INFORMATION TO USERS

This was produced from a copy of a document sent to us for microfilming. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help you understand markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure you of complete continuity.

2. When an image on the film is obliterated with a round black mark it is an indication that the film inspector noticed either blurred copy because of movement during exposure, or duplicate copy. Unless we meant to delete copyrighted materials that should not have been filmed, you will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed the photographer has followed a definite method in “sectioning” the material. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For any illustrations that cannot be reproduced satisfactorily by xerography, photographic prints can be purchased at additional cost and tipped into your xerographic copy. Requests can be made to our Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases we have filmed the best available copy.
FUNK, KENNETH HARDING, II

TERMS AND AXIOMS FOR A THEORY OF HUMAN-MACHINE SYSTEMS

The Ohio State University

Ph.D. 1980

University Microfilms International

300 N. Zeeb Road, Ann Arbor, MI 48106
TERMS AND AXIOMS
FOR A THEORY OF HUMAN-MACHINE SYSTEMS

A DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By


* * * * *

The Ohio State University

1980

Reading Committee:

Richard A. Miller
Richard J. Jagacinski
George L. Smith, Jr.

Approved By

R A Mill
Adviser
Department of Industrial and Systems Engineering
This dissertation is dedicated to my parents, Kenneth and Rachel Funk. They contributed more to it than they will ever know.
ACKNOWLEDGMENTS

I wish to thank my adviser, Dr. R. A. Miller, for his interest in my efforts and for his very active and very beneficial participation in the process of my graduate education. My other reading committee members, Dr. Richard Jagacinski and Dr. George Smith, are also deserving of my thanks for their assistance in the preparation for and the production of this dissertation. Finally, I wish to gratefully acknowledge the U. S. Air Force Aerospace Medical Research Laboratory for funding and monitoring the research project from which this work emerged.
VITA

7 January 1953 Born - Ashland, Ohio

May 1975 B.A. Taylor University, Upland, Indiana

1976-1980 Administrative Assistant, Department of Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio

1977 M.Sc., The Ohio State University, Columbus, Ohio

1978-1980 Research Assistant, Department of Industrial and Systems Engineering, The Ohio State University, Columbus, Ohio

PAPERS


FIELDS OF STUDY

Major Field: Industrial and Systems Engineering

Studies in Systems Theory. Professor Richard A. Miller

Studies in Man-Machine Systems. Professor Richard A. Miller

Studies in Computer Applications in Industrial and Systems Engineering. Professor Richard A. Miller

Studies in Operations Research. Professor Gordon M. Clark
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>VITA</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>0 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>0.1 Structures and Knowledge</td>
<td>2</td>
</tr>
<tr>
<td>0.1 Objectives</td>
<td>4</td>
</tr>
<tr>
<td>0.3 Overview</td>
<td>5</td>
</tr>
<tr>
<td>1 Languages, Theories, Models, and Formalization</td>
<td>7</td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Languages</td>
<td>8</td>
</tr>
<tr>
<td>1.3 Theories</td>
<td>10</td>
</tr>
<tr>
<td>1.4 Models</td>
<td>12</td>
</tr>
<tr>
<td>1.5 Formalization</td>
<td>17</td>
</tr>
<tr>
<td>1.6 Summary</td>
<td>19</td>
</tr>
<tr>
<td>2 Human-Machine Systems</td>
<td>20</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>20</td>
</tr>
<tr>
<td>2.2 Definition of a Human-Machine System</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Examples of Human-Machine Systems</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Human-Machine System Issues</td>
<td>26</td>
</tr>
<tr>
<td>2.5 Existing Human-Machine System Theories</td>
<td>28</td>
</tr>
<tr>
<td>2.6 Shortcomings of Existing Theories</td>
<td>31</td>
</tr>
<tr>
<td>2.7 Summary</td>
<td>33</td>
</tr>
<tr>
<td>3 Formal Development of a Theory of Human-Machine Systems</td>
<td>34</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>34</td>
</tr>
<tr>
<td>3.2 Preliminaries</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Subject Domain</td>
<td>37</td>
</tr>
</tbody>
</table>
3.4 Observation Terms ..................................................... 37
3.5 Primitives ................................................................. 38
3.6 Correspondence Rules .............................................. 39
3.7 Defined Terms ........................................................... 39
3.8 Axioms ...................................................................... 72
3.9 Logical Calculus ...................................................... 78
3.10 Discussion ............................................................... 79
3.11 Summary ................................................................. 80

4 An Interpretation of the Terms and Axioms .... 82
  4.1 A Bicycle and Rider ................................................ 83
  4.2 The Bicycle and Rider as a System, S ............... 84
  4.3 A Decomposition of S ............................................. 88
  4.4 The Bicycle and Rider as a Human–Machine system .......... 97
  4.5 Structures for Organizing Knowledge About Bicycles and Riders ...... 139
  4.6 Summary ................................................................. 140

5 Some Implications Raised by the Structure .... 141
  5.1 Issues Raised by the Structure ...................... 141
  5.2 Measures, Performance, and Workload ........ 157
  5.3 Summary ................................................................. 168

6 Summary, Discussion, and Conclusions .......... 170

LIST OF REFERENCES ...................................................... 176
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 data</td>
<td>85</td>
</tr>
<tr>
<td>2 s</td>
<td>89</td>
</tr>
<tr>
<td>3 rr</td>
<td>94</td>
</tr>
<tr>
<td>4 bb</td>
<td>96</td>
</tr>
<tr>
<td>5 pe - State</td>
<td>126</td>
</tr>
<tr>
<td>6 pe - Inputs</td>
<td>127</td>
</tr>
<tr>
<td>7 pe - Outputs</td>
<td>128</td>
</tr>
<tr>
<td>8 m</td>
<td>129</td>
</tr>
<tr>
<td>9 pp</td>
<td>130</td>
</tr>
<tr>
<td>10 pg</td>
<td>131</td>
</tr>
<tr>
<td>11 pc</td>
<td>132</td>
</tr>
<tr>
<td>12 mxp</td>
<td>133</td>
</tr>
<tr>
<td>13 p</td>
<td>134</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time Line For BR</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Cascade Connection</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>Parallel Connection</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>Feedback Connection</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>S</td>
<td>86</td>
</tr>
<tr>
<td>6</td>
<td>Decomposition of S</td>
<td>90</td>
</tr>
<tr>
<td>7</td>
<td>Procedural Description of S</td>
<td>101</td>
</tr>
<tr>
<td>8</td>
<td>Procedure Profile of S</td>
<td>135</td>
</tr>
</tbody>
</table>
CHAPTER 0

Introduction

As our desires to travel faster and farther, to work at greater depths, to deal with more complex distribution and routing problems, and to provide better, more effective military systems grow, so does our need to better understand how to construct, operate, and maintain complex systems of humans and machines. The analysis and design of such human-machine systems pose a great number of challenges to us. At one time, these systems were constrained primarily by the nature of the machines they contained, for our technology addressed mainly the mechanical aspects of things. In these early human-machine systems the human operators were servants to the machines and conformed mostly to their needs and limitations.

Automation technology has progressed to the point now, though, that the boring, repetitious activities once performed by humans, as servants of the machines, have largely been taken over by the machines themselves. This, in part has led us to the production of highly complex human-machine systems in which new problems of analysis and design face us. These problems lie to a large extent in complexity and in the required coordination of human and machine activities. The tremendous opportunities that technology has brought to us come associated with questions of how to produce systems in which machines do what they are best suited for, humans do what they are best suited for, and both perform their activities in a consistent, predictable, and efficient manner.

The study of human-machine systems, then, has become a rather important area, for herein lies a key to our ability to deal with environments of growing complexity. Many government and private organizations spend large amounts of money investigating the issues surrounding the design, production, and operation of aircraft systems, air traffic control systems, electric power generation and distribution control systems, and weapon delivery systems. A large body of literature in the fields of psychology and engineering
has emerged to address these issues.

But what is a human-machine system anyway? Although this will be covered in more detail in Chapters 2 and 3, for our purposes now we can regard a human-machine system as an interrelated collection of humans and machines which function together to achieve some goal or set of goals.

It is important to note that even though we can point to particular examples of human-machine systems, the concept itself is just a convenient way to organize our thinking about a particular class of objects and phenomena. David Meister, in the introduction to Behavioral Foundations of System Development (Meister, 1976), calls a human-machine system "...an abstraction, not a physical configuration or a type of organization. It is primarily a framework for analyzing systems." It seems appropriate, then, to regard all of the concepts and issues relating to human-machine systems as components of structures that help us understand and pose questions about a subject domain we call human-machine systems. The value of these structures lies in their usefulness to us in designing and operating systems in such a manner as to help us in achieving personal and societal goals.

0.1 Structures and Knowledge

We can informally define a structure as a collection of things along with some relationships among those things. For example, a globe is a structure consisting in part of the points on its surface. The spatial relationships between those points are very important for they can tell or remind us of important political and geographical relationships in the real world. Since the globe is not considered to be an end in itself (we study geography, not globology) it assumes symbolic importance. In other words it is a symbolic structure which helps us to organize our knowledge about the world.

More formally speaking, we can define a structure in terms of two sets:

\[ S = (U, R) \]
where $U$ is a set of objects and $R$ is a set of relations. $U$ defines the objects, physical or conceptual, of the structure and $R$ defines the relationships, spatial, temporal, or logical, among those objects.

Like our globe, a drawing or a photograph is a structure which can be described as a set of points related spatially. A block diagram or a flow diagram is a structure describable in the same manner. A sentence in a language is a structure as is the language itself. Here the objects in the set $U$ include symbols in an alphabet and $R$, the set of relations includes order relation on the alphabets.

We seem to love order or, at least what we call order. We therefore create order by building such symbolic structures as these to organize what we know about the world and about ourselves. The knowledge that we possess, including our own sense data as well as information we acquire through others, provides the basis for the resulting structure. Usually, though, this structure is incomplete, that is there are certain elements of $U$ which do not appear in any relation or do not appear in a particular relation we think they should. Therefore, we say that present knowledge forms the structure and future knowledge, what we may come to know or observe later, adds to the structure and fills in some of the holes.

When we observe or learn of something new to us, we attempt to find how it fits into our structure. If it fits neatly, we are comfortable. On the other hand, political, scientific, and religious revolutions can result when a mismatch is found.

These symbolic structures, therefore, guide our actions as well as organize our thoughts. Scientists spend a lot of time testing the structures that have been postulated by themselves and by others, attempting to see if observations fit the structures and vice versa.

Almost from the moment of birth many of our activities are directed to adding to the symbolic structures we already possess. We expand and revise, based on what we observe and what we learn through the indirect channels of books, television, and other people.

When we find that an existing structure just does not adequately symbolize what we have experienced ourselves, as when Lavoisier noted the contradiction between phlogiston theory and his own observations of combustion, we normally
orient our activities around explaining the discrepancy in terms of our structure or, perhaps as Lavoisier did, around developing a new, more appropriate structure. (For an account of Lavoisier and phlogiston, see Gale, 1979.)

0.2 Objectives

This notion of the use of structures to organize knowledge and actions is the point of departure for this dissertation. Hold that thought in mind while we consider a quotation from a well-known text on human-machine systems.

A model may be effective if it serves only to help the engineer organize his thinking about how people perform, and enables him to distinguish those variables that are likely to be important, or if it helps him to design an ad hoc experiment or simulation to answer specific questions. This function of models is, we believe of considerable importance both pedagogically and in practice. (Sheridan and Ferrell, 1974).

With this motivation, we can state the three major objectives of the dissertation. The first objective is to develop a structure to aid in organizing knowledge about human-machine systems. It is clear that human-machine systems are of some importance to nearly everyone. It is also clear that in order to pose questions and try to answer them, one must have an appropriate vocabulary, an appropriate interpretation of the terms of the vocabulary, and a realistic set of assumptions about the domain of interest. In this dissertation a structure consisting of a set of formally defined terms for describing human-machine systems and human-machine system issues, and a set of axioms or assumptions about human-machine systems will be developed.

The second major objective of the dissertation is to show that the terms and axioms form a structure that leads to important issues that might not have been raised otherwise, or that might not have been raised in such a manner, and to implications for further thought and inquiry.
The third and most important objective of the dissertation is to show through an example (that is, the attempt to satisfy the first two objectives) how we go about developing and using structures to organize knowledge and actions.

If these objectives can be satisfied, contributions will have been made to two major areas. The structure will be a contribution to the field of human-machine system research if it allows us to more precisely describe what characteristics a proposed human-machine system should possess or to pose interesting and helpful research questions. It will be a contribution to science and to the process of systematic inquiry if it illustrates how a particular structure has been developed and used to organize what is already known about some subject domain and how it is used to guide activities aimed at increasing what is known.

0.3 Overview

To accomplish these objectives, an approach somewhat different from that traditionally used by psychologists or engineers will be used. This approach will draw a bit from the philosophy of science, for philosophers and historians of science are perhaps best qualified to tell us of the successes and failures of the many different approaches that have been used in the process of systematic inquiry. It will depend to a certain extent upon formal mathematics and mathematical logic for mathematicians and logicians are experts in the precise and orderly description of often complex concepts and phenomena. It will also rely on the field of computer science for computer scientists and engineers have given us (usually quite unintentionally) rich, yet simple and very systematic, analogs for human and human-machine system behavior. To summarize this approach, a brief overview of the dissertation follows.

Chapter 1 will introduce the concepts of languages, theories, and models, and how they relate to each other. These ideas are extremely important in the process of inquiry and a basic understanding of them is essential in understanding that process. The concept of the formalization of theories will be presented and the advantages of the formalization of empirical theories will be discussed.
Chapter 2 will review definitions and examples of human-machine systems to more clearly describe the subject domain of the current research. Some of the important issues relating to the design and analysis of such systems will be stated and a number of current theories developed to deal with these issues will be reviewed. Some of the shortcomings of these theories will be outlined and a statement of need for a new theory will be made.

The development of the foundations for such a theory of human-machine systems will be presented in Chapter 3. A set-theoretic approach will be used to define some key concepts and relationships and to state some important assumptions that will serve as the axioms of the theory.

One of the requirements of the formalization of a theory is the presentation of models or interpretations of (informally, examples of things consistent with) the theory. Such an interpretation will be presented in Chapter 4. In addition to satisfying the requirements for formalization, it will help to illustrate the ideas presented in the preceding chapter.

A theory can be truly tested only by time. The next best test is to show that it sheds light on its subject domain. In Chapter 5 will be presented a number of issues and implications raised by the terms and axioms.

Chapter 6 will summarize the research and analyze the contributions made by it.
CHAPTER 1

Languages, Theories, Models, and Formalization

1.1 Introduction

Certainly no human activity takes place entirely outside of some meaningful context. Context underlies and shapes these activities and gives them identity. Therefore, to introduce this dissertation it is first necessary to present some of its context.

Too often we as scientists and engineers forget or ignore the fundamental assumptions that have gone into our activities. In doing this we become technicians in the sense that we apply ourselves merely to the tasks at hand, losing site of the overall process in which we are involved. This is certainly not always without merit, for if we questioned and analyzed every act before performing it, we would accomplish very little. On the other hand, it is occasionally useful to pause to gain some perspective to determine if our aims are really reasonable and our methodologies are really consistent with those aims.

The purpose of this chapter is to provide some perspective on the dissertation research and to provide a structure in terms of which the more specific individual activities can be explained. To do this, a particular interpretation of the process of scientific inquiry will be presented, one in which the terms "languages", "theories", and "models" are particularly significant. This interpretation is certainly not the only one possible, for the history of science clearly tells us that such a complex activity does not lend itself well to description from a single perspective. Nevertheless, it is often quite useful to use rather narrow, simplified descriptions to serve as bases from which useful work can proceed.
Therefore, in this chapter, it will be put forward that theories (whether scientific or not) are collections of statements, made in some language about particular subject domains (a subject domain being that thing or group of things about which a theory speaks) and that models are structures in which the statements of these theories are interpreted as true. Finally, it will be argued that formalization, a process by which theories are constructed through the use of models, is particularly useful in dealing with complex realms which present us with substantial problems.

