periods separated by a longer period when no control is present. Inspection of the closed loop response reveals the same behavior.

Suppose now the desire existed to design the controller to produce the minimum variations about the lower limit of 45^{V}_{-}. Assuming the same state transitions (the structure of the controller is not to be changed) are to be used, then the two parameters which may be varied are the control U_0 and the sample time, the time between state transitions.

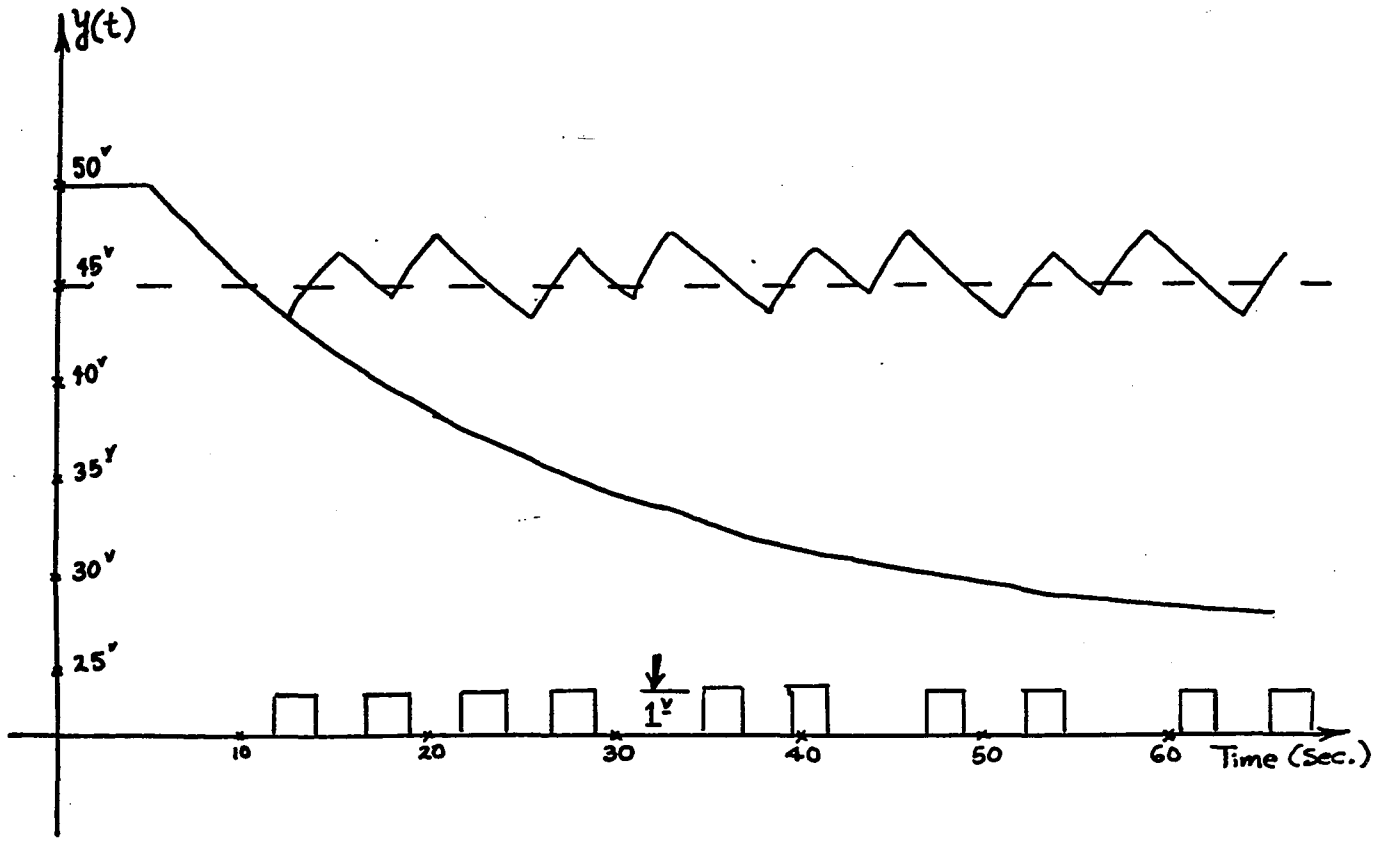


Fig. 18. Open and Closed Loop Response

Intuitively, it is desired to minimize the variations about $45\underline{V}$, the lower control limit. Observation of the system behavior, though, reveals that an insufficient control may cause the system to always be below $45\underline{V}$. Also it would be wise to allow time to elapse before looking at the system behavior, thus allowing the system to settle into a steady-state behavior. So the following index will be minimized:

$$V_R = V_1 + V_2,$$

where: V_1 = 45 -(minimum value less than $45\underline{V}$, 50 seconds after the application a step function perturbation),

V_2 = (maximum value greater than $45\underline{V}$, 50 seconds after the application of a step function perturbation. If no maximum greater than 45, then let V_{max} = 10) -45. The assignment of V_{max} = 10 to V_2 if no maximum above $45\underline{V}$ is an arbitrary penalty placed on the system's failure to come back into control.

Now it is necessary to obtain data on the behavior of the system for various values of control, U_0, and the sample time between state transitions. Using the analog computer simulation of the system this is easily done. The following values of U_0 were selected: U_0 = +1, +2, $+3\underline{V}$. Sampling times of 1 sec., 3 sec., and 5 sec. were also chosen. At each sampling time the three values of control

were used. A perturbation consisting of a -1 V step was applied and the output plotted, as before on an x-y recorder. The following three figures show the output for each different combination of sample and control. Each graph is one sample time for the three different controls. The scale is shifted down for different controls to try to minimize the confusion of three graphs intersecting. (See Figures 19, 20, 21).