1.2 Languages

We use language primarily to communicate, whether it be simple everyday information or the intermediate or final results of scientific investigation. However, language serves another highly important role, that of organizing and providing structure to what we know. This latter point is made clear in that the language we use determines to a large extent how we think about our experiences and how we go about approaching and solving problems.

Many views may be taken of language, many definitions may be suggested, and many different categories of languages may be identified. Here, however, a rather general definition will be given. A language may be defined as a set of strings of symbols from some alphabet. We normally think of alphabets in terms of written letters or spoken phonemes, but an alphabet may consist of graphic symbols, electronic signals, or the magnetic orientation of bits of iron oxide on a magnetic tape.

Natural languages, such as English, Chinese, or French, are characterized by their informal and gradual evolution over time, and in some cases by the existence of two distinct forms, one written and read and the other spoken and understood by human beings. The strings of symbols which compose natural languages are usually referred to as sentences and the users of a given natural language can usually determine whether or not a given string of symbols from the language's alphabet is in fact a sentence in that language (Fromkin and Rodman, 1978).

This evolutionary development of natural languages and the rather imprecise manner in which natural language sentences are constructed are not, however, characteristics
of formal languages. A formal language is a subset of all possible strings over some alphabet with the important difference that the strings (also often called words or sentences) are constructed according to some very precise set of grammatical rules (Allwood et al., 1977). In addition, for every formal language there is a well-defined procedure for determining whether or not a given string of symbols is in fact a sentence in the language (Hopcroft and Ullman, 1969; Prather, 1976). FORTRAN and BASIC are examples of formal languages as is the set of well-formed formulas of symbolic logic.

Natural languages allow us to speak about a wide variety of things but often in a rather ambiguous and cumbersome way. Formal languages, on the other hand provide the mechanism to speak very precisely about certain well-defined subject domains, but they are usually either very tightly constrained or they are highly abstract. It is not surprising, therefore, that the languages used by scientists and engineers are often combinations of natural and formal languages. This of course has many pragmatic advantages but the precise characterization and analysis of the resulting languages can be quite elusive.

Any language, be it natural, formal, or a combination, possesses syntax, the characteristics of how individual symbols and groups of symbols are related in the sentences of the language. Syntactical rules dictate how symbols may be combined to form "legal" strings, that is, sentences in the language. These rules also influence the decision of whether or not a given candidate string of symbols is a sentence of the language. The semantics of a given language are the rules that attach meaning to symbols, strings of symbols, and sentences. In the English language, for example, the decision to append the symbol "e" to the string "blu" to form the string "blue" is a syntactical issue. To associate the string "blue" with the color of my notebook is of semantic interest.

In any given language there may be more than one category of sentences. For example, in English, a sentence may be classified as a question, a statement, or an imperative. Statements form a class of particular interest when speaking about the relationships of languages and subject domains. A statement may be most generally defined as a sentence which takes on a truth value when interpreted in terms of some subject domain. That is, a statement may be either true or false (but not both). Pragmatically speaking, this truth value is determined by the subject domain and the semantics of the language. The concept of a
statement forms the basis for the definition of a theory.

1.3 Theories

The term theory is used in many contexts and, therefore, it may be inferred to have many definitions. For example, Gale (1979) defines a theory as a conceptual system having at least two elements: a set of observation correlations and a metaphysical hypothesis which is linked to those correlations. Achinstein (1969) on the other hand seems to consider a theory to be a collection of propositions or assertions. Both of these authors are primarily concerned with empirical theories, theories based on observation. Chang (1973), a mathematician and model theorist concerned primarily with the deductive theories of mathematics, defines a theory somewhat more abstractly as simply a set of sentences. Suppes (1961), although speaking of empirical science, defines a theory in much the same way.

Since this dissertation is not concerned on the one hand with the philosophy or sociology of science or, on the other, with purely deductive mathematics, a definition of the term theory will be used which is somewhat of a compromise between that of Achinstein and that of Suppes: a theory is a set of statements about some subject domain.

Note that regardless of whether it is deductive or empirical, a theory is a linguistic entity. This is an important consideration in distinguishing theories from models, which will be defined below.

Some insight into how theories are structured can be gained by breaking the concept of a theory down into components, showing how the components are related, and categorizing the statements of the theory. Theory components include primitive terms, defined terms, observation terms, correspondence rules, axioms, logical calculi, and theorems.

Tarski (1965) defines primitives or primitive terms as those expressions in a theory which are immediately understandable, needing no explanation. He also refers to them as undefined terms to further explicate their meaning. Primitive terms are those components on which all other components are defined.
Tarski refers to the defined terms of a theory as those terms which are explained through the use of the primitive terms. Achinstein (1968) describes a defined term (or an explicitly defined term, in his terminology) as one for which necessary and sufficient conditions have been given by employing primitive terms.

Any term in a theory, be it primitive or defined, may be considered to be either an observation term or a theoretical term. Hempel (1970) defines an observation term as one naming an "observable", some component of the subject domain of the theory which can be observed, either directly or indirectly through instrumentation. A theoretical term, on the other hand, refers to something which is not observable but, in empirical theories, a theoretical term may, and in fact must, be defined by reference to observation terms. Theoretical terms are purely artificial in the sense that they refer to things which cannot be observed.

For an empirical theory to be empirical, there must be some way of relating the theoretical terms to the observation terms. Achinstein (1968) refers to the semantical rules which specify such relationships as correspondence rules.

To illustrate, consider a physical theory in which the mass of an object is an issue. In such a theory, mass would be considered to be a theoretical term since mass itself is not observable. Instead, mass must be explained in terms of some observation terms, such as a scale and a pointer on a mass balance. The correspondence rules relating mass to these observation terms would specify how the observable scale reading leads to the determination of the object's mass when it is placed on the balance pan.

Axioms are the first type of theoretical statement to be considered. Tarski (1965) defines an axiom or primitive statement as one which is simply accepted as true with no requirement for the establishment of its validity. Axioms then are the assumptions of a theory on which the truth of the other statements in some way depends.

The logical calculus of a theory is a set of rules which allow one to construct new, true statements from the set of axioms. In deductive theories we think most commonly of the rules of the first order predicate calculus, such as modus ponens and modus tollens, which allow us to deduce new statements from ones already shown to be true, based on the axioms. In empirical theories we
in addition use inferential logic to generalize from specific observations which have been made.

The new statements resulting from the application of the rules of the logical calculus to the axioms are called theorems. The axioms and the theorems resulting from them form the statements of the theory. More formally stated, the statements of the theory, including the axioms, are closed under the logical calculus.

By way of illustration, consider a simple human-machine system, a bicycle and rider, BR. A theory about BR would consist of statements about the rider, his bicycle, his environment, and relationships among these things.

In the bicycle/rider system observation terms could include pedal, foot, hand, and names for various landmarks our rider might pass on his journeys. Theoretical terms could include pedaling (the act) and forward movement. The necessary correspondence rules could relate the theoretical term pedaling to sequences of observations of feet and pedals, and the theoretical term forward movement to sequences of observations of BR in relation to various landmarks.

An axiom of our theory could be "Pedaling produces forward movement." We could develop a logical calculus which would include the following rule: "If landmark B is between landmarks A and C, and the bicycle/rider passes A and C then it passes B as well." Such logic rules, aided by observations, could be used to develop a large collection of statements about our system's travels.

1.4 Models

Probably more confusion surrounds the use of the term model even than that of the term theory. It has rather recently become quite fashionable to use this term in a very wide variety of areas in a wide variety of contexts. Kaplan (1964) and Suppes (1961) both present a survey of some of the definitions and uses of the term model in the scientific and mathematical literature. Their accounts serve as a basis for the summary of definitions that follows. This list of alternate definitions of the term model is not intended to be exhaustive, but it does give an idea of the major concepts surrounding the term.
The first and possibly simplest idea associated with the term model is that of a physical analog of the domain of interest. Both Kaplan and Suppes explicitly mention this use of the term and Achinstein (1968) gives examples of it as well. In a physical model, such as a model airplane or a model of a molecule of an organic compound, certain physical properties of the subject domain are preserved in the model. For instance, in both of these examples, spatial relationships of the subject domain are represented by spatial relationships in the models but on a much smaller or much larger scale, respectively.

The second concept of a model is that of a conceptual analog. Kaplan and Achinstein mention or use the term in this sense and Hesse (1963), although distinguishing two types of models, is clearly referring to this definition in both cases. A conceptual analog, such as the physicist's imaginary mass and spring, is like a physical model in that it preserves certain properties of the subject domain. However, the relationship of interest in the subject domain may be represented by a relationship of an entirely different kind in the model. For example, the mass and spring may be used to model an oscillator in an electronic circuit. In this case the vertical displacement of the mass is analogous to the voltage output of the oscillator while, physically speaking, these two properties are quite different.

The third sense of the term model is undoubtedly the most confusing. Kaplan refers to it as the definition of a model as a theory, formulated in fairly precise terminology, such as that of mathematics. Suppes, Leatherdale (1974), and Achinstein (1968) also give this as an alternate definition of the term. In each case, the authors refer to situations in which some mathematical expression used to describe a system, such as the equations of motion of a physical body or the probability distribution function of some population, is called a model. It is not clear in these cases how a model is distinguished from a theory.

In all this confusion over the term model, we wonder with Suppes whether or not there is some definition which could provide a structure within which the other concepts of the word could be explained. Fortunately, if we look to the mathematical logicians, such as Tarski (1965, Tarski et al, 1953) and the model theorists (Chang and Keisler, 1973) we can find such a definition. Although their approach and terminology differ somewhat, these authors offer us a quite simple definition: a model of a theory is a structure in
which the statements of the theory are interpreted as true. This definition needs a bit of explanation since if subject domain and semantics determine the truth value of the statements, changing the subject domain will, in general, alter truth values.

We start with the original statements of the theory which speak about the subject domain, which is called the intended model of the theory by logicians. These statements contain a number of constants, the terms, which have special significance in relation to the intended model. If we replace these constants with variables, we have what Tarski (1965) calls sentential functions in place of the original statements. These sentential functions cannot assume truth values until the variables they contain are given values. When we relate them to a structure, the intended model or another, the variables are assigned values unique to that structure. If the resulting statements are true, we say that the structure is a model of the theory.

More formally speaking, a model may be defined by two sets. That is, a model $M$ may be defined:

$$M = (U, R)$$

where $U$ is a set of objects, sometimes called a universe, and $R$ is a set of relations, some defined on the set $U$, some defined on other relations of $R$ themselves. This definition is a generalization of that of Przelecki (1969).

Although sets and relations will be more completely defined in Chapter 3, we can clarify this formal definition a bit now. When defining any model, $U$ refers to the set of objects which comprise the structure that is the model. These objects may be physical, conceptual, or they may be totally abstract, having no interpretation, as in many mathematical structures. The elements of the set $R$ relate these objects and often define relationships between the relations themselves. For example, the geometrical relationships of the components of some physical structure are defined by a relation. These physical relationships themselves may change with time. This temporal relationship is itself defined by a relation.

This definition relating models to theories can be used to interpret the other definitions listed above.
Physical models are true models in the sense that they are physical structures in which the statements of the theory are interpreted as true. Conceptual analogs are conceptual structures in which the statements are interpreted as true. The only difficulty arises when trying to justify the third sense of the word model using this definition. It is clear that there is a mismatch for this sense cannot distinguish between models and theories. In this case, we must just keep in mind that some authors use the term model to refer to what is more properly called a theory.

While mathematicians can define abstract mathematical structures that satisfy all the statements of a theory, practically speaking, the identification or construction of structures that satisfy all the statements of an empirical theory may be impossible simply because of the practical implications of listing all statements of the theory and trying to identify or construct a structure in which all of them are interpreted as true. To identify or construct a model of an existing empirical theory, it may be necessary to first "narrow down" the theory, that is develop a theory of a restricted domain contained in but not identical to the original subject domain. It may then be possible to identify or construct one or more models of the resulting theory. This is what is done when models are constructed that represent some of the characteristics of a given subject domain but ignore others. We must be careful when considering models of limited domains not to make invalid inferences about parts of the domain not addressed by the modeled theory.

To illustrate the relationship between models and theories we could consider many theories of our bicycle/rider system, BR, but let us restrict our attention to a possible theory which could be developed from an observed trip in which, using a stopwatch started at the beginning of the trip, a record was kept of where the system was at particular times. Assuming that the language in which this theory was formulated was English, it might contain the following statements:

BR left HOME at time 0.
BR passed the BUS STOP at time 16.
BR passed the CAMERA STORE at time 33.
BR arrived at the PARK at time 42.

These statements would be considered true when interpreted in terms of the system on the observed trip.
Figure 1 is a time line indicating where our bicycle/ rider system was at particular times during the trip. It is a model of our theory in that it is a structure (a diagram) in which the statements of the theory may be interpreted as true. To do this we can perform the following:

Replace "BR left HOME" with "Arrow H"
Replace "BR passed the BUS STOP" with "Arrow B"
Replace "BR passed the CAMERA STORE" with "Arrow C"
Replace "BR arrived at the PARK" with "Arrow P"
Replace "at time" with "points at"

Note that although the numbers in the statements remain unchanged they refer in one case to stopwatch readings and in the other to points on a line on a piece of paper.

Models have several uses in conjunction with empirical theories. For example, if we are interested in the behavior of a particular system which comprises the subject domain of a theory, it may be more convenient to identify or construct a model of the theory which is easier, safer, or more economical to manipulate than the real system. Assuming that this structure is in fact a model of the theory, as defined previously, the behavior of the model,
when interpreted in terms of the real system, can possibly tell us what we want to know.

A more significant value of models, though, is that they can often summarize and condense theories in a way that is much easier to interpret than the original statements of the theory. Therefore models provide an alternate, and often more effective, way of representing the knowledge contained in theories. In other words, a model is a structure for organizing knowledge.

1.5 Formalization

Of course not all of the things called theories and models have the characteristics described above. In the development of many theories, little thought is given to the identification of primitive terms or a logical calculus, and the selection of a language in which to form the statements of the theory is based primarily on what is convenient or traditional. Consequently, the resulting theories, while useful, often lack clarity. For example, many theories contain terms which are not defined but certainly do not satisfy Tarski's definition of a primitive term, that is, one needing no explanation.

Therefore, when we wish to develop a theory in which clarity is an important consideration and to which we may wish to apply mathematical methodologies, there are certain advantages to the development of a formal theory along the lines of the definitions given above.

The formalization of a theory consists of two components, its axiomatization and its interpretation by models.

Axiomatization consists of identifying those components of a theory defined above. That is, to axiomatize a theory a suitable language must be selected, primitive terms must be identified, defined terms must be defined, axioms must be stated, and a logical calculus must be selected or developed. From these components, the statements or theorems of the theory may be derived.

By itself, axiomatization would produce only deductive theories. Formalization also involves the interpretation of the theory by models. This, unlike axiomatization, which is primarily a syntactical process, is a semantical
process.

Of course there are any number of languages which may be used in developing a theory, but the use of the language of set theory is particularly attractive because it is very general, yet it is very precise. Suppes (1965) gives six advantages to a set-theoretic approach to the formalization of theories.

1. Formalization of a theory brings out the meanings of its concepts in an explicit fashion.

2. A set-theoretic approach presents the possibility of the standardization of formal theories in different branches of science.

3. A more general perspective, often required by a formal approach, can give a better idea of the context of the statements of the theory.

4. A formal approach requires a great deal of objectivity.

5. A formalized theory contains its own assumptions in the form of axioms.

6. It is often possible to determine from a formal theory just what assumptions are necessary to derive its theorems. This permits the removal of redundancy from the theory.