From these graphs the value of V_R as a function of sample time and control can be determined parametrically. A set of curves can be plotted, as in Fig. 22. The results of the curves is conclusive in establishing $U_0 = 2$ as the best control value, no matter what the sample time. The absolute minimum of V_R occurs for $U_0 = 2$ and sample time = 1 second. Another set of curves, with the opposite parameter held constant verifies the result. The first figure shows V_R versus sample time for U_0 a parameter. The second figure shows V_R versus U_0 with sample time as a parameter. Thw two figures together give an idea of the overall shape of the surface formed by V_R, U_0, and sample time. It is interesting to observe that for $U_0 = 2$, the plot of V_R versus sample time displays a linear relationship with V_R going to zero as the sample time goes to zero! The control $U_0 = 3$ displays this behavior for sample times

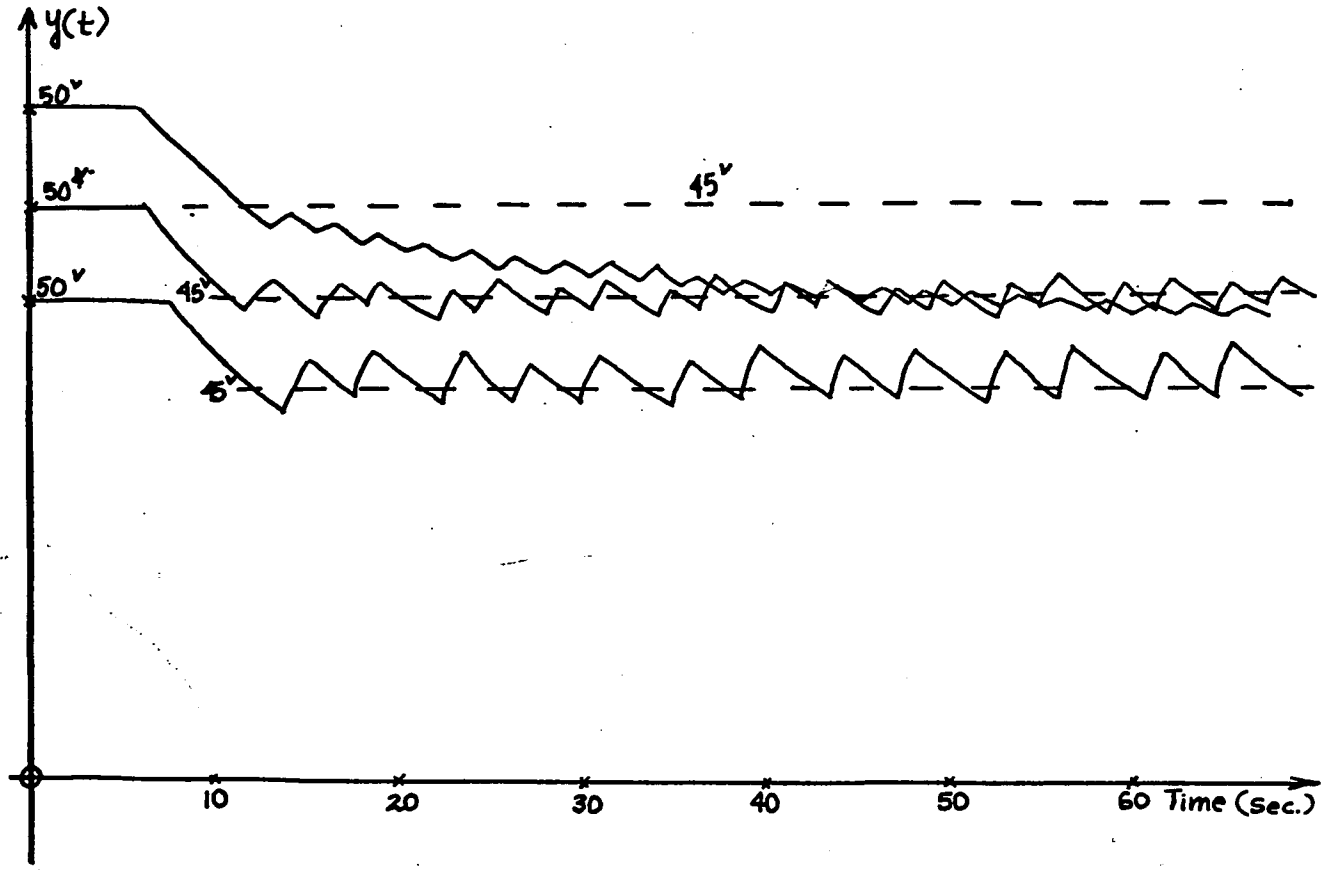


Fig. 19. 1 sec. Sample Time.