In addition to Suppes' advantages, there are two other advantages to the set-theoretic formalization of theories, which may in fact be more significant.

1. The explicit nature of the formalization process can help one to identify and avoid logical paradoxes.

2. Since one must start from a very abstract and general perspective in formalization, the tendency
to use inappropriate methods in describing the subject domain simply because they are convenient or traditional can be avoided.

1.6 Summary

In this chapter some background for the dissertation research has been presented. A theory was defined as a set of statements about some domain and a model of a theory was defined as a structure in which the statements of the theory are interpreted as true. Finally some motivation was given for the development of formal theories. It is now possible to look at the subject domain of interest with an eye towards the development of the foundations for a theory which will be useful in describing and addressing design and analysis questions in it.
CHAPTER 2

Human-Machine Systems

2.1 Introduction

The purpose of this dissertation is to develop the foundations for a theory which will provide a structure for organizing knowledge and activities related to the study of human-machine systems (HMS's). In this chapter, the subject domain of HMS's will be explored by defining an HMS, listing some common characteristics of HMS's and giving several examples. Some existing theories of HMS's and some tools currently used in HMS design and analysis will be discussed and the need for a new and somewhat different theory of HMS's will be presented.

2.2 Definition of a Human-Machine System

Although any collection of humans and machines might arbitrarily be called an HMS, a more restricted definition will be used here: A human-machine system is a set of interrelated components, some human, some machine, which function together to achieve an objective or goal which could not be achieved consistently, efficiently, or safely by any of the components independently.

This definition could refer to any number of systems ranging in complexity from a carpenter with a hammer to a manned spacecraft system. But, while a rather simple bicycle/rider system has been and will be used in this dissertation to illustrate some of the concepts presented, we are primarily concerned with the development of a theory which speaks about a class of HMS's with some rather special characteristics.
HMS's of interest are generally somewhat more complex than the bicycle/rider system. That is, they contain more components and the machine components are often quite sophisticated. These systems usually possess a high degree of automation. Today, this means that some of the machine components are digital computers or other similar controllers. Such components perform functions for which the humans are unsuited for or are likely to become bored by.

While single operator HMS's are common, in general the systems of interest include multiple operators which share the human responsibilities. Consequently, communication and coordination of activities become rather important considerations in this class of HMS's.

By virtue of the fact that a high degree of automation exists in such systems, machines perform repetitious, moment-to-moment monitoring and control activities and the human operators are responsible for monitoring the machines, planning activities, and supervising overall system operation. For this reason, this class of HMS's has been referred to as supervisory control systems (Sheridan and Johannsen, 1976).

All HMS's of interest are capable of storing information. Procedural information, that which directly guides system activities, and factual information, commonly called data, are stored in one or more knowledge bases which may be implemented in the form of such things as human memory, handbooks, or magnetic tape.

HMS's are goal-directed. That is, their activities are oriented around achieving goals or objectives set for them either from outside or from within the system. They are capable of breaking higher level goals down into simpler goals which may be achieved by executing procedures through a process called planning. The procedures used may be explicit, as in the case of handbook procedures, or they may be highly personalized. They may be very simple or they may be complex combinations of simpler procedures synthesized to achieve the goal at hand.

Not only are the systems of interest physically complex, their activities are complex as well. At any one time, an HMS may be engaged in several activities which, in a sense, compete for the attention of the human and machine components of the system. It is therefore necessary for at least some of the components to be able to allocate their limited attentional resources to specific activities. This
attentional allocation is a dynamic process driven by the relative importances attached to the activities and the events occurring within and in the environment of the HMS.

2.3 Examples of Human-Machine Systems

These characteristics are exhibited in a rapidly-growing number of systems. Brief descriptions of a fairly representative set of human-machine systems are presented below.

1. Flight Management Systems - The control of modern aircraft, such as military fighters and commercial airliners, has become more of an exercise in systems management than one in moment-to-moment continuous control. Consequently, flying such aircraft is becoming known as flight management.

The cockpit of a modern aircraft is quite complex but much of this complexity is associated with management of the automated systems that actually control the aircraft. The multiple operators in flight management systems, including the pilot or aircraft commander, the copilot, and the engineer, are now responsible to a large extent for commanding, sequencing, and monitoring the automated systems.

Flight management systems usually have well-defined goals or objectives, and the aircrew is responsible for achieving them by planning and carrying out activities through the use of procedures. Quite often, especially in the case of commercial flight operations, these procedures themselves are well-defined and the operators have received intensive training in their execution. Many procedures are explicitly recorded in the handbooks and manuals carried by law or regulation in the cockpit. The members of the aircrew often refer to such knowledge bases for the contained procedures and other information useful in operation of the aircraft.

Of course the cockpit can be a very busy place with many on-going activities competing for the attention of the crew members. The ability of
the crew to attend to many conflicting activities in a meaningful way is an important performance measure of the entire aircraft system.

2. Air Traffic Control (ATC) Systems - With the increasing density of aircraft at air terminals, the job of safely and efficiently routing them to and from airports is becoming more and more complex. ATC systems are designed to deal with this issue through the integration of people and machines to control the air traffic flow.

ATC systems themselves are quite complex entities, incorporating large numbers of sensing, display, and communication devices as well as human controllers. While some Korean War vintage systems are still used (especially when primary systems fail) most ATC systems are becoming more and more automated with computers improving system effectiveness through display enhancement, record keeping, and cognitive aiding. The operators of these systems are still responsible for a great deal of decision making and coordination in addition to the supervision of the automated components.

The major objectives of an ATC system involve coordinating the movements of aircraft within a specific control area in such a way that separation of aircraft is maintained. The activities of the controllers can be explained in terms of achieving this goal by developing subgoals and executing procedures to achieve them. Some of the procedures used are dictated by laws and regulation as well as local policies while others are developed and used by the individuals themselves. The coordination of monitoring and control activities in ATC systems are crucial to the safety and convenience of many thousands of people daily.

3. Anti-Aircraft Artillery (AAA) Systems - In the military arena the increasing speed and survivability of hostile aircraft, including airplanes, missiles, and cruise missiles, has necessitated the development of some rather sophisticated, highly automated weapons systems. Such AAA systems may be fixed, mounted on land
vehicles, or aboard ships. They typically incorporate one or more human operators who perform complicated decision and visual functions, such as locating incoming targets using optical systems. When such a target has been acquired, the automatic tracking components of the system can (often) follow the target until it is within firing range at which time the firing is accomplished either manually or automatically.

The goals of such systems are quite simple: to destroy hostile aircraft. The AAA operators, though, must use information from existing conditions to decompose these high level goals to bring the equipment and procedures to bear on them. Procedural and other information exists in such systems in the form of manuals, handbooks, and machine storage systems. The operators must coordinate their activities and allocate their attention to procedures in such a way as to protect friendly forces and equipment while destroying hostile aircraft.

4. Command, Control, and Communications (C3) Systems - C3 is another concept that has grown as a result of rapidly advancing military technology. C3 systems control the movement of strategic information in times of both war and peace. In a C3 system large numbers of sensors, including radar, imaging systems, and human observers, gather information about (potentially) hostile forces and relay it through a communications system to an integrated data base. Analysts may query that data base to provide information to military commanders for informational and decision making purposes.

C3 systems are quite complex and have become highly automated as a result of advances in computer and communications technology. They often involve large numbers of people from the observers, infantrymen, and analysts to the top military commanders so coordination is an important issue in C3 design and operation.

Although the acquisition, storage, and presentation of strategic information is a generic goal of C3 systems, the immediate goals of any particular system can be quite varied. The
procedures for achieving them are therefore correspondingly varied and the planning involved in C3 system operation can be quite complex.

5. Teleoperator Systems - Teleoperator systems are used to protect humans from potentially hazardous environments. They are used in the handling of dangerous materials (such as reactor fuel elements), in mining operations, and in deep sea operations.

The physical separation of the operator from the actual manipulating components (mechanical hands, arms, etc.) requires rather sophisticated sensing, communication, and control equipment. To compensate for the rather low bandwidth that may be characteristic of such systems, much automation of sensing and motion functions is often necessary.

The operator of such teleoperator systems, then becomes a supervisor or system manager, using explicit or implicit procedures to control the equipment to perform the physical activity at hand. Some coordination with other operators may be necessary, especially in mining operations. The operator is most apt to use the knowledge base that is his/her memory in storing procedural and factual information necessary in the operations.

6. Process Control Systems - The demands for increased productivity as well as the advancing complexity of production systems have made process control systems the subject of much interest. The control of continuous production systems (such as steel, paper, and chemical plants), discrete part manufacturing systems, nuclear reactors, and electric power generation and distribution systems is an important design and operation factor which must be considered in the very early planning stages of such systems.

Much of the control of these systems is done automatically by computers and other control devices. Human operators must make process decisions, establish set points, and monitor the operation of the process. They are typically provided with complex displays which may be visual
analogs (flow or block diagrams) of the process being controlled. Their goals may be to produce as much of a product as possible or to maintain specified operating conditions. They plan their activities to achieve these goals and execute explicit or implicit procedures to direct these activities.

The coordination of activities, procedures, and people in such systems is a major problem influencing profits, consumer satisfaction, and safety.

2.4 Human-Machine System Issues

We can easily see that human-machine systems are quite important and that important HMS issues should be of some concern to us. Probably the central HMS issue, or at least the one that most others can be explained in terms of, is the development of better human-machine systems, systems that perform their functions more effectively, more efficiently, or more safely than earlier systems.

One way to go about this is to modify and improve existing systems. Much effort is spent annually to enhance the operations of HMS's, improving equipment, training operators, and revising procedures. Sometimes, though, the requirements for an HMS are just too demanding to be met by an existing system and a new one must be designed. Consequently design becomes an important issue.

For simple systems, rule-of-thumb design techniques are adequate but there exists no optimal design technique for HMS's. In other words, it is impossible to produce a single design which, when implemented, will result in a system which will meet all of its requirements in the most efficient, most effective, and safest manner possible.

This raises another HMS issue, that of the development and evaluation of alternative designs from which the best one can be selected and implemented. From the designs alternatives, prototypes can be built and tested under real and simulated conditions to determine their strengths and weaknesses before any design is committed to production.
But even prototyping of many system is impossible until all but just a few of the very best alternatives have been eliminated. Large aircraft, C3 systems, and nuclear reactor control rooms are much too expensive and complex to test every possible alternative design. We therefore come to the bottom line issue, the prediction of HMS performance. The vast bulk of HMS research is related to posing and answering very specific questions about how different HMS components perform under various conditions.

The easiest task here (but by no means a simple one) is the prediction of machine performance. Since we build machines from simpler, better understood components, we know what their inherent capabilities and limitations are and can therefore make fairly confident statements about how they will perform in given situations.

The same is not true about human operators. In spite of centuries of observation of human behavior, it is not possible to predict how an operator will perform except, possibly, in very highly constrained circumstances. Fortunately, the HMS is sometimes a fairly constrained system in which the operators, due to physical limitations imposed by the machine components or by their own training and skill levels, have relatively few degrees of freedom. Human performance prediction in such systems is a much more certain endeavor.

But the real issue here is not how machines perform or how humans perform, but how systems of humans and machines perform. It is sometimes possible (again, in very highly constrained situations) to predict system performance by considering only inputs and outputs. However, in most realistic situations, we must consider how HMS's, and specifically the humans in HMS's, go about organizing the activities that produce this input/output behavior. The study of goals, procedures, and planning is therefore another important HMS topic.

In summary we can say that the main human-machine system issue, that of producing better HMS's, boils down to the evaluation of alternative HMS designs, primarily through performance prediction. Much HMS research is aimed at gaining a better understanding of how HMS components behave under various conditions to aid in this prediction and evaluation process.
2.5 Existing Human-Machine System Theories

This research produces HMS theories of very limited domains, theories that speak about rather specific aspects of HMS's, such as operator behavior under certain conditions. Although these theories quite often do not readily fall into distinct categories, it is possible to classify and discuss them in terms of the general approach that underlies them. The theories that we will talk about below include those based on information theory, control theory, decision theory, queueing theory, and the occurrence of discrete events.

One of the earliest systematic approaches to understanding and predicting the performance of complex human-machine systems was based on information or communication theory. According to Deininger and Fitts (1955), there existed a need for a single index of performance for the evaluation of systems in which information processing was an important function.

Fitts and Posner (1967) divide human operator information processing into three categories: conservation and transmission; reduction; and creation and elaboration. Much early research concentrated heavily on the first category, treating the operator as an information channel and trying to determine the information capacity and transmission rate of that channel. For example Elkind and Sprague (1961) measured the information transmission rate of humans in simple pursuit and compensatory tracking experiments. Fitts and Peterson (1964), in an attempt to understand the relationship between the human's perceptual and motor systems, investigated the effect of uncertainty (directly related to the amount of information of a stimulus) on reaction time in discrete motor responses.

This approach has potential applications in HMS design, specifically in the design of displays. Senders (1955, 1964) investigated relationships between information and instrument panel layout with respect to both total time spent in observing a single display and to the operator's visual sampling strategy in multiple instrument situations.

The physical behavior of many systems, including aircraft, submarines, and chemical plants, can often be adequately described as linear dynamic systems. It therefore makes some sense to attempt to describe the operators of such systems as servomechanisms, linear components of the systems themselves. Manual control
theories of HMS's have therefore relied heavily on control engineering for concepts and methodologies to describe such systems.

Some of the earlier manual control theories of HMS's used the concept of quasi-linear descriptions of the human operator. That is, it was felt that the inherently non-linear, mode-switching behavior of the human operator could be described by means of a number of linear descriptions, one for each mode. For example, in the well-known crossover 'model' (McRuer and Jex, 1967) the human operator is described as a good servo, one whose gain is greater than unity for input frequencies less than the crossover frequency and less than unity for frequencies greater than the crossover frequency.

The quasi-linear theories depended heavily upon classical control theory and the techniques used in control engineering at that time, including transform techniques and Bode plots, were very popular tools. Unfortunately, classical control theory dealt primarily with single input systems and HMS's are generally considered multidimensional. The multiple state variable approach of modern control theory provided a fix for this problem.

One of the chief manual control results to come out of modern control theory is the description of the operator as an optimal controller. Originally developed by Baron and Kleinman (Baron and Kleinman, 1969; Kleinman et al, 1971), the optimal control 'model' assumes that the HMS operator is well-trained and therefore performs as an optimal servomechanism. This servomechanism includes a Kalman filter for state estimation and a predictor component for predicting future system state to accommodate for operator time delays.

Engineers are primarily interested in system performance while psychologists are interested to a large degree in explaining operator behavior. Pew (1974), while using a manual control perspective, describes the operator in terms of functional units corresponding to what we know about how humans process information in perceptual-motor tasks.

Since the operators of highly automated systems function to a large extent as decision makers, decision theory based descriptions of HMS's have also been widely used. As in economic and management applications of decision theory, the decision maker is faced with a number of alternatives, in our case control strategies, routes, or
possible causes of malfunctions. He/she must select one course of action based upon some explicit criterion such as expected benefits, expected costs, or personal utility. Cohen and Ferrell (1969) describe an application of decision theory in a manual control task in which data is used to generate a scalar, p. The p scale is divided into regions associated with control strategy alternatives and the region associated with p determines the strategy used.

Queueing theory has been used in at least two applications in HMS analysis and design. The work of Carbonell in instrument sampling is quite well known (Carbonell, 1966; Carbonell et al., 1968). In this approach, instruments are described as queueing up for service, that is, visual sampling by the operator, and a number of parameters relating to the operator's sampling policy can be determined by using queueing theory techniques. Requirements for instrument panel layout and visual sampling strategies can be determined in part from such investigations. Senders and Posner (1976) used the same approach to develop manning requirements for HMS's containing multiple operators who must scan displays.