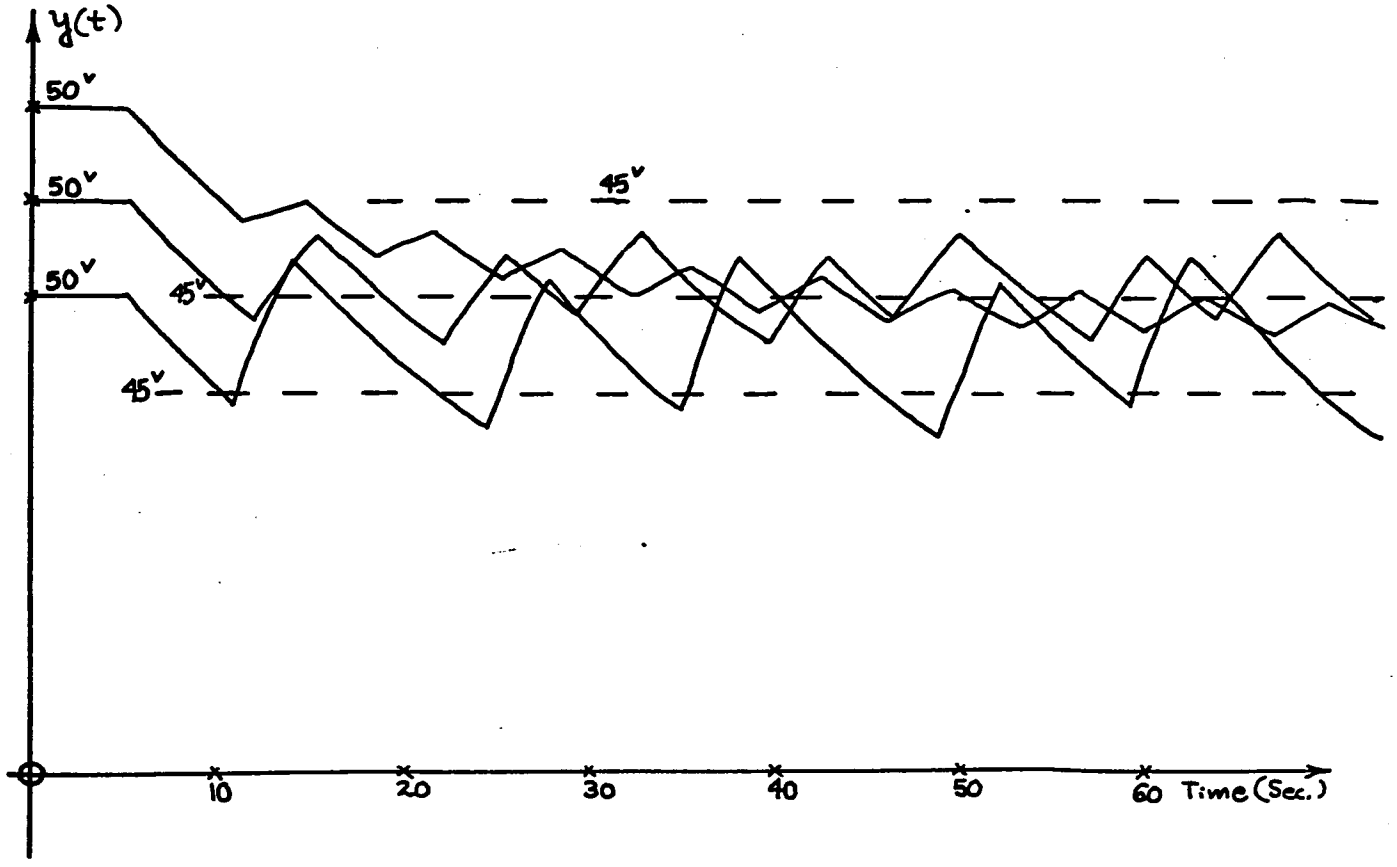


Fig. 20. 3 sec. Sample Time.

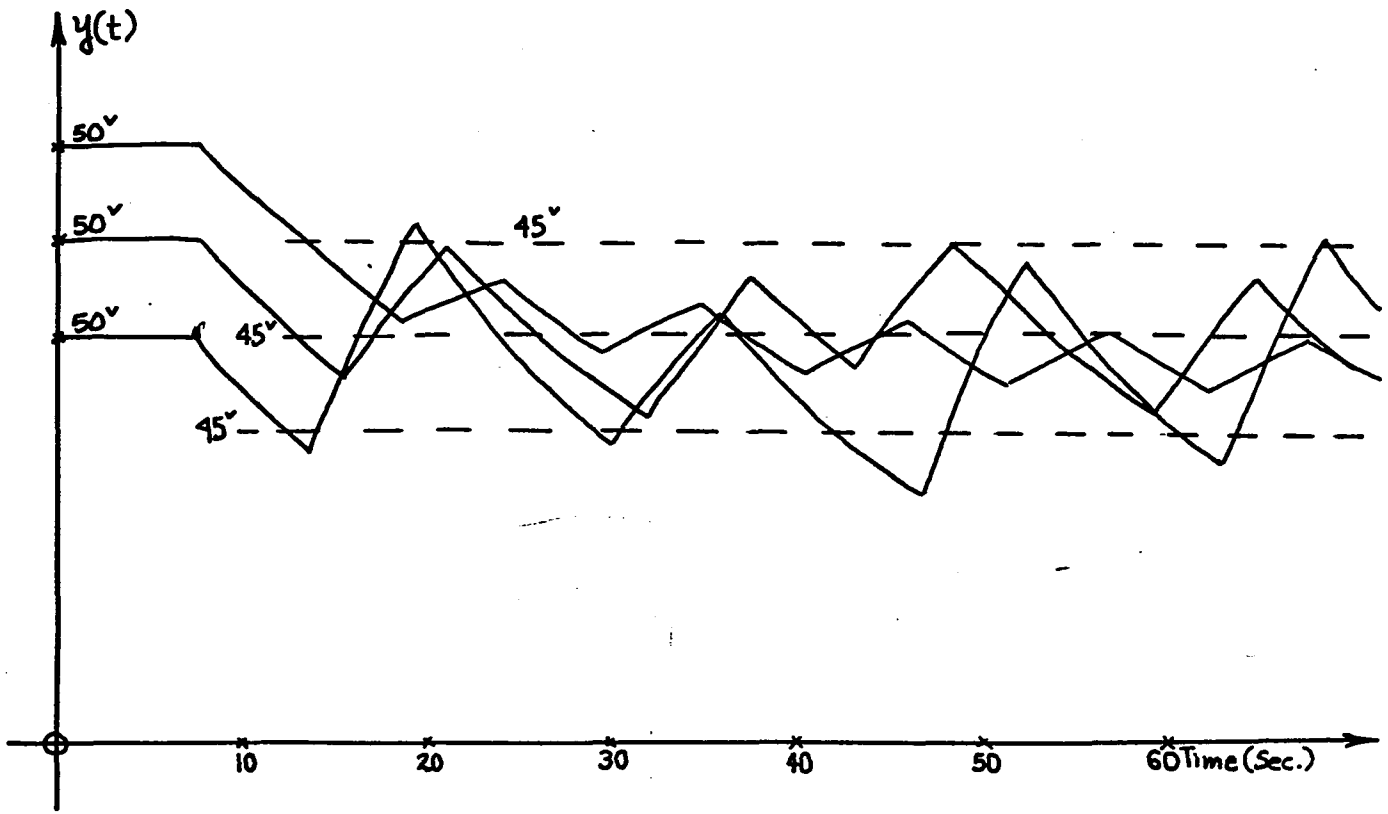


Fig. 21. 5 sec. Sample Time.

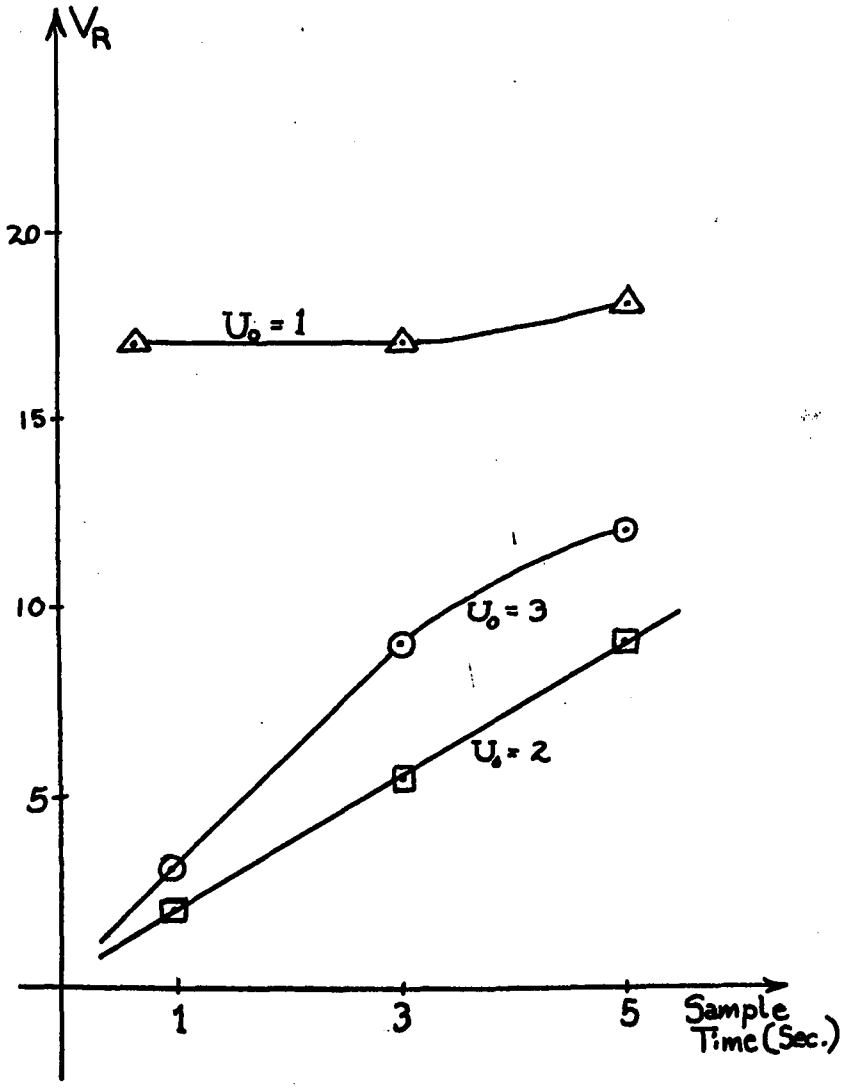


Fig. 22. Index V_R versus Sample Time for Different U_0.

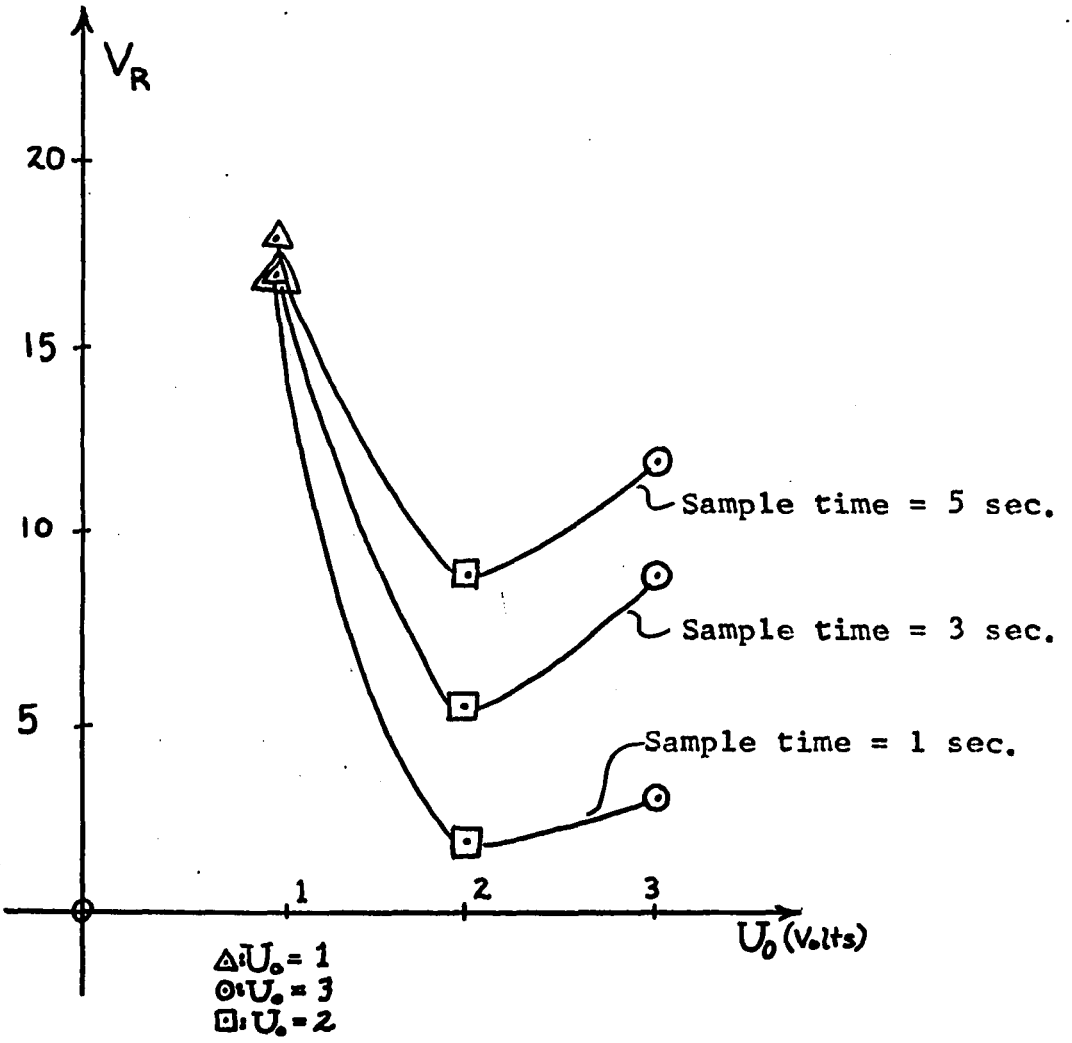


Fig. 23. Index V_R versus U_0 for Various Sample Times.

of 1 and 3 seconds. This is not surprising behavior, but is interesting to see nonetheless.