The idea has also been extended to multi-task situations. Walden and Rouse (1978) describe an HMS in terms of multiple tasks which queue up for an operator's attention and receive service according to a priority scheme. Using such an approach it is possible to determine the mean time spent on each task, the average waiting time for a task to receive attention, and overall system task throughput.

The idea of queues of things may be extended naturally to the concept of the arrival or occurrence of discrete events. Discrete event theories of HMS's are usually modeled by computer simulation. Siegel and Wolf (1969) have written the classic work on this approach in which events, such as system malfunctions, occur according to some specified probability distribution and are serviced by the human and machine components of the HMS according to other specified distributions.

Another approach used in discrete event theories is to describe an HMS using Petri nets. Petri nets are directed bipartite graphs that have found wide use in the field of computer science to describe systems undergoing sequential and concurrent activities (see Peterson, 1977 for an excellent survey) but they have also been used by Schumacher and Geiser (1978) to describe a human operator in a multi-task situation.
A number of efforts have been made to integrate two or more of the above theoretical perspectives to provide a somewhat more comprehensive view of HMS's. For example, Muralidharan and Baron (1979) have combined the concepts of decision making and optimal control to form a description of an RPV operator engaged in decision-making, monitoring, and control tasks. A somewhat more general perspective has been used by Johannsen and Rouse (1979) in describing an HMS using the analogy of a time-sharing computer system. In the analogy the tasks the operator must perform are represented by the programs of the computer system which, like activities competing for the operators' attention, compete for computer system resources based in part on program priority and resource availability. The authors discuss briefly how such a perspective could be used to integrate a number of methodologies used in the study of HMS's.

2.6 Shortcomings of Existing Theories

Most of the theories considered above are intended to address very limited domains. The research that led to their formulation was usually conducted to answer some very specific design and analysis questions and when applied in the intended domain, very useful results have often been produced. However, in designing and improving complex human-machine systems, we must consider the global issues of component compatibilities, coordination and interaction. A very detailed knowledge of how a component behaves in a very restricted class of conditions is, by itself, of little use in planning system architecture. Consider the following quotation from Johannsen and Rouse (1979):

... the success of models in limited domains has not had substantial impact in realistically complex domains. For example, manual control models are not everyday tools of the aircraft designer. Further, manual control models capture only a small portion of the total task of driving an automobile. For these reasons, designers have been known to claim that mathematical models of human behavior are not particularly useful...

Lack of scope and lack of the ability to account for other aspects of HMS behavior would then seem to be rather
important limitations of these theories of limited domains. Let us consider some examples.

Human-machine system theories based on information or communication theory treat the various HMS components as information channels. This is certainly appropriate in some cases, but consider the strategy of the human operator. Information theory descriptions of the operator can account for things like channel capacity and transmission rates, but they are wholly inadequate in explaining what the operator does with information, how he/she organizes it, and how he/she goes about planning information processing activities.

Manual control theories are useful in describing situations in which an essentially linear human operator controls an essentially linear machine. Again, this is quite adequate in many cases, especially when the actions of the operator are highly constrained by the physical characteristics and dynamics of the machine. Unfortunately, in most interesting cases, the operators of HMS's must deal with inherently non-linear phenomena. Manual control descriptions cannot consider man's role as decision maker for such behavior lies outside the realm of control engineering.

On the other hand, though decision theoretic descriptions of the operator have often given us much insight into how the operator goes about selecting from various alternatives, they can tell us nothing of the dynamic behavior of HMS's. In fact, most decision theoretic descriptions are time independent, a significant disadvantage in the analysis of systems whose behaviors as functions of time are very important.

Queueing theories and discrete event theories of human-machine systems share a common disadvantage. Although it is often useful to think in terms of queues of instruments, tasks, and events, such theories can usually only speak of things like mean waiting times, mean service times, and system throughput. It is significant that they do not address the issues of display characteristics, how tasks are performed, and what HMS components actually do when a particular type of event occurs.

There have been a number of attempts to combine two or more of the above approaches. However, it seems as though they all manage to leave out some important aspect of HMS's. For example, the combination of the decision theoretic approach with a manual control description still
does not address the issue of operator strategy in the control and decision making processes.

Those theories that have attempted to integrate most of the aspects of HMS's have possibly shed more light on global HMS concerns than those theories of limited domains. Unfortunately, though, this has usually been at the expense of precision and clarity.

There is in principle no reason why we cannot have a human-machine system theory that is at once broad in scope, capable of addressing a significant portion of the major HMS issues, and formulated in a language sufficiently precise to allow the posing and answering of specific design and analysis questions. It is not reasonable to presume that such a theory would at the outset cover the entire field of HMS study to a depth already attained in specific areas, but it seems feasible to expect that it could provide a framework into which existing HMS theories could be integrated. Such a theory could provide the designer with a means for organizing what is known about very specific situations into a systematic, usable whole.

The formulation of the foundations of such a theory is the objective of this dissertation research. No hope is held of producing a structure that is complete (in the formal sense) or one addressing all HMS issues adequately at the outset. It does seem possible, though, to develop a list of terms, relationships, and assumptions, defined and stated formally in a precise language, that can serve as a basis for understanding HMS architecture and behavior and for formulating precise questions to guide research.

2.7 Summary

This chapter has served to introduce the subject domain of the theory to be developed subsequently. Human-machine systems were defined and special characteristics of interest were discussed. Several examples of HMS's exhibiting these characteristics were given. Issues involved in the study and design of HMS's were listed along with some currently popular approaches to resolving these issues. Finally, the deficiencies of these existing HMS theories were identified and the need for a new theory of HMS's, one having at the same time adequate scope and sufficient focus and precision, was stated.
CHAPTER 3

Development of Terms and Axioms

3.1 Introduction

As discussed in Chapter 1, a theory is a collection of statements about some subject domain. Theories capture what we know about our environment in linguistic structures that we use to organize our thoughts and guide our actions. Consequently, how we develop these theories is an important issue. In Chapter 1, it was argued that formal development of scientific theories, using the concepts and notation of set theory, has a number of distinct advantages, including explicitness, generality, and qualities that assist one in avoiding certain traps that can come about through the use of methodologies whose assumptions are not clearly understood.

In chapter 2 the subject domain of our interest, human-machine systems, was briefly outlined, and existing HMS theories and methodologies were examined. It was concluded, though, that no current HMS theory has both the broad scope and the precision necessary to deal satisfactorily with complex HMS design and analysis issues. It was our conclusion that such a theory was needed.

In this chapter we will undertake the development of some defined terms and axioms for such an HMS theory. This development, while not completely rigorous, will be as precise and formal as is possible given the level of abstraction desired. The first section will briefly introduce some background and some of the notational conventions used throughout. The subsequent sections will cover the components of a formal theory: observation terms, primitives, correspondence rules, defined terms, axioms, and a logical calculus. Finally, some conjectures about what types of theorems might eventually be incorporated in the theory will be discussed.
3.2 Preliminaries

It is assumed that the reader is somewhat familiar with the theory of sets and the notation used therein. In the way of review, however, in this section we will explain some of the notational conventions which will be used in the formal development that follows.

In many cases, we will use the notation of symbolic logic to shorten the expressions made in the development. Assuming that $P$ and $Q$ are statements (or propositions), the following conventions will be used:

- $P \iff Q$ is read $P$ if and only if $Q$
- $P \Rightarrow Q$ is read $P$ implies $Q$
- $P \& Q$ is read $P$ and $Q$
- $P \lor Q$ is read $P$ or $Q$ (inclusive or)
- $\neg P$ is read Not $P$
- $\forall x$ is read For all $x$
- $\exists x$ is read There exists an $x$ or
    For some $x$
- $\exists ! x$ is read There exists exactly one $x$
- $\exists \,$ is read Such that

In the way of set notation, brackets, { and }, are used to delimit lists of the elements of sets. For example, if $A$ is a set containing the elements $a$, $b$, and $c$, we can write:

$$A = \{ a, b, c \}$$
\( G \) denotes set membership:

\[
\begin{align*}
\text{a} & \in \mathcal{A} \\
\text{b} & \in \mathcal{A} \\
\text{c} & \in \mathcal{A}
\end{align*}
\]

while \( \emptyset \) denotes its negation:

\[ x \notin A \iff \neg (x \in A) \]

Set inclusion is denoted by \( \subseteq \). \( B \subseteq A \) is read \( B \) is a subset of (or is contained in) \( A \) and is true if and only if all elements of \( B \) are also in \( A \).

The union of sets \( A \) and \( B \), denoted \( A \cup B \), is a set consisting of all elements of \( A \) and all elements of \( B \) (with no duplication). If we wish to speak of the union of multiple, indexed sets, \( A_1, A_2, \ldots, A_m \), for example, we write:

\[
A = A_1 \cup A_2 \cup \ldots \cup A_m = \bigcup_{i<m} A_i
\]

The null or empty set (having no elements) is denoted by \( \emptyset \).

The power set of \( A \), the set of all subsets of \( A \), is denoted \( \mathcal{T} \mathcal{T} \mathcal{A} \).

\( I = \{ \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \} \) is the set of integers. \( N = \{ 1, 2, 3, \ldots \} \) is the set of natural or counting numbers. Certain subsets of \( N \) will be used as index sets occasionally and are defined:

\[
N = \{ i \mid i \in N \& i \leq m \}
\]
3.3 Subject Domain

Recall from Chapter 1 that a theory's subject domain is what that theory speaks about. Alternatively, it is called the intended model of the theory. Chapter 2 covered the subject domain of the theory to be developed here, that of human-machine systems, but, for the sake of completeness, let us briefly summarize it.

A human-machine system is a set of interrelated components, some human, some machine, which function together to achieve goals or objectives which could not be achieved by any individual component. The HMS's of particular interest in this research are highly automated, complex, and their operators tend to function as monitors and supervisors rather than as continuous moment-to-moment controllers. These systems incorporate knowledge bases for storing information and use procedures to guide their activities in achieving goals. Examples include automated aircraft systems, air traffic control systems, and electric power generation and distribution control systems.

3.4 Observation Terms

The observation terms of a theory are observables in the subject domain. They are related to the theoretical terms of the theory by correspondence rules. Together, observation and theoretical terms serve as components of the statements of the theory.

The observation terms of the HMS theory that we are aiming at fall into three categories. The first contains the names of the physical components of all possible HMS's. For example, since we can observe a pilot, an instrument panel, and a set of controls in an aircraft cockpit, names for these three items are observation terms of the theory.

The interrelationships of the components of an HMS may be described in terms of inputs and outputs which often can be observed. Names for observable inputs belong to the second category of observation terms of the theory. For example, alphanumeric characters which appear on the
instrument panel of the above example may serve as input to the pilot and pilot hand movements may serve as input to the controls. In this case, names for the displayed information and the pilot's movements are observation terms.

Finally, the names for outputs form the third category of observation terms. Characters may be considered outputs of the instrument panel and hand movements may be considered outputs of the pilot, so names for these are observation terms. In fact, as in this case, this is redundant, since these terms have already appeared in the theory as inputs.

3.5 Primitives

The primitives for the terms and axioms of the theory to be developed of this theory of HMS's are sets and elements.

A set is quite simply a collection of things which we can give a name, and an element is a member of a set. This intuitive notion will be perfectly adequate for the formal development.

For example, we can define the set $A$ consisting of the elements $a$, $b$, and $c$ as:

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

and we may then write:

$$ a \in A $$
$$ b \in A $$
$$ c \in A $$
3.6 Correspondence Rules

We can use the terms element and set to talk about things and collections of things in the subject domain of HMS's. Therefore, a correspondence rule is just that explicit or implicit statement that relates an element or a set with a thing or a collection of things.

For example, if we wish to refer to the instrument panel, pilot, and controls of the aircraft mentioned above, we can let COMPONENTS be the set of components of the cockpit. We then have:

\[
\text{COMPONENTS} = \{ \text{instrument panel, pilot, controls} \}
\]

Clearly, instrument panel is an element of COMPONENTS and when we use the term "instrument panel" we are referring to the instrument panel of the cockpit. When we refer to "COMPONENTS" we are referring to the collection of physical components in the cockpit. It is important, here as always, to distinguish between things and names of things. Quotes will not be used in what follows.

3.7 Defined Terms

Two types of defined terms will be used in this development, standard defined terms and specific defined terms. The standard terms are not unique to this discussion, but are common to many formally developed theories. The specific terms are, on the other hand, unique here. The standard defined terms will be covered first.
3.7.1 Standard Defined Terms

Standard defined terms include ordered n-tuple, cartesian product, relation, function, time set, and system.

An ordered n-tuple is a special type of set having n elements. An n-tuple, \( a \), is written:

\[ a = (a_1, a_2, a_3, a_4, \ldots, a_n) \]

where the order of elements is significant.

If \( A_1, A_2, \ldots, A_n \) are sets, then:

\[ A = A_1 \times A_2 \times \ldots \times A_n = \{ (a_1, a_2, \ldots, a_n) | (\forall i \in \mathbb{N}) a_i \in A_i \} \]

is the cartesian product of \( A_1, A_2, \ldots, A_n \) and \( A_1, A_2, \ldots, A_n \) are called component sets of \( A \).

This may also be written:

\[ x \ A \]
\[ i \in \mathbb{N} \]
\[ n \]

or equivalently:

\[ x \ \{ A \mid i \in \mathbb{N} \} \]
\[ i \]
\[ n \]

When we consider cartesian products of a set with itself we frequently use a shorthand notation:
\[ A^n = A \times A \times \ldots \times A \]
(n times)

For example:

\[ A^3 = A \times A \times A \]

The cartesian product of \(n\) sets may be thought of as simply the set of all possible \(n\)-tuples resulting from taking one element from each set and ordering them in the \(n\)-tuple.

A relation is some subset of a cartesian product. For example, a binary relation, \(R\), on the sets \(A\) and \(B\) is defined:

\[ R \subseteq A \times B \]

That is to say, a relation is a set defined on other sets. This is a general definition of a relation. To define a specific relation, all elements must be listed or a predicate characterizing them must be stated.

Order relations are binary relations having special properties. Let \(Q \subseteq A \times A\) be a relation. \(Q\) is a partial order on \(A\), or we say \(A\) is partially ordered by \(Q\), if and only if the following conditions hold:

Reflexivity:
\[ \forall a \in A \quad (a, a) \in Q \]

Antisymmetry:
\[ \forall a, b \in A \quad (a, b) \in Q \quad \land \quad (b, a) \in Q \quad \Rightarrow \quad a = b \]

Transitivity:
\[ \forall a, b, c \in A \quad (a, b) \in Q \quad \land \quad (b, c) \in Q \quad \Rightarrow \quad (a, c) \in Q \]
Q is called a linear order on $A$, or we say $A$ is linearly ordered by $Q$, if and only if every two elements of $A$ are comparable by $Q$. That is:

$$Q \text{ is a linear order} \iff (\forall a,b \in A)(a,b) \in Q \lor (b,a) \in Q$$

The relation $\leq$ on the integers is a linear order with an infinite number of elements. For a finite example consider the following set:

$$\text{QUALITY} = \{\text{bad, fair, good}\}$$
on which is defined:

$$B \subseteq \text{QUALITY} \times \text{QUALITY}$$

$$B = \{(\text{bad, bad}), (\text{fair, fair}), (\text{good, good}), (\text{bad, fair}), (\text{bad, good}), (\text{good, fair})\}$$

Since $B$ has the properties specified above, $B$ is a linear order and QUALITY is linearly ordered by $B$. If $(a,b)$ is an element of $B$ we can write "$a \leq b$" and read it "$a$ is no better than $b$." For example, we see that "bad $\leq$ bad" ("bad is no better than bad") and fair $\leq$ good ("fair is no better than good") which corresponds with our normal interpretation of these symbols. This notational convention applies to all binary relations.

Functions are special kinds of relations. The following definitions are taken from Stanat and McAllister (1977).

If $A$ and $B$ are sets (either or both of which, incidentally, may be cartesian products) a function (or map or transformation) $f$ on $A$ and $B$, denoted $f : A \to B$, is a relation on $A$ and $B$ such that for every $a \in A$, there exists a unique $b \in B$ such that $(a,b) \in f$. If $(a,b) \in f$ we write $f(a) = b$.

Let $f$ be a function from $A$ to $B$ and let $A'$ be a subset of $A$. Then $f(A') = \{f(a) | a \in A\}$ is a subset of $B$ called
We may classify certain functions according to their characteristics. Let $f$ be a function defined as above. $f$ is said to be surjective (onto) if $f(A) = B$. $f$ is injective (one-to-one) if $a \neq a'$ implies $f(a) \neq f(a')$. $f$ is bijective (one-to-one and onto) if it is both surjective and injective. Functions with these properties are called surjections, injections, and bijections, respectively.

For an example of a bijection, let QUALITY be a set defined as above and let the set RATING be defined:

$$\text{RATING} = \{1, 2, 3\}$$

We may then define a function $f$:QUALITY→RATING as:

$$f = \{(\text{bad}, 1), (\text{fair}, 2), (\text{good}, 3)\}$$

Although not really crucial to the development of the terms and axioms, but since we have discussed the significance of structures in organizing knowledge, this is an appropriate point to more formally define the notion of a structure. Even though we cannot say precisely what a structure is, for we can only write down the names of things and not the things themselves, we can define a structure on sets.

A structure is defined by an ordered pair:

$$S = (U, R)$$

where $U$ is a set of objects and $R$ is a set of relations. Some of the relations in $R$ are relations on $U$. That is:

$$(\exists Q \in R)(\exists n \in N) \rightarrow Q \subseteq U^n$$
On the other hand, relations in $R$ may be defined on other relations in $R$. That is, we might have a case where:

$$(\exists K,L,M \in R) \; K \subseteq L \times M$$

As an illustration, we can describe a brick wall as a structure where $U$ is the set of bricks making up the wall and $R$ contains the relation defining adjacency of bricks. A sentence may be defined as a structure where $U$ is the set of symbols in the sentence and $R$ contains relations defining the arrangement of the symbols in the sentence. A flow or block diagram may be described as a structure where $U$ contains the nodes (blocks) and a relation in $R$ identifies the arcs (arrows connecting the blocks), or we might describe both the nodes and arcs as elements of $U$, with connectedness being a relation in $R$. A theory and its subject domain may be described as a structure in which $U$ contains the terms of the theory and the objects in the subject domain and $R$ contains relations relating terms to objects. Finally, a model is a structure for it is defined in precisely the same way as a structure. A model is generally considered to be a special case, structure being the more general term.

In studying human-machine systems we are often interested in their behavior over time. It is therefore necessary to introduce the concept of a time set. A time set, $T$, is a set, linearly ordered by the relation $<$, with definitions of the relations $<$, $>$, and following naturally. Of course the set of non-negative integers ordered by the relation $<$ is a common example, but other linearly ordered sets will often be more useful while not carrying along the additional properties implied by the integers.

For example consider the following time set:

$T =$ \{beginning, middle, end\}

and the relation:
\[ \leq = \{(\text{beginning}, \text{beginning}), (\text{middle}, \text{middle}), (\text{end}, \text{end}) \]
\[\quad (\text{beginning}, \text{middle}), (\text{beginning}, \text{end}), (\text{middle}, \text{end})\}\]

Such a linearly ordered time set may be perfectly appropriate for many applications. It is important not to confuse this \(\leq\) relation with the common ones defined on the integers or the real numbers. In the following development, meaning will be evident from context.

We have here another example of a structure used to organize knowledge. Recall that a structure, \(S\), was defined \(S = (U, R)\). Our new structure is \((T, \{\leq\})\). That is, \(U = T\) and \(R = \{\leq\}\). We can use such a structure to organize our knowledge about the occurrence of events and many of us would agree that the structure is fairly consistent with our own ideas of chronology.

If we wish to consider such things as time intervals, though, this definition of a time set is inadequate and we must add additional properties to the structure or, more formally speaking, add additional relations to \(R\), to make it consistent with our requirements. We now turn to Windeknecht (1971) for some additional time set properties.

First, we define an operation (another relation) on \(T\). Let \(+\) be a binary operation on \(T\) with the property of associativity:

\[ + : T \times T \rightarrow T \]
\[ (\forall t, t', t'' \in T) t + (t' + t'') = (t + t') + t'' \]

Mathematicians would call \((T, +)\) a semigroup. If we assure that \(T\) has an identity element, \(0\), \((T, +, 0)\) is considered a monoid (semigroup with identity). That is:

\[ (\forall t \in T) t + 0 = t = 0 + t \]

Windeknecht also requires the following properties of a time set:
These properties insure that the time set structure carries with it the concept of time interval (i.e. the interval between two elements is itself an element of the time set), linear ordering, and the notion of one-directionality of time (i.e. there is no way to get back to time 0). These properties are not always required in the following development so a linear ordering will be considered to be a necessary condition for a time set and the other properties will be used only as required.

We will often find it necessary to speak of time segments, special subsets of T. The following definitions of time segments are based on those of Mesarovic and Takahara (1975). For all t,t'GT, where t'>t:

\[ T = \{ t' \mid t' \geq t \} \]

\[ T = \{ t' \mid t' < t \} \]

\[ T = \{ t'' \mid t < t'' < t \} \]

We may define other sets on the time set. For example, let A be an alphabet or set of possible observations, say, of instrument reading inputs to a pilot during a flight or mission and let T be a linearly ordered time set corresponding to the time duration of the period or mission in seconds or some other appropriate time unit. We may then define a function \( x : T \rightarrow A \) where if \( (t,a) \in x \), we say that a was the reading observed at time t. x is called a time function.

Again, following Mesarovic and Takahara, we can define the following time restrictions on x:
A time restriction on a time function allows us to talk about the function over a specified time segment.

We will often find it necessary to speak of multi-dimensional time functions and sets of them. For that purpose, let $A_1, A_2, \ldots, A_m$ be input alphabets and let $X_1, X_2, \ldots, X_m$ be sets of time functions defined as $X_i = \{ x : T \rightarrow A_i \}$. If we define $X$ as a multi-dimensional object, $X = X_1 \times X_2 \times \ldots \times X_m$, we can use the notation $x(t) = (x_1(t), x_2(t), \ldots, x_m(t))$. That is, we can regard the value of $x$ at time $t$ to be an ordered $m$-tuple consisting of the values of its component time functions at time $t$.

To complete the list of standard defined terms, we will now consider the concepts of systems and subsystems. We may use the idea of a system to describe a human-machine system or a component of a human-machine system if we are willing to accept certain restrictions on the description. Specifically, we must allow that such an HMS or component may be described in terms of inputs, outputs, and, possibly, states. Since these restrictions are really consistent with most existing HMS theories, the definitions to be considered are quite natural, although they will be given from a more general and abstract perspective than is normally the case. The following material is adapted from Mesarovic and Takahara (1975) and the ideas are consistent with Windeknecht (1971) and Wymore (1976).

We may view an HMS physically as consisting of a number of interrelated components, some human, some machine, which function to aid in accomplishing the objectives imposed on the HMS. One way to describe these components is in terms of input and output as they evolve over time. This is not an unusual perspective to take for both Miller (1974) and Sheridan and Ferrell (1974) treat the human operator as an information processor which converts inputs to outputs and Van Cott and Kinkade (1972)
devote an entire handbook to the description of various input/output components of human-machine systems.

We have chosen the term system to be used to describe a human-machine system or an HMS component in terms of its inputs and outputs. That is, a system is a structure to organize knowledge about dynamic input and output behavior. Therefore let \( A \) be an input alphabet, a set corresponding to the inputs that an HMS component could receive, and let \( B \) be similarly defined as an output alphabet. Since we are interested in the behavior of a resource over time, we need a time set. Therefore, let \( T \) be a time set, linearly ordered by the relation \( \leq \) with least element 0.

Let \( X \) and \( Y \) be input and output objects, respectively, defined as sets of time functions:

\[
\begin{align*}
X & \subseteq \{ x \mid x : T \rightarrow A \} \\
Y & \subseteq \{ y \mid y : T \rightarrow B \}
\end{align*}
\]

A system, \( S \), may then be formally defined in its simplest form as a relation on the input and output objects:

\[ S \subseteq X \times Y \]

In other words, the system may be described in terms of its input and output over the time set \( T \).

The case we have here is a single channel system, one in which we describe only a single input channel and a single output channel. This may be adequate as, for example, if we wish to use this notion to describe a simple HMS component such as a single axis joystick. Here \( X \) would represent a set of possible inputs to the joystick, that is, a record of joystick displacements by an operator as a function of time. \( Y \) would then represent output of the joystick. If connected to a potentiometer, \( Y \) would consist of output voltages as a function of time. To allow the description of multi-channel systems, such as two-axis joysticks, the input objects \( X \) and \( Y \) must be defined as multi-dimensional. Therefore, let \( A_1, A_2, \ldots, A_m \) be input alphabets and let \( X_1, X_2, \ldots, X_m \) be input time functions defined \( X_i \subseteq \{ x_i \mid x_i : T \rightarrow A_i \} \) let \( B_1, B_2, \ldots, B_n \) be output alphabets, and let \( Y_1, Y_2, \ldots, Y_n \) be output time functions defined \( Y_j \subseteq \{ y_j \mid y_j : T \rightarrow B_j \} \). \( X \) and \( Y \) may then be defined:
\[ X = X_1 \times X_2 \times \ldots \times X_m \]
\[ Y = Y_1 \times Y_2 \times \ldots \times Y_n \]

S may then be defined as above but now in terms of these multi-channel objects.

Since S is a relation, we cannot guarantee that given an input, a unique output will always result. This is not a realistic description of all HMS components since most are designed to have, or are selected because they exhibit, consistent, predictable responses to repeated inputs. In those cases where we feel that a system can be described deterministically, it is a simple matter to modify the formal definition to account for this deterministic behavior. We simply add the notion of state by defining a set \( C \) as a state set or state space of S. We then can define a system, S, as a function:

\[ S : C \times X \rightarrow Y \]

We can use this concept to describe, say, a joystick with selectable gain. The state set, C, captures the notion of the gain selection where when a switch is in one position, output of high amplitude is produced while, for the very same input, if the switch is set to another setting, output of lower amplitude is produced.

The description of a HMS or a component as a system is really quite useless unless we can talk about how systems can be connected to form more complex ones, and how complex systems can be decomposed into simpler units. These ideas are conceptually quite simple, for to connect two systems together, we need only apply an output of the one to an input of the other. As Mesarovic and Takahara point out, however, formalizing this intuitive idea is quite a bit more complex. To do it, we must first define a class of connectable systems, that is, systems in which some inputs and outputs are available for connecting with other systems. Practically speaking, we can use this set to refer to those physical components which can be connected by means of wires, mechanical linkages, communication channels, and the like.

Let \( A_1, A_2, \ldots, A_n \) be input alphabets, let \( B_1, B_2, \ldots, B_n \) be output alphabets, let \( C_1, C_2, \ldots, C_n \) be sets of states, and let \( T \) be a time set. We will now define a set of state, input, and output objects for n connectable
systems:

$$\left( \forall i \in \mathbb{N} \right) \left( \forall mi \in \mathbb{N} \right) \left( \forall ni \in \mathbb{N} \right)$$

$$X_i = X_{i1} \times X_{i2} \times \ldots \times X_{i<mi>} = X_i' \times X_i''$$

where $$\left( \forall j \in \mathbb{N} \right) X_{ij} \subseteq \{x_{ij} | x_{ij} : T \rightarrow A_i \}$$

and where the component sets of $$X_i'$$ are not available for connection but the component sets of $$X_i''$$ are available for connection

$$Y_i = Y_{i1} \times Y_{i2} \times \ldots \times Y_{i<ni>} = Y_i' \times Y_i''$$

where $$\left( \forall j \in \mathbb{N} \right) Y_{ij} \subseteq \{y_{ij} | y_{ij} : T \rightarrow B_i \}$$

and where the component sets of $$Y_i'$$ are not available for connection but the component sets of $$Y_i''$$ are available for connection

SYSTEMS is the set of connectable systems:

$$\text{SYSTEMS} = \{ S_i | \left( \forall i \in \mathbb{N} \right) S_i : C_i \times (X_i' \times X_i'') \rightarrow (Y_i' \times Y_i'') \}$$

For each member of SYSTEMS, $$S_i : C_i \times (X_i' \times X_i'') \rightarrow (Y_i' \times Y_i'')$$, $$C_i$$ is the state set. The input object, $$X_i = (X_i' \times X_i'')$$, is the cartesian product of $$mi$$ component sets and is therefore $$mi$$-channel. Some of the inputs, $$X_i'$$, are not available for connection with other systems while the remainder, $$X_i''$$, are. The output object, $$Y_i = (Y_i' \times Y_i'')$$, is $$ni$$-channel, some channels, $$Y_i'$$, not available for connection, others, $$Y_i''$$, available for connection. Note that any object, $$X_i'$$, $$X_i''$$, $$Y_i'$$, or $$Y_i''$$, may be null ($\emptyset$) and, formally speaking, there is nothing to prevent all of the objects from being null.

Now we may define three connection operators on SYSTEMS to allow the formal description of system interconnections.
Let CAS be a function defined:

\[ \text{CAS : SYSTEMS} \times \text{SYSTEMS} \rightarrow \text{SYSTEMS} \]

\[ \text{Sk} = \text{CAS}(S_i, S_j) \leftrightarrow \]

\[ S_i : C_i \times X_i \rightarrow (Y_i' \times Y_i'') \]

\[ S_j : C_j \times (X_j' \times X_j'') \rightarrow Y_j \]

\[ \Rightarrow X_j'' = Y_i'' \]

\[ S_k : C_k \times (X_i \times X_j') \rightarrow (Y_i' \times Y_j) \]

\[ \Rightarrow (\exists f_k : C_i \times C_j \rightarrow C_k) \Rightarrow f_k \text{ is a surjection} \]

\[ (c_k, (x, x'), (y', y)) \in S_k \leftrightarrow \]

\[ (\exists z)(c_i, x, (y', z)) \in S_i \& (c_j, (x', z), y) \in S_j \]

\& f_k(c_i, c_j) = c_k \]

\[ \text{Sk} = \text{CAS}(S_i, S_j) \] is a system formed by connecting \( S_i \) and \( S_j \) in cascade. That is, one or more output channels of \( S_i \) are input channels to \( S_j \) and the state set of \( S_k \) has an onto relationship with the cartesian product of \( C_i \) and \( C_j \) so that states of \( S_i \) and \( S_j \) can be related to states of \( S_k \). The cascade connection is illustrated in Figure 2.

Let PAR be a function defined:

\[ \text{PAR : SYSTEMS} \times \text{SYSTEMS} \rightarrow \text{SYSTEMS} \]

\[ \text{Sk} = \text{PAR}(S_i, S_j) \leftrightarrow \]

\[ S_i : C_i \times X_i \rightarrow Y_i \]

\[ S_j : C_j \times X_j \rightarrow Y_j \]

\[ \Rightarrow X_i = X_j \]

\[ S_k : C_k \times X_i \rightarrow (Y_i \times Y_j) \]

\[ \Rightarrow (\exists f_k : C_i \times C_j \rightarrow C_k) \Rightarrow f_k \text{ is a surjection} \]
\((ck,x,(y,y')) \in Sk \iff (ci,x,y) \in Si \land (cj,x,y') \in Sj \land fk(ci,cj)=ck\)

\(Sk = \text{PAR}(Si,Sj)\) is a system formed by connecting \(Si\) and \(Sj\) in parallel. \(Si\) and \(Sj\) are systems which share one or more input channels. The parallel connection is shown in Figure 3.

Let \(FDB\) be a function defined:

\[ FDB : \text{SYSTEMS} \rightarrow \text{SYSTEMS} \]

\(Sk = FDB(Si) \iff Si : Ci \times (X_1 \times X_2) \rightarrow (Y_1 \times Y_2) \quad \downarrow X_2 = Y_2 \)

\(Sk : C_2 \times X_1 \rightarrow Y_1 \quad \downarrow (\exists f_k : C_1 \rightarrow C_2) \downarrow f_k\) is a surjection

\((ck,x,y) \in Sk \iff (\exists z) (ci,(x,z),(z,y)) \in Si \land fk(ci)=ck\)

\(Sk = FDB(si)\) is a system formed by closing a feedback loop of \(Si\). Figure 4 illustrates the feedback connection.

Let \(Si\) and \(Sk\) be elements of \(\text{SYSTEMS}\). \(Si\) is a subsystem of \(Sk\) if:

\((\exists Sj \in \text{SYSTEMS}) \downarrow Sk = \text{CAS}(Si,Sj) \quad \lor Sk = \text{PAR}(Si,Sj) \quad \lor Sk = FDB(Si)\)

Also, \(Si\) is a subsystem of \(Sk\) if \(Si\) is a subsystem of \(Sj\)
Figure 2

Cascade Connection
Figure 3
Parallel Connection
Figure 4
Feedback Connection
and Sj is a subsystem of Sk.

To this point we have defined a system $S : C \times X \rightarrow Y$ using a functional description. An alternate description of a system is a constructive specification which is defined by two sets:

$$D = \{ d \mid (\forall t, t' \in T \rightarrow t < t') \quad d : C \times X \rightarrow C \}$$

$$L = \{ l \mid (\forall t \in T) \quad l : C \times A \rightarrow B \}$$

$D$ is a set of state transition functions and $L$ is a set of output functions. They comprise a constructive specification because we can use them to construct elements of $S$. That is:

$$(\forall s = (c, x, y) \in S) \quad (\forall t \in T)$$

$$l \ (d \ (c, x)) = y(t)$$

$$t \ 0 \ 0$$

(Recall that $0$ is the least element of $T$.)

A state transition function (an element of $D$) tells us what state results when a system is started in a particular initial state and given an input over a time interval. An output function (an element of $L$) tells us what output is produced by the system in such a case. Together they allow us to construct elements of $S$ which we call behaviors.

In the case where $T$ is a discrete time set, we may call the state transition function a next state function for, given a current state and input, it tells us the resulting state at the next value of $T$. For example, suppose that $T$ is the set of non-negative integers, $T = \{ 0, 1, 2, 3, \ldots \}$. Then we might have, in the time-invariant case:
D = \{ d \}

d : C \times A \rightarrow C

and:

c(t+1) = d( c(t), x(t) )

In any behavior of a system, that is, for any \( s = (c,x,y) \in S \), we denote the state at time \( t \) by:

c(t) = d( c, x )

0 0

t t

We can also develop some notation for restricting systems to time intervals. That is:

\[
S : C \times X \rightarrow Y
\]

\[
t' t' t'
\]

\[
t t t t t
\]

where:

\[
C = \{ c' \mid (\forall s = (c,x,y) \in S) c' = d( c, x ) \}
\]

\[
t t 0 0
\]

and:

\[
(c',x',y') \in S
\]

\[
t'
\]

\[
\leq
\]

\[
t
\]
This notation allows us to refer to particular system behaviors over particular intervals of time, which will often be useful in the ensuing development.

3.7.1.1 Events -

The definition for an event comes directly from the definition of the constructive specification for a system.

Let \( S : C \times X \rightarrow Y \) be a system defined on the time set \( T \) which is linearly ordered by the relation \( \leq \), having least element \( 0 \). Let \( D \) be a family of state transition functions for \( S \):

\[
D = \{ d : (\exists t, t' \in T \cap t \leq t') \rightarrow d : C \times X \rightarrow C \ |
\]

Let \( \text{TRANS} \) be the set of all possible state transitions of \( S \). That is:

\[
\text{TRANS} = \{ (c, c') | (\exists s = (c, x, y) \in S) (\exists t', t'' \in T \cap t' \leq t'')
\]

\[
(\forall t \in T) d (c, x) = c' \land d (c, x) = c'' \land c' \neq c''
\]

Then an event is defined as a subset of \( \text{TRANS} \):

\[
\text{EVENT} \subseteq \text{TRANS}
\]
Let $S : C \times X \rightarrow Y$ as before and let $s$ be an element of $S$. The event EVENT occurred at time $t''$ (*event(EVENT,s,t'')*) if and only if:

$$\forall t' \in T \rightarrow t' \leq t'' \rightarrow (\forall t \in T \quad t \neq t')$$

$$d(c,x) = c' \land d(c',x) = c'' \land (c',c'') \in EVENT$$

3.7.2 Specific Defined Terms

The formal definitions to this point have given us the ability to describe an HMS or its components in terms of sets of inputs, outputs, and states. While this level of description provides a very general yet potentially a very precise way to talk about HMS's and classes of HMS's, we really do not yet possess the ability to talk about how HMS's plan and organize their activities.

Towards that end we will now present a set of specific defined terms that we can use to describe these higher level aspects of HMS's. Specifically, we introduce in this section the concepts of goals, procedures, and knowledge bases.

3.7.2.1 Goals

According to Meister (1976), purposiveness is a common element in all human-machine system definitions. He goes on to say that the HMS is purposefully directed to produce specified outputs on the basis of anticipated inputs, that the operators in HMS's are governed by overall goals, and that the significance of operator behavior can be determined only in relation to these goals.
Kelley (1968) says that the conception of and selection among goals is typical of living systems (and we consider an HMS to be a living system since it contains living components). He defines a goal as any possible future state selected from two or more alternatives.

This is a fairly common definition of goal. However it suffers from the same means-ends problem that we worry about in social and political life. That is, the end does not always justify the means. In HMS terminology, we would say that it is not enough for a system to simply arrive in a desired final state in some reasonable period of time. The intervening behavior must also be acceptable.

Miller (1974) recognizes and avoids this problem by defining a goal image as a picture of a task well done. This notion can capture not only the attainment of the desired final state, but the intervening behavior of the system as well. For our formal definition of a goal we will use a perspective very similar to Miller's.

A goal is defined as a subset of all the behaviors that a system can exhibit. That is, let \( S : C \times X \rightarrow Y \) be a system. Then a goal, GOAL, may be defined:

\[
\text{GOAL} \subseteq \{ s \mid s \in S & t, t' \in T \rightarrow t < t' \}
\]

Like most definitions of goals, this one includes the notions of initial and final states, i.e. state at time \( t \) and state at time \( t' \), but unlike most other definitions, it explicitly considers intervening behaviors as well. GOAL is achieved in a behavior, \( s \), of \( S \) if and only if:

\[
(\exists t, t' \in T \rightarrow t < t') \rightarrow s \in \text{GOAL}
\]

Operationally speaking, a goal is not generally defined by enumeration, that is by specifying each of the elements of the set. Rather, it is defined by a predicate, a statement characterizing the elements of the set.
For example, in landing an aircraft we can think in terms of a number of goals that the aircrew must be concerned with. The overall goal, that of getting the aircraft on the ground in the right place at the right time, may be defined by a predicate characterizing those behaviors in which the initial state of the aircraft is in the air, in which the final state of the aircraft is at one of a particular points on the ground, and the intervening behavior corresponds to specifications developed by the aircraft's manufacturer, the air traffic controller in charge of the flight, and regulatory agencies concerned with aviation.

3.7.2.2 Procedures -

The goal-directed activities of HMS's are often quite complex and many of them (those dealing with emergency conditions, for example) are often un-rehearsed and unfamiliar to the human operators. Consequently, HMS's require a certain amount of assistance in guiding their activities.

Authors commonly use the term procedure to describe the sequencing of activities to achieve goals. Meister and Rabideau (1965) define a procedure as step-by-step instructions for performing operating and maintenance tasks. Van Cott and Kinkade (1972) define a procedure as a step-by-step series of activities.

Other authors use slightly different yet equivalent terminology. Johannsen and Rouse (1979), for example define a protocol as a detailed explanation of activities. Schank and Abelson (1977) define a script as a structure which describes appropriate sequences of events in a particular context. Response programs are defined by Gartner and Murphy (1976) as preplanned or previously established performance objectives which govern the execution of anticipated crew tasks. Miller (1971), too, uses the term program in defining a task as a series of goal-directed transactions controlled by one or more programs.

If we substitute the word procedure for the words protocol, script, and program in the above paragraph, we will see a consistent use of the term to describe entities used to direct the activities of a system in achieving a goal. Given the goal-directed nature of HMS's it is clear that such a concept is very important in the development of
an HAM theory.

Let PALPH be an alphabet of symbols. PALPH* is defined as the set of all strings of finite length formed from symbols in PALPH. A procedure, proc, is defined:

$$\text{proc} = ( \text{pname}, \text{pstring}, \text{PP} )$$

where:

$$\text{pname} \in \text{PALPH}^*$$

$$\text{pstring} \in \text{PALPH}^*$$

pname is the name of the procedure and pstring is a string of symbols that corresponds to our intuitive notion of a procedure as a linguistic structure consisting of symbols we call steps or instructions.

Let PSTRING be defined:

$$\text{PSTRING} \subseteq \text{PALPH}$$

$$\text{PSTRING} = \{ \text{p} \mid \text{p} \text{ is a symbol in pstring} \}$$

and let PROCSTATES be defined:

$$\text{PROCSTATES} = \{ \text{inactive, executing, suspended} \}$$

Let AP be an input alphabet, let BP be an output alphabet, and let CP be a set of states defined:

$$\text{CP} \subseteq \text{PROCSTATES} \times \text{PSTRING}$$

CP's definition may include additional component sets as well. Let TP be a time set. PP may then be defined:

$$\text{PP} : \text{CP} \times \text{XP} \rightarrow \text{YP}$$
where:

\[ X_P \subseteq \{ x \mid x : TP \rightarrow AP \} \]
\[ Y_P \subseteq \{ y \mid y : TP \rightarrow BP \} \]

XP and YP may be multi-channel.

PP is a special kind of system called a procedure processor. It is distinguished by the fact that its state object consists of at least two component sets. One consists of the symbols in pstring, the other component set is PROCSTATES. PP has a few additional properties. Let start be the first symbol in pstring and let null be a distinguished element of BP. We then have the following:

\[ (\forall pp = ((cp',cp''),xp,yp) \in PP)(\forall t \in TP) \]
\[ cp'(t) = \text{inactive} \Rightarrow sp''(t) = \text{start} \]
\[ cp'(t) \neq \text{executing} \Rightarrow yp(t) = \text{null} \]

\[ (\forall t',t'' \in TP \downarrow t' < t'') \]
\[ cp''(t') \neq cp''(t'') \Rightarrow (\exists t \in TP_{t'}^{t''}) \downarrow \]
\[ cp'(t) = \text{executing} \]

Ignoring the PROCSTATES component of CP for the moment, PP may be regarded to be the same as any other system with the exception that the PSTRING component of the state object is comprised of symbols from the string of a procedure. Like any other system, PP transitions from state to state. When the current state is p, a symbol in pstring, we say that PP is processing p. The PROCSTATES component of CP, though, governs this "normal" behavior. First of all, no change in the PSTRING component of state may occur unless the PROCSTATES component is executing. In addition, no output other than the value null may be produced unless PP is executing. Also, if PP is in the
inactive state, then the PSTRING component of state is the first symbol in pstring. The PROCSTATES component of state may be controlled by a particular input channel. This will be developed further in Axiom 5.

This formal definition of a procedure captures a number of intuitive notions about procedures. Suppose we are interested in a pilot and an aircraft. The pilot may have a number of procedures to use, either written in some handbook, memorized, or internalized (so that when needed the procedure is performed automatically without explicit reference to instructions or steps). Before the pilot begins executing say, a routine landing procedure, we can call that procedure inactive, for it is not of immediate concern to the pilot. When the procedure is initiated, it may be referred to as active, and the pilot processes (consciously or subconsciously) the steps or instructions one at a time, making decisions and performing actions to comply with the procedure. While active, the procedure may be executing or suspended. When executing, inputs are attended to and actions are produced. If interrupted, for example, by some subsystem failure that must be attended to, we say that the landing procedure is suspended and no actions regarding it are taken. At some later time, though, it may be resumed, at which time it is described as executing again. In summary, the procedure is initially inactive. When started it becomes active and may be executing or suspended. When finished, it becomes inactive again.

In order to fully describe a system in terms of procedures, two additional structures are needed: demultiplexers and multiplexers. A demultiplexer converts one input channel into multiple equivalent channels. A multiplexer converts multiple output channels into a single channel.

Let \( A \) be an alphabet, let \( CD \) be a set of states, and let \( T \) be a time set. Let \( X \) be an input object:

\[
X \subseteq \{ x \mid x : T \to A \}
\]

Let \( AP_1, AP_2, \ldots, AP_m \) be \( m \) alphabets. Let \( TP_1, TP_2, \ldots, TP_m \) be time sets such that:

\[
(\forall i \in \mathbb{N}) \quad T \geq TP_i
\]
and:

\[(\forall i \in \mathbb{N}) \text{ } X_{pi} \subseteq \{ x : x : T_{pi} \rightarrow A_{pi} \} \]

A demultiplexer may then be defined:

\[D_{M} : C_{D} \times X \rightarrow (X_{p1} \times X_{p2} \times \ldots \times X_{pm})\]

such that:

\[(\forall dm = (c_{d}, x, (x_{p1}, x_{p2}, \ldots, x_{pm})) \in D_{M})(\forall t \in T)\]

\[x(t) = x_{p1}(t) = x_{p2}(t) = \ldots = x_{pm}(t)\]

That is, a demultiplexer makes \(m\) copies of an input and converts time sets. The outputs of the demultiplexer \((X_{p1}, X_{p2}, \ldots, X_{pm})\) may be inputs to procedure processors or other systems.

Let \(B_{p1}, B_{p2}, \ldots, B_{pm}\) be alphabets. Let \(C_{M}\) be a set of states. Let \(T_{p1}, T_{p2}, \ldots, T_{pm}\), be time sets (not necessarily the same as those defined above). Also let:

\[(\forall i \in \mathbb{N}) \text{ } Y_{pi} \subseteq \{ y : y : T_{pi} \rightarrow B_{pi} \} \]

Let \(B\) be an alphabet, let \(C_{M}\) be a set of states, and let \(T\) be a time set such that:

\[(\forall i \in \mathbb{N}) \text{ } T \subseteq T_{pi} \]

Also let:
A multiplexer may then be defined:

\[ MX : \mathbb{C}M \times (YP_1 \times YP_2 \times \ldots \times YP_m) \to Y \]

such that:

\[
(\forall \text{ } m x = (cm, (yp_1, yp_2, \ldots, yp_m), y) \in mx)(\forall \text{ } t \in T)
\]

\[
(\exists j \in \mathbb{N}) \text{ } y(t) = yp_j(t)
\]

That is, a multiplexer converts multiple outputs to a single output. Of course, the single channel can take on only one value at a time, so there must be arbitration. The inputs to the multiplexer (YP_1, YP_2, ..., YP_m) may be the outputs of procedure processors or other systems.