Finally, it may be of interest to observe the way the system responds in starting from zero and rising to steady-state. This is shown in Fig. 24 for the case of no control, the control yielding minimum V_R, and the control yielding maximum V_R. For maximum V_R the time to rise is a minimum. However, the response in rising for the minimum V_R control is appreciably better than no control and almost as good as the fastest rise time with the other control.

The simple parametric study in this appendix shows the possibility of using analog simulation of finite state and continuous state systems to design systems with different kinds of components. At least from the viewpoint of step function perturbation, the value of $U_0 = 2$ and sample time of 1 second is the best design for the simple controller presented here. A truly serious study would involve much, much more in the way of computer runs and different inputs. This example illustrates the rudiments of the process, though.

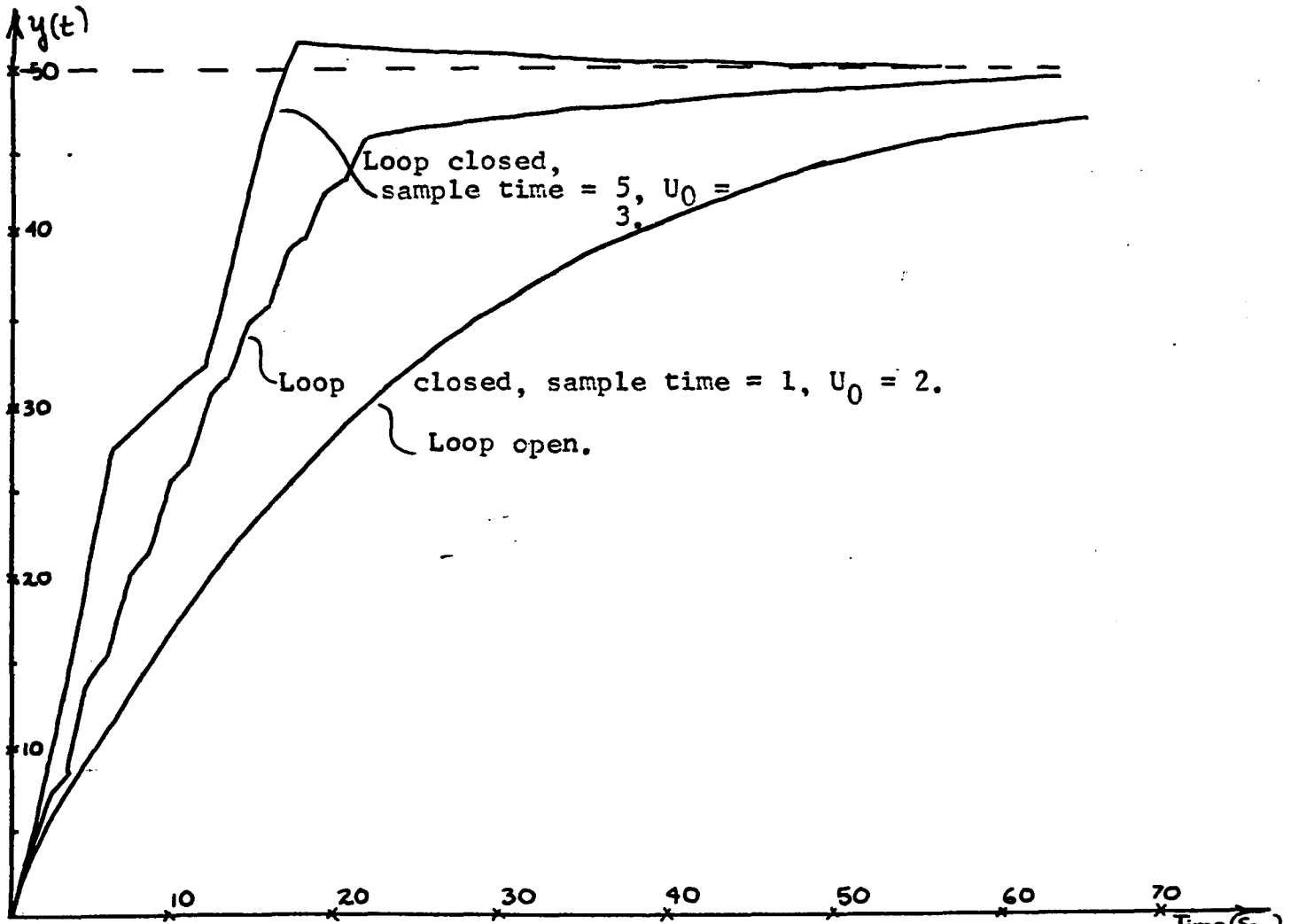


Fig. 24. Response in Rising to Steady-State.

APPENDIX II

Except for the last example, the illustrative computations carried out in Chapter 4 served only to demonstrate how the necessary equations for the coefficients of a real number representation can be generated, and how the formal manipulations for the analysis and synthesis procedures are to be carried out. The examples were chosen as the simplest possible in order to illustrate the computational process free of possible confusion from other factors. As a result, they are not representative of the type of applications for which real number representations can be employed. The purpose of this appendix is to present an example (admittedly still simple) to show how finite state systems may be utilized in the analysis of a given system, and how the real number representations can be conveniently employed in both the modeling and analysis.

The example to be modeled and analyzed is purely hypothetical, the production scheduling system of the Ajax Widget Corporation. The production scheduling system's function is to examine every month the sales and inventories of widgets and then make the decision as to whether or not to increase widget production, decrease widget

production, or leave it at the same level. Ajax hires the operations research firm of Jones and Forsyth to study the production scheduling system. Upon arriving at the factory the O.R. analysts from Jones and Forsyth are presented with company records and procedures which disclose the following information:

(1.) Widgets are a product for which a stable market exists. In times of normal economic activity the long term average sale of widgets is 10,000 units per month, with random fluctuations above and below this average.

(2.) Even though the sales of widgets fluctuate about 10,000 units per month, experience has tended to make the management of Ajax feel that there is little need for a change in production unless sales rise or fall more than 1000 units per month from this average. When sales fall 1000 units per month below the average, the back-log in on-the-shelf inventories becomes great enough to justify a production cut-back. Similarly the rise in sales of more than 1000 means sufficiently increased sales to justify the extra effort necessary for production increases.

(3.) An increase in sales of 1000 units does not bring an automatic increase of 1000 units. Again, experience has tended to make management feel that in the long run it is more economical to over-produce in times of increased sales, for then the economy of larger production makes it cheaper

to build up inventories when the sales of widgets begin to fall. But Ajax is a small company, geared to the stable market of 10,000 widgets per month. The maximum increase in production possible by allotment of overtime to workers is only 1500 units per month. Any further increases would require capital investment and more employees. Desiring to maintain its conservative posture, Ajax has set 1500 units per month as the limit on production increases.

(4.) The production scheduling organization is composed of 3 departments: Production Forecast, Scheduling 1, and Scheduling 2. Production Forecast's function is to set the production level for the coming month based on the sales and forecast for the previous month. Long experience has evolved the following set of policies for the forecasting group: If the previous month's forecast was normal production, then the coming month's forecast will be normal production, decrease production by 1000 units, or increase production by 1500 units, depending upon whether sales in the previous month were normal, 1000 units below average, or 1000 units above average, respectively. If the previous month's forecast was decreased production, then the coming month's forecast will be normal production, decreased production, or increased production, depending upon whether sales in the previous month were normal, below average by 1000 units, or above average by 1000 units. Finally, if

the previous month's forecast was increased production, then the coming month's forecast will be normal production or increased production, depending upon whether the previous month's sales were below or normal, or above normal. Note that if the previous forecast was 1500 extra units and sales went 1000 units below the average, production does not go directly to a forecast level of 1000 units below average. Instead the forecast goes to normal (10,000 units per month) production and the inventories are built up in the process.

The production scheduling departments have to translate the increases and decreases into actual job assignments, material procurements, overtime, etc. Department 2, in charge of scheduling decreases, has a fairly simple task. Its functionings can be described in terms of the previous level of activities and the previous forecast. If it was previously engaged in normal production, then it changes its activities only if the forecast is for a decrease of 1000 units in production. If normal production was engaged in and normal production is forecast then it continues with its activities of helping Department 1 coordinate normal production. If normal production was under way and 1500 extra units are forecast, it continues with ordinary functions and lets Department 1 worry about the increase. Similarly, if Department 2 was coordinating below normal

production, it will continue to do so only if the forecast is for more below normal production. Any other forecast causes the department to return to normal activities.

Department 1 has more problems, however. Jones and Forsyth discover that it requires more effort to immediately raise production to an extra 1500 units than would be thought. Material must be expensively procurred on a rush basis, or a very large warehouse maintained to keep enough material on hand. Job assignments must be made, operations scheduled, etc. Over the years, management has evolved the following compromise. If an extra 1500 units is called for, a majority of the staff of Department 1 goes to work to raise the production immediately by 1000 extra units, a level which is not too difficult to reach. Then in the next month, if the forecast still calls for 1500 extra units, the rest of Department 1 pitches in to the task of coordinating the next 500 units of production. Thus, the attainment of maximum production occurs in two steps. Production stays at 1500 units until the forecast indicates a change. If production is at the extra 1000 unit level and the forecast does not call for 1500 units, but below normal or normal production, then the activity goes back to the normal level. Department 2 assumes the responsibility of coordinating any decrease called for.

Thus, the scheduling in Department 1 takes place in a more complex fashion than Department 2. Even though a forecast for maximum production, 1500 extra units, may be called for, the necessities of plant operation require two steps to this level. To simplify the forecasting process, this intermediate step is not made a part of the forecast: either an extra 1500 units is forecast or not at all. Experience tends to show that this, within the constraints of the company, is the production level to aim for, so no further nuances are really called for in the process of preparing the next month's forecast.

These are the chief functions of the production scheduling system presented to Jones and Forsyth by Ajax. Ajax has decided that it is concerned about the production scheduling function, and thereby asks Jones and Forsyth to create a mathematical model of the system and then employ simulation to answer the following questions:

(1.) With the fixed warehouse and inventory policies of Ajax, how successful is the production scheduling system in preventing excessive inventory depletion or excessive overstocking?

(2.) Under what combinations of circumstances does excessive depletion or overstocking arise using the production scheduling system as outlined?

(3.) Can a change in inventory and warehouse policies offset excessive depletion or overstocking without changes in the production scheduling system?

The first step in deriving the model is to model each of the three departments as a finite state system. The important fact about all the departments is that actions take place in terms of quantities above or below normal. Thus, the states and inputs for the systems can be translated into units above and below the average of 10,000 units per month. Thus, when the forecast from the Production Forecast Department is normal production, this is describable by a state of zero additional units forecast. Similar arguments hold for other states, as well as inputs.

The production forecast process described in (4.) above can be modeled by the following finite state system:

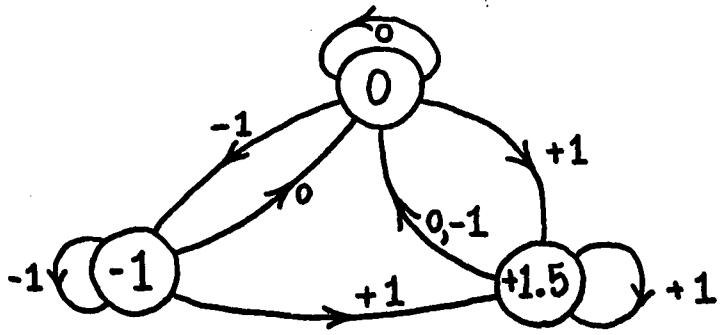


Fig. 25. Production Forecast Department.

The states and inputs have the following meanings.