We may now define a procedural representation of a system. Let PROCURURES = \{ proc1, proc2, ..., proc_m \} be a set of m procedures where:

\[
(\forall \text{ } i \in \mathbb{N}) \text{ } proc_i = (pname_i, pstring_i, PP_i)
\]

and:

\[ PP_i : \mathbb{C}Pi \times XPi \to YPi \]

as before. Let SYSTEMS = \{ S_1, S_2, ..., S_n \} be a set of n connectable systems where:

\[
(\forall \text{ } i \in \mathbb{N}) \text{ } Si : Ci \times Xi \to Yi
\]
Now let \( PR : CPR \times XPR \rightarrow YPR \) be a system where:

\[
(\forall i \in N) \quad PP_i \text{ is a subsystem of } PR
\]

\[
(\forall i \in N) \quad Si \text{ is a subsystem of } PR
\]

and:

\[
CPR \subseteq CP_1 \times CP_2 \times ... \times CP_m \times C_1 \times C_2 \times ... \times C_n
\]

Let \( pr = (cpr, xpr, ypr) \in \) and let:

\[
pp_1 = (c_{p1}, x_{p1}, y_{p1}) \in PP_1, \\
pp_2 = (c_{p2}, x_{p2}, y_{p2}) \in PP_2, \\
..., pp_m = (c_{pm}, x_{pm}, y_{pm}) \in PP_m,
\]

\[
s_1 = (c_{1}, x_{1}, y_{1}) \in S_1, s_2 = (c_{2}, x_{2}, y_{2}) \in S_2, ..., \\
s_n = (c_{n}, x_{n}, y_{n}) \in S_n
\]

be behaviors of the subsystems of \( PR \) that correspond to \( pr \).

Then:

\[
cpr = (c_{p1}, c_{p2}, ..., c_{pm}, c_1, c_2, ..., c_n)
\]

Now let \( S : C \times X \rightarrow Y \) be a system such that:

\` 
\( S \in \text{SYSTEMS} \)

Then \( PR \) is a procedural representation of \( S \) if and only if:

\[
(\exists \ pr = (cpr, xpr, ypr) \in PR)(\exists \ s = (c, x, y) \in S) \quad xpr = x \land ypr = y
\]
Let:

\[ pr = (cpr, xpr, ypr) \in PR \]

and:

\[ s = (c, x, y) \in S \]

Then \( pr \) is a procedural representation of \( s \) if and only if:

\[ xpr = x \& ypr = y \]

A procedural representation of a system is another system (not containing the original as a subsystem) composed partly of procedure processors which imitates the original system under at least one set of circumstances. A procedural representation of a particular behavior of that system is a behavior of the procedural representation of the system which is identical, in terms of input and output, to the original system's behavior. In other words, a procedural representation is an alternate representation incorporating procedures.

Let:

\[ pr = (cpr, xpr, ypr) \in PR \]

and:

\[ s = (c, x, y) \in S \]

such that \( pr \) is composed of the behaviors of its subsystems as defined above and:

\[ opr = (cp1, cp2, ..., cpm, c1, c2, ..., cn) \]
We will use the following conventions in speaking about $s$:

$\forall i \in \mathbb{N} \exists t \in \mathbb{T}$

- proci is inactive in $s$ at time $t$ $\iff$
  \[ cpi(t) = (q,p) \land q = \text{inactive} \]
- proci is active in $s$ at time $t$ $\iff$
  \[ cpi(t) = (q,p) \land q \neq \text{inactive} \]
- proci is executing in $s$ at time $t$ $\iff$
  \[ cpi(t) = (q,p) \land q = \text{executing} \]
- proci is suspended in $s$ at time $t$ $\iff$
  \[ cpi(t) = (q,p) \land q = \text{suspended} \]

A procedural representation is a structure to organize knowledge about a system. It gives us another way to describe that system's behavior in terms of procedures and, in that sense provides an "explanation" for the behavior of the system. If we have a procedural representation for a particular behavior of a system, we can describe that system at any time in terms of inactive, active, executing, and suspended procedures.

3.7.2.3 Knowledge Bases -

Since many of the activities that take place in human-machine systems are information processing activities, HMS's must posses means for storing information. According to Kelley (1968), information storage in HMS's is accomplished by imposing special structure on some physical object from which a pattern of information can be re-created for later use.

Van Cott and Kinkade (1972) state that the memory subsystem of the human operator provides long and short term storage of encoded information. Johannsen and Rouse (1979) define long-term memory as a knowledge base of facts, models, and procedures. Miller (1971) writes that
long-term memory stores procedures (programs, in his terminology). In addition, memory contents are used in input processing, complex mediating processes, and organization and selection of output processes. In a later paper (Miller, 1974) he uses slightly different terminology in stating that the contents of storage are information about data and information about procedures.

Here we will use the term knowledge base to capture these concepts and provide a very simple definition. A knowledge base is simply a system capable of storing symbols representing data or procedures in its state object.

Let $S : C \times X \rightarrow Y$ be a system, defined on the input alphabet, $A$, the output alphabet $B$, the state set, $C$, and the linearly ordered time set $T$ with least element 0, and let $D$ and $L$ comprise a constructive specification for $S$. Any kind of information, be it data or procedures, simply consists of symbols of some kind that have particular interpretations in some given context. Therefore, let $\text{ALPH}$ be an alphabet of symbols and let $\text{ALPH} \subseteq B$. Let $\text{ALPH}^*$ be, as described before, the set of all strings of finite length over $\text{ALPH}$ and let string be an element of $\text{ALPH}^*$. We define $\text{STRING}$ as the set of all symbols in string. Clearly, $\text{STRING} \subseteq \text{ALPH} \subseteq B$.

We say that $S$ is a knowledge base capable of storing string if and only if:

$$(\forall p \in \text{STRING}) (\exists s = (c,x,y) \in S) (\exists t \in T) \quad t_1 = p$$

Informally speaking, a knowledge base stores information in the form of symbols. A system is a knowledge base capable of storing a string of such symbols, call it a datum or a procedure, if for some initial state and some input, the symbols in that string are produced as output. Since this ability depends on the state of the system, we say that the information is stored in the system's state object. The associated input that produces the output has two possible interpretations. First, it can be interpreted as the action of storing information in the
knowledge base, as when characters are printed on paper or when sounds are recorded on magnetic tape. The input is also interpreted as a retrieval signal which elicits the desired information from the knowledge base. The output is interpreted as the retrieved information. This information may serve as input to another system in which it may serve as a procedure to organize the activities of that system or as data, information used to supplement those activities.

Neither a goal nor a procedure is entirely symbolic, but we may represent a goal or a procedure symbolically. We say that a knowledge base is capable of storing a goal or a procedure if it is capable of storing its symbolic representation.

Examples of knowledge bases have already been mentioned, including magnetic tape devices and human memory. The interpretation of the state object, inputs, and outputs in the case of the magnetic tape are rather clear, for the devices have been designed by humans to function in a particular manner. The corresponding interpretations for human memory are not well defined, although they can be described in rather general terms.

3.7.2.4 Human-Machine Systems -

A human-machine system is defined as a triple:

\[ \text{HMS} = (S, \text{GOALS}, \text{PROCEDURES}) \]

where \( S : C \times X \rightarrow Y \) is a system, GOALS is a set of goals which may be used to describe the behaviors of \( S \) and PROCEDURES is a set of procedures which may be used to describe the behaviors of \( S \).

This definition of a human-machine system provides two levels of description. \( S \) is the first level. It provides a means for describing a human-machine system in terms of input, output, and state, but it gives us no way of explaining or interpreting specific behaviors. GOALS and PROCEDURES provide the second level of description. Behaviors of \( S \) may be interpreted in terms of the achievement or non-achievement of goals, and may be described as being produced by the execution of one or more procedures. That is, PROCEDURES, possibly in conjunction with additional associated subsystems, forms a procedural
description of $S$. One or more knowledge bases may be incorporated at either level of $S$. In other words, a knowledge base may be a subsystem of $S$ or it might be an associated subsystem in the procedural description of $S$.

The definition then provides a two-level structure for organizing knowledge about a human-machine system. $S$ is a structure for organizing knowledge into input, output, and state components. GOALS and PROCEDURES are structures for organizing outputs, outputs, and states into purposive behavior caused by the execution of procedures.

Notice that there is no formal requirement placed on this structure which prevents its interpretation in systems that do not contain human operators. This is due to the fact that a strictly formal definition of a human being is a rather difficult proposition. We will instead place a rather informal restriction on the structure by stating that human operators are present in all intended models of the emerging theory. In fact, we can state that all intended models possess one or more of the characteristics described in Chapter 2.

Note also that the definition of a human-machine system is a hierarchical one. That is, given HMS as defined above, we can define another human-machine system on a subsystem of $S$, a subset of GOALS, and a subset of PROCEDURES. It is then possible that a given human-machine system may contain one or more others.

3.8 Axioms

The axioms of a theory are the assumptions on which it is based. The axioms form the basis for all other theorems of the theory. Using the terms defined above, we will now state several axioms for a theory of human machine systems. All of the axioms are based on the fact that HMS is a human-machine system defined:

$$\text{HMS} = (S, \text{GOALS}, \text{PROCEDURES})$$

where:

$$S : C \times X \rightarrow Y$$
GOALS is a set of goals, and:

\[
\text{PROCEDURES} = (\text{proc1, proc2, \ldots, proc}_n)
\]

where:

\[
(\forall i \in N) \text{ proc}_i = (N_i, \text{pstring}_i, P_i)
\]

and:

\[
\text{PP}_i : C_{P_i} \times X_{P_i} \rightarrow Y_{P_i}
\]

Furthermore, \(C_{P_i}\) may include more components than just \text{PROCSTATES} and \text{PSTRING}_i and \(X_{P_i}\) and \(Y_{P_i}\) may be multi-channel.

Also, let \(PR\) be a procedural representation of \(S\) which includes all of the procedures in \text{PROCEDURES}.

3.8.1 Axiom 1 -- Procedural Plans

A human-machine system develops procedural plans to achieve goals.

Let \text{EVENTS} be a set of events defined for \(PR\) and let \text{EVENTNAMES} be a set of names, one name for each event in \text{EVENTS}. Let \text{PROCNames} be set of the procedure names. That is \text{PROCNames} = \{N_1, N_2, \ldots, N_n\}. Let \text{Pplans} be a set of procedural plans defined:

\[
\text{Pplans} = \text{TT}(\text{EVENTNAMES} \times \text{PROCNames})
\]

That is, a procedural plan is a set of ordered pairs in which the first element of each ordered pair is an event name and the second element is a procedure name.
For any human-machine system, HMS, there is a relation that associates each goal in GOALS with one or more procedural plans:

\[ \text{GPLAN} \subseteq \text{GOALS} \times \text{PPLANS} \]

such that:

\[ (\forall \text{GOAL} \in \text{GOALS})(\exists (G, \text{PLAN}) \in \text{GPLAN}) \]

\[ G = \text{GOAL} \]

Informally speaking, for any goal there is plan to achieve it. Ideally, the human-machine system can determine what that plan is and implement it.

3.8.2 Axiom 2 - Activation of Procedures

In a human-machine system, procedures are activated according to procedural plans.

Let s be an element of S and let pr be an element of PR such that pr is a procedural representation of s. Then:

\[ (\forall t \in T)(\forall i \in N) \]

\[ \text{proci is active in s at time } t \leftrightarrow \]

\[ (\exists (\text{GOAL}, \text{PLAN}) \in \text{GPLAN})(\exists (e,n) \in \text{PLAN}) \]

\[ (\exists t' \in TP \mid t' < t) \to \text{event}(e,pr,t') \land n=\text{Ni} \]

That is, if a procedure is active, then there must be a goal whose procedural plan contained an entry in which the procedure was associated with an event that has already occurred. Another way of saying this is that in a human-machine system, plans are made to achieve goals and these plans require that specific procedures be invoked to handle specific events. When an event contained in a
procedural plan occurs, the corresponding procedures become active. The success of the procedural plan process depends on the ability to foresee events that will occur and in the ability to select procedures to process the events.

3.8.3 Axiom 3 - Procedure Competition

In a human-machine system, procedures compete for output channels.

Let procj and prock be procedures in PROCEDURES and let MX be a multiplexer such that MX is a subsystem of PR:

$$MX : CM \times (YP_j \times YP_k) \rightarrow Y$$

Then we say that procj and prock compete for output channel Y.

As was mentioned previously, a multiplexer converts multiple output channels into a single channel. At any time, the single output channel may take on only a single value and that value must be the same as one of the input channels. Consequently, the other outputs are "lost." This is referred to as procedure competition for an output channel.

3.8.4 Axiom 4 - Limitations on Executing Procedures

In general, at any time, all active procedures are not executing.

Let s be an element of S and let pr be an element of PR such that pr is a procedural representation of s. Then we have:

$$\forall t \in T (\exists i \subseteq N)$$

n

proc is active in s at time t

& proc is suspended at time t
Due to the fact that a procedure is competing for an output channel and the output of another procedure is currently chosen for the output of the multiplexer, or for some other reason, that procedure may be active but not executing.

3.8.5 Axiom 5 - The Executive Procedure

In a human-machine system, an executive procedure controls the activation and execution of other procedures.

Let proce and proci be defined:

\[
\text{proce} = (\text{pnamee}, \text{pstringe}, \text{PE}) \in \text{PROCEDURES}
\]

\[
\text{proci} = (\text{pnamei}, \text{pstringi}, \text{Pi}) \in \text{PROCEDURES}
\]

& proci \neq proce

where:

\[
\text{PE} : \text{CPE} \times \text{XPE} \to (\text{YPE} \times \text{YPEi})
\]

\[
\text{YPEi} \subseteq \{ y \mid y : \text{TP} \to \{\text{enable,disable}\} \}
\]

\[
\text{Pi} : \text{CPi} \times (\text{XPi}' \times \text{XPi}'') \to \text{YPi}
\]

\[
\text{XPi}' = \text{YPEi}
\]

and we can form a system PEi:

\[
\text{PEi} = \text{CAS(PE}, \text{Pi})
\]

\[
\text{PE} : (\text{CPE} \times \text{CPi}) \times (\text{XPE} \times \text{XPi}'') \to (\text{YPE} \times \text{YPi})
\]

such that:

\[
(\forall \text{pei }= ((\text{cpe}, \text{cpi}), (\text{xpe}, \text{xpi}''), (\text{ype}, \text{ypi})) \in \text{PEi})
\]
A procedure $e$ is called the executive procedure. It is cascaded by means of a single channel to each other procedure and through that channel it controls the PROCSTATES component of that procedure. Recall that, in turn, the PROCSTATES component of a procedure's state object controls the PSTRING component, that component consisting of the symbols in the procedure's symbol string. Therefore, the executive controls the activation and
disabled state of each other procedure.
execution of the other procedures. It activates an inactive procedure by sending a value of enable. It suspends an executing procedure by sending a value of disable. It resumes the execution of a suspended procedure by sending a value of enable. Multiple procedures may at any time be active. Whether they are executing or suspended at that time is controlled by the executive.

In the literature, such a situation is often called time-sharing. According to Alluisi (1967), time-sharing is a characteristic of most if not all operational situations. Using somewhat different terminology, Meister (1976), states that tasks compete with each other for the operator's attention. Wickens (1978) writes that although we know little about how resources are allocated dynamically, we do know that that allocation process itself requires added attentional capacity.

Informally speaking, the executive procedure determines which procedures are allowed to execute at any time. Just how the executive does this is not completely understood, but Wickens (1978) believes that resources (subsystems) are distributed among tasks (procedures) according to priorities. Navon and Gopher (1979) write that the allocation is governed by supply-demand "economic" considerations, with, here, supply being provided by subsystems and demand originating from the active procedures.

In a sense, Axiom 5 may be likened to the time-sharing computer system analogy of HMS's given by Johannsen and Rouse (1979) and by Audley (Audley et al., 1978). In such analogies, activities compete for system resources in much the same manner as computer programs compete for computer system resources. This competition is mediated by a supervisory or executive program which allocates resources according to program priority and resource availability.