Inputs: 0 -- normal sales, +1 -- sales 1 thousand units

above normal, -1 -- sales 1 thousand units below normal. States: 0 -- normal production forecast, +1.5 -- additional 1.5 thousand units forecast, -1 -- decrease production by 1 thousand units forecast.

The states and inputs being signed real numbers, the real number representation is a very logical representation to be employed for this system. By the techniques illustrated in Chapter 4, the following polynomial can be derived to describe the state transitions of this system:

$$pf_1'(s_1,i_1) = \frac{5}{4}i_1 + \frac{1}{4}i_1^2 - \frac{2}{15}i_1s_1 + \frac{2}{15}i_1^2s_1 - \frac{2}{15}i_1s_1^2 + \frac{2}{15}i_1^2s_1^2$$

$$= \frac{i_1}{4}(i_1+5) + \frac{2}{15}i_1s_1(i_1-1) + \frac{2}{15}i_1s_1^2(i_1-1).$$

Department 1 can be modeled with a finite state system:

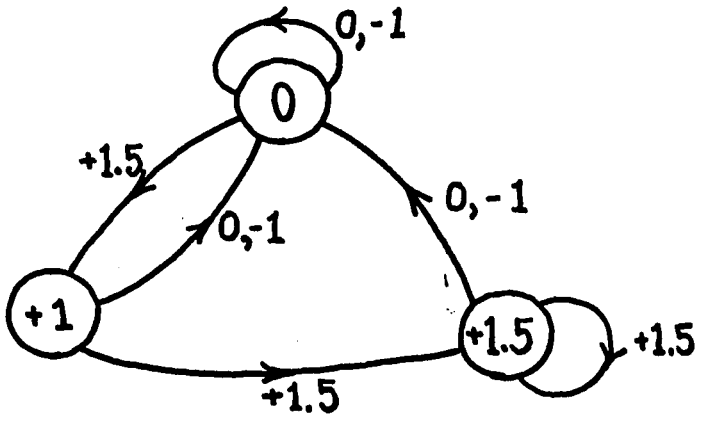


Fig. 26. Department 1.

The states and inputs have the following meanings. Inputs: 0 -- normal production forecast, +1.5 -- additional 1.5 thousand units forecast, -1 -- decrease production by 1 thousand units forecast. States: 0 -- normal activities, +1 -- increase production by 1 thousand units, +1.5 -- increase production by 1.5 thousand units.

Once again a real number representation may be found for this system. Using the methods illustrated in Chapter 4, the following polynomial can be derived to describe the state transitions of this system.

$$pf_2(s_2,i_2) = \frac{4}{15}i_2 + \frac{4}{15}i_2^2 + \frac{44}{90}i_2s_2 - \frac{8}{45}i_2^2s_2 - \frac{44}{90}i_2s_2^2 + \frac{8}{45}i_2^2s_2^2$$

$$= \frac{4}{15}i_2(1+i_2) + \frac{44}{90}i_2s_2(1-s_2) - \frac{8}{45}i_2^2s_2(1-s_2)$$

Finally, the second scheduling department can be modeled as:

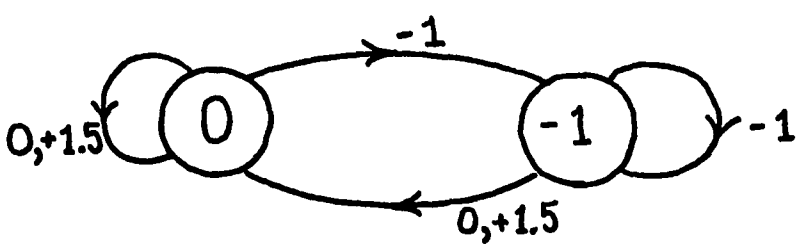


Fig. 27. Department 2.

The states and inputs have the following meanings. Inputs: 0 -- normal production forecast, +1.5 -- additional 1.5 thousand units forecast, -1 -- decrease production

by 1 thousand units forecast. States: 0 -- normal activities, -1 -- decrease production by 1 thousand units.

The state transition polynomial for this system takes the form:

$$pf_3(s_3, i_3) = \frac{3}{5}i_3 - \frac{2}{5}i_3^2.$$

It is now possible to describe the whole production scheduling system by the block diagram:

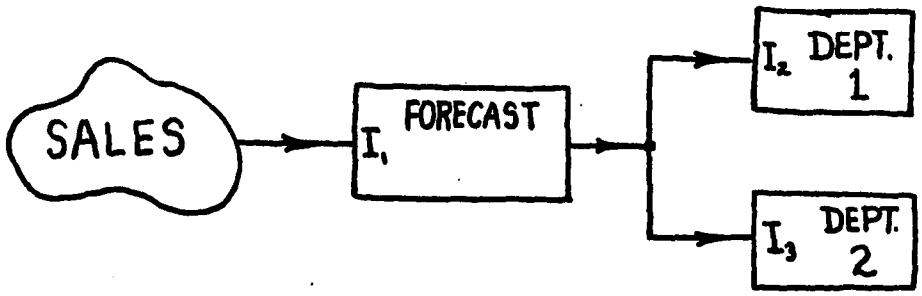


Fig. 28. Production Scheduling System.

The total system can be modeled by the following interconnection:

$$K = (M_1, M_2, M_3, (I), (M_1, I_2), (M_1, I_3)).$$

Of prime interest to the analysts at Jones and Forsyth, though, is the study of the whole system behavior. Thus the resultant of this interconnection is formed. Using the polynomials described above plus the interconnection K, the resultant may be immediately arrived at as:

$$RES(K) = (S_1 \times S_2 \times S_3, I_1, S_1 \times S_2 \times S_3', \underset{\sim}{f}', \underset{\sim}{w}),$$

where $\underset{\sim}{f}'$ is such that:

$\underset{\sim}{f}' = \text{res}(p\underset{\sim}{f}', S_1 \times S_2 \times S_3 \times I_1)$, and $p\underset{\sim}{f}'$ is a polynomial function:

$$p\underset{\sim}{f}'(s_1,s_2,s_3,i_1) = \begin{bmatrix} \frac{i_1}{4}(i_1+5) + \frac{2i_1}{15}s_1(i_1-1) + \frac{2i_1}{15}s_1^2(i_1-1) \\ \frac{4}{15}s_1(1+s_1) + \frac{44}{90}s_1s_2(1-s_2) - \frac{8}{45}s_1^2s_2(1-s_2) \\ \frac{3}{5}s_1 - \frac{2}{5}s_1^2 \end{bmatrix}$$

and $\underset{\sim}{w}$ is the identity function.