3.9 Logical Calculus

The logical calculus of a theory is that set of rules by which additional statements are derived from the axioms or assumptions. The logical calculus, then, determines the overall nature of the resulting theory and governs its growth.
The theory of human-machine systems for which we are developing these terms and axioms is primarily intended to be an empirical theory. Consequently, the logical calculus of this theory must be in part inferential. That is to say that new theorems in the theory will be based in part upon the axioms that have been presented above and in part upon data collected by observing actual and simulated HMS's.

Since the purpose of the structure we are developing is to serve as just the basis for a theory of human-machine systems, no attempt will be made here to pose and prove/disprove any specific theorems. We can, however, list a number of issues for which theorems might be posed to address. These include:

1. The nature of the executive procedure.
2. The subsystem requirements of the executive procedure.
3. The method by which the executive procedure determines which procedures are allowed to execute.
4. A taxonomy of systems and subsystems.
5. The kinds and characteristics of systems and subsystems.
6. A taxonomy of goals and procedures.
7. The kinds and characteristics of goals and procedures.
8. What characterizes sets of procedures that are inherently conflicting and whose simultaneous execution should be avoided.
9. What considerations should be made in designing procedures for HMS's.

3.10 Discussion

The objectives in Chapter 3 have been two. The first and more general objective has been to illustrate the formal development of a structure for organizing knowledge.
The second objective, consistent with the first, has been to produce a structure useful for organizing knowledge about human-machine systems. The process of attempting to achieve these objectives has led to a structure consisting of some defined terms and a set of axioms or assumptions for describing human-machine systems. Since the terms are defined on primitive objects, specifically, observable HMS inputs and outputs and abstract state sets, in a formal manner, they may be considered to be much more precise than the corresponding terms found in most of the literature.

In the opinion of the author, the development of this structure has achieved the goals set forth. It is now possible to speak of the resulting product as a reasonably formal structure for organizing knowledge, specifically one useful for precisely describing human-machine systems and posing and addressing HMS research questions. Of course it is now necessary to demonstrate that these claims are true, which is the purpose of the following two chapters. In Chapter 4 we will describe an interpretation of the terms and axioms developed above. In Chapter 5 some of the issues and implications raised by the structure will be discussed.

3.11 Summary

In this chapter the foundations for a theory of human-machine systems, including a set of formally defined terms and a set of axioms, were developed. The approach used was a formal, set-theoretic one to take advantage of the generality and explicitness of such an approach.

First, some preliminary set definitions and notational conventions were given. Observation terms based on observable inputs and outputs were briefly described. Some standard terms generally used in formal developments, including relations, functions, time sets, and systems were considered next.

The next sections were devoted to defining a human-machine system in terms of systems, goals, and procedures.

A system is a structure for describing human-machine systems and human-machine system components in terms of inputs, outputs, and states. Systems can be connected to form more complex systems or can be decomposed into simpler
systems.

A goal is a specification of what system behaviors are acceptable and which are not.

A procedure is a structure, linguistic in part, which guides the activities of a system.

Several axioms were stated to provide a set of basic assumptions for a theory of human-machine systems. A human-machine system develops procedural plans to achieve goals and follows these plans by activating procedures to handle specific events. In general, procedures compete for output channels of the system and consequently not all active procedures are allowed to execute. The activation, execution, suspension, and resumption of procedures is controlled by the executive procedure.
The rules of formalization of theories discussed in Chapter 1 require the interpretation of those theories by models. While the structure of terms and axioms developed in Chapter 3 does not strictly qualify as a theory, for no theorems yet exist, there is still a need to interpret it in keeping with the spirit of formalization. In this chapter, a structure which interprets the terms and axioms will be presented. To be completely accurate, this structure could not be called a model, for a model must model a theory. For this purpose, the structure will be called an interpretation.

The interpretation chosen to illustrate and explain the terms and axioms is a bicycle and rider stopping at a stop sign. The description of the bicycle and rider is not intended to be a highly accurate one. Instead, several simplifying assumptions have been made to make the discussion feasible. The result is a structure which is rather naive (as far as stopping a bicycle goes), always simple, and often inaccurate, but one which has been carefully developed to illustrate the points raised and defined in Chapter 3.

To present the interpretation, the first step will be to give an informal description of the rider and the bicycle and to speculate on how one could go about describing the pair in more precise terms. Then several descriptions will be provided to examine them from differing levels of abstraction. They will be described as a simple system, as a two-component system composed of a human subsystem and a machine subsystem, and finally as a human-machine system, using the concepts of goals and procedures. The fundamental purpose is to investigate how more complex structures can be used to organize knowledge about the bicycle and rider to answer more specific questions, raise more interesting issues, and provide more explanatory power.
4.1 A Bicycle and Rider

The interpretation consists of a hypothetical bicycle and its rider stopping at a stop sign. The bicycle has two speeds or gears (1 and 2) which are selected by the rider. Front and rear caliper brakes are actuated by levers mounted on the bicycle's hand grips with the left lever controlling the rear brake and the right lever controlling the front brake. The rider cannot simultaneously actuate both the front brake and the shift mechanism. Forward movement is imparted to the bicycle by rotating the pedals.

In order to stop the bicycle, the rider must stop pedaling, apply the rear brake, and apply the front brake until the bicycle comes to a complete stop. In addition, since the bicycle is much easier to start up again if it is in first gear and since the rider normally rides in second gear, the bicycle must be downshifted during the stop. Since this is usually the last operation performed in the stop, front braking must be interrupted for the downshift to be made. Front braking may then be resumed.

Our immediate objective is to describe the behavior of the pair when stopping at a stop sign in as precise a manner as possible, and to do it in such a way as to give insight into why the bicycle finally stops (if it does), where it stops, and how the rider made decisions and coordinated actions to bring about the stop.

The first step we can take is to collect some quantitative data by observing the bicycle and rider stop at a stop sign. We can time the behavior with a stop watch, starting at time 0, the beginning of observations, and measure and record the distance of the bicycle from the stop sign (in feet) and its velocity (in feet per second, fps) each second. To describe this we need the following sets:

\[
\text{TIME} = \{ t \mid t \in \mathbb{I} \text{ and } 0 \leq t \leq 16 \}
\]

\[
\text{DISTANCE} = \{ d \mid d \in \mathbb{I} \text{ and } -500 \leq d \leq 500 \}
\]

\[
\text{VELOCITY} = \{ v \mid v \in \mathbb{I} \text{ and } 0 \leq v \leq 30 \}
\]

We can now specify a structure to organize knowledge about the bicycle and rider stopping at the stop sign:
DATA = \{ data \mid data : T \rightarrow \text{DISTANCE x VELOCITY} \}

The structure consists of a universe formed by the union of the sets TIME, DISTANCE, and VELOCITY, and the relations of the structure consist of the elements of the set DATA. Each element of DATA provides one possible account of the situation described above. We could build up the set by repeated observations or by taking just a few and developing a predicate characterizing the elements of the set. Table 1 lists one element of DATA. We can induce from this that when considering elements from DATA, we will notice constantly decreasing distance and velocity values. We might also notice that the rate of velocity decrease changes, possibly according to some pattern.

The advantage of such a structure is that it provides a precise way to describe bicycle/rider behaviors. What it does not do is explain why the behaviors were observed. For instance, why does velocity begin dropping when it does? Why does velocity decrease at the rates it does? Could either or both of these be explained in terms of some input to the bicycle or rider? We can explore these and other questions by making some assumptions about how a bicycle and rider work and describing them as an input/output/state system.

4.2 The Bicycle and Rider as a System, S

We notice from element data that we can describe the distance at time $t$ as the distance at time $t-1$ less the velocity at time $t-1$, and that velocity itself drops at rates of 0, 1, 2, or 3 fps each second. If we are willing to assume that this is always true (and from our knowledge of physics, at least the former does not seem to be an unreasonable assumption) we can use this information to develop a somewhat more complex structure to describe the bicycle and rider. The structure we will use is a system as defined in Chapter 3.

The system will be named S and is diagrammed in Figure 5. S's outputs are just distance and velocity as before, but we include an input as well: the visual scene. Actually, to be realistic, input based on what a rider could see would be much too complicated to address here.
### TABLE 1

<table>
<thead>
<tr>
<th>t</th>
<th>Distance</th>
<th>Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>298</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>269</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>211</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>183</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>156</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>131</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>87</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>68</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>52</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>39</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>29</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Instead we will consider the input to consist of only two possible values, null and stop-sign, in which stop-sign implies that the stop sign has been seen and null implies that it has not. In addition we need a state object to account for distance, velocity, and the rate at which velocity is decreasing. Therefore we can define $S$ by:

$$A = \{\text{null, stop-sign}\}$$

$$BD = \text{DISTANCE}$$

$$BV = \text{VELOCITY}$$

$$C = \text{MODE} \times \text{DISTANCE} \times \text{VELOCITY}$$

$$\text{MODE} = \{0,1,2,3\}$$

$$S : C \times X \rightarrow (YD \times YV)$$

$$X = \{x \mid x : T \rightarrow A \& (\exists ! t) x(t) = \text{stop-sign}\}$$

$$YD = \{y \mid y : T \rightarrow BD\}$$

$$YV = \{y \mid y : T \rightarrow BV\}$$

$S$ maps time functions of inputs (in which a stop-sign value appears only once in each function) to distance and velocity time functions. We now need to characterize $S$ either by some predicate or by a constructive specification. Since $T$ is a discrete time set and since we will assume that the state transition function and the output function are time-invariant, we will provide one-step state transition rules, predicates that are assumed to be true in $S$.

$$(\forall ((cm,cd,cv),x,(yd,yv)) \in S)$$

$$cm(0) = 0$$

$$(\forall t \in T - \{16\})(\exists t' \in T \& t' = t + 1) \rightarrow$$

$$cd(t') = cd(t) - cv(t)$$

$$cv(t') = \max(0,(cv(t) - cm(t)))$$

$$cm(t) = 0 \& x(t) \neq \text{stop-sign} \Rightarrow cm(t') = 0$$
\[ \text{cm}(t) = 0 \quad \& \quad x(t) = \text{stop-sign} \Rightarrow \text{cm}(t') = 0 \]

**The predicates for output are:**

\[ \text{yd}(t) = \text{cd}(t) \]
\[ \text{yv}(t) = \text{cv}(t) \]

But something is missing. The predicates for computing \( \text{cm}(t) \) are not complete. The fact is, even though we know that the MODE component of state affects the change in velocity, we cannot yet say just what affects that component itself. Nevertheless, \( s \), an element of \( S \) which satisfies our above statements and corresponds to the bicycle/rider data is presented in Table 2.

We could not complete the state transition rules because we have ignored the role of the rider in sensing, decision-making, and control. If we can decompose \( S \) into a bicycle and rider, possibly additional insight can be gained into \( S \)'s behavior.

4.3 A Decomposition of \( S \)

Following the definitions of systems and subsystems of Chapter 3, we can develop a new structure to describe \( S \) and, in turn, the bicycle and rider. This new structure will be based on the decomposition of \( S \) into a rider subsystem (RR) and a bicycle subsystem (BB). The decomposition is diagrammed in Figure 6.

In the remainder of this chapter, a standard notational convention will be used for naming sets of definition. Let \( MN, SS, \) and \( DD \) be systems (they could be subsystems, as well). Suppose that we want to define \( MN \). \( AMN \) is a general input alphabet. \( AMNSS \) is an input alphabet of symbols that \( MN \) can receive from \( SS \). \( BMN \) is a general output alphabet. \( BMNDD \) is an output alphabet of symbols that \( MN \) can send to \( DD \). \( CMN \) is a set of states and let \( T \) be a time set. Then we will define \( MN \) as:

\[
MN : CMN \times (XMN \times XMNSS) \rightarrow (YMN \times YMNDD)
\]
### TABLE 2

<table>
<thead>
<tr>
<th>t</th>
<th>cm(t)</th>
<th>cd(t)</th>
<th>cv(t)</th>
<th>x(t)</th>
<th>yd(t)</th>
<th>yv(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>298</td>
<td>29</td>
<td>null</td>
<td>298</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>269</td>
<td>&quot;</td>
<td>stop-sign</td>
<td>269</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>240</td>
<td>28</td>
<td>&quot; null</td>
<td>240</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;</td>
<td>211</td>
<td>27</td>
<td>&quot;</td>
<td>211</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>183</td>
<td>25</td>
<td>&quot;</td>
<td>183</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>&quot;</td>
<td>156</td>
<td>23</td>
<td>&quot;</td>
<td>156</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>&quot;</td>
<td>131</td>
<td>21</td>
<td>&quot;</td>
<td>131</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>&quot;</td>
<td>108</td>
<td>19</td>
<td>&quot;</td>
<td>108</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>87</td>
<td>16</td>
<td>&quot;</td>
<td>87</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>&quot;</td>
<td>68</td>
<td>13</td>
<td>&quot;</td>
<td>68</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>52</td>
<td>10</td>
<td>&quot;</td>
<td>52</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>39</td>
<td>9</td>
<td>&quot;</td>
<td>39</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>&quot;</td>
<td>29</td>
<td>8</td>
<td>&quot;</td>
<td>29</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>&quot;</td>
<td>20</td>
<td>5</td>
<td>&quot;</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>&quot;</td>
<td>12</td>
<td>2</td>
<td>&quot;</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>&quot;</td>
<td>7</td>
<td>0</td>
<td>&quot;</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>&quot;</td>
<td>5</td>
<td></td>
<td>&quot;</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 6
Decomposition of $S$
where:

\[ X_{MN..} \subseteq \{ x \mid x : T \rightarrow AM_{N..} \} \]
\[ Y_{MN..} \subseteq \{ y \mid y : T \rightarrow BM_{N..} \} \]

Then we say that \( X_{MN} \) is external input to \( MN \), \( XM_{N..}SS \) is input to \( MN \) from \( SS \), \( Y_{MN} \) is external output from \( MN \), and \( Y_{MNDD} \) is output from \( MN \) to \( DD \). An additional one-character code may be added to the name of a set to further distinguish it. For example, if there are two input channels to \( MN \) from \( SS \) we could have \( XM_{N..}SS1 \) and \( XM_{N..}SS2 \) or \( XM_{N..}SSA \) and \( XM_{N..}SSB \). Now we can define our subsystems.

\( RR \) is the subsystem with visual input. In addition, \( RR \) can sense velocity and the particular gear that the bicycle is in. \( RR \)'s outputs to \( BB \) include front and rear brakes, shifting, and pedaling. \( BB \) retains \( S \)'s distance and velocity components. These are affected by the inputs from \( RR \). \( RR \) may be defined:

\[ ARR = \{ \text{null, stop-sign} \} \]
\[ ARR_{BBG} = \{ 0, 1 \} \]
\[ ARR_{BBV} = \text{VELOCITY} \]
\[ BRR_{BBP} = \{ \text{null, pedal, stand} \} \]
\[ BRR_{BBL} = \{ \text{null, brake} \} \]
\[ BRR_{BBR} = \{ \text{null, brake, down} \} \]
\[ C_{RR} = \{ 0, 1, 2, 3, 4 \} \]
\[ RR : C_{RR} \times (X_{RR} \times X_{RRBBG} \times X_{RRBBV}) \rightarrow (Y_{RRBBP} \times Y_{RRBBL} \times Y_{RRBBR}) \]

and we follow the general form:

\[ X_{..} = \{ x \mid x : T \rightarrow A_{..} \} \]
\[ Y_{..} = \{ y \mid y : t \rightarrow B_{..} \} \]
XRR is visual input to RR (corresponding to X in S),
XRRBBG is input to RR telling RR what gear the bicycle is in,
and XRRBBV is velocity input from B. YRRBBP is the output of the rider's feet to BB,
YRRBBL is left hand output, and YRRBBBR is right hand output.

For all \( t