Using this resultant, which is the model of the behavior of the entire production scheduling system, Jones and Forsyth can now test Ajax's policies by extensive computer simulation, using sales data gleaned from previous business of the Ajax Widget Corporation, and thus answer the original questions put to them about inventory policies, etc.

The previous example illustrates the usefulness of finite state systems in the analysis of general systems. Many general systems have a behavior whose function is describable (or approximated, at least) by a finite set of states. The practicality of applying such models is apparent if one considers the following: in a very complex system, the number of possible combinations of states in the component systems is very large. There may exist possible combinations of states and inputs that could drive

the system into a set of persistent states, thus causing it to cease functioning in the larger sense for which it was intended. Such possibilities can be studied by forming the resultant of the components and examining its behavior.

Real number representations prove useful in general systems studies, first because the states and inputs can be associated so naturally with the actual quantities in the system (as was done in the previous example) and second because the construction of the resultant is such a simple process. If Simon's method were employed, it would be necessary to analyze a general system by a large number of steps. That is, two systems would have to be combined into a new system, this new system combined with another system, etc. The real number representation allows the whole analysis to be completed in one step.

It is desirable to also note the role that synthesis can play in synthesizing more general systems. Suppose that the actual structure of the production scheduling systems was not fixed, but instead a set of overall policies was given specifying how the total system was to function in terms of sales inputs, forecasts, current coordination of factory activity, etc. Then by modelling this set of overall policies by a finite state system a production scheduling system could be synthesized by deciding upon the number of states in the component systems, what those states would

be, and then applying the synthesis procedure. The synthesis would then detail a set of systems whose behavior yielded the overall behavior. The final step would be to implement these component systems, simpler certainly than trying to implement the total system.

REFERENCES

(1.) Ashby, W. R., Introduction to Cybernetics, Science Editions, John Wiley & Sons, New York, 1963.

(2.) Bekey, G. A. and D. L. Gerlough, "Simulation", in System Engineering Handbook ed. by R. Machol, McGraw-Hill Co., New York, 1965.

(3.) Bellman, R. E., Adaptive Control Processes, Princeton Press, Princeton, N. J., 1961.

(4.) Derusso, P. M. with R. J. Roy and C. M. Close, State Variables for Engineerings, John Wiley & Sons, New York, 1965.

(5.) Ellis, D. O. and F. J. Ludwig, Systems Philosophy, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1962.

(6.) Feller, W., An Introduction to Probability Theory and Its Applications, John Wiley & Sons, New York, 1950.

(7.) Friedland, B., "Linear Modular Sequential Circuits", IRE Trans., Vol. CT-6, No. 1, 1959, pg's 61-68.

(8.) Friedland, B. and T. E. Stern, "The Linear Modular Sequential Circuit Generalized", IRE Trans., Vol. CT-8, No. 1, 1961, pg's 79-80.

(9.) Gill, A., Introduction to the Theory of Finite State Machines, McGraw-Hill Co., New York, 1962.

(10.) _____, "Cascaded Finite State Machines", IRE Trans., Vol. EC-10, No. 3, 1961, pg's 366-370.

(11.) Haring, D. R., Sequential Circuit Synthesis -- State Assignment Aspects, Monongraph #31, MIT Press, Cambridge, Mass., 1966.

(12.) Hartmanis, J., "Loop-Free Structure of Finite State Machines," Information and Control, Vol. 5, Aug. 1962, pg's 25-43.

(13.) _____, "Symbolic Analysis of a Decomposition of Information Processing Machines", Information and Control, Vol. 3, June 1960, pg's 154-178.

(14.) Hartmanis, J. and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice-Hall Inc., Englewood Cliffs, N. J., 1966.

(15.) Hoffman, K. and R. Kunze, Linear Algebra, Prentice-Hall Inc., Englewood Cliffs, N. J., 1961.

(16.) Howard, R. A., Dynamic Programming and Markov Processes, MIT Press, Cambridge, Mass., 1960.

(17.) Kautz, W. H. editor, Linear Sequential Switching Circuits, Holden-Day, San Francisco, 1965.

(18.) Korn, G. A., "Electronic Analog/Hybrid Computers and their use in System Engineering", in System Engineering Handbook, R. Machol editor, McGraw-Hill Co., New York, 1964.

(19.) Linvill, W. K., "System Theory as a Extension of Circuit Theory", IRE Trans. Vol. CT-3, No. 4, 1956, pg's 217-223.

(20.) Pask, G., An Approach to Cybernetics, Harper and Bros., New York, 1961.

(21.) Simon, J. M., "Some Aspects of the Network Analysis of Sequence Transducers", J. Franklin Institute, Vol. 265, June 1958, pg's 439-450.

(22.) Tobey, R. G., "Experience with Formac Algorithm Design", Comm. ACM, Vol. 9, No. 8, 1966, pg's 589-597.

(23.) Webster's Dictionary, Peerless Edition, World Syndicate Publishing Co., Cleveland, Ohio, 1939.

(24.) Wymore, A. W., A Mathematical Theory of Systems Engineering, To be Published by John Wiley & Sons.

(25.) Yau, S. S. S. and K. C. Wang, "Linearity of Sequential Machines", I.E.E.E. Trans., Vol. EC-15, No. 3, 1966, pg's 337-354.

(26.) Yoeli, M., "The Cascade Decomposition of Sequential Machines", IRE Trans., Vol. EC-10, No. 4, 1961, pg's 587-592.

(27.) ————, "Cascade-Parallel Decompositions of Sequential Machines", IRE Trans., Vol. EC-12, No. 3, 1963, pg's 322-324.

(28.) Zadeh, L., "From Circuit Theory to Systems Theory", Proceedings IRE, Vol. 50, No. 5, 1962, pg's 856-865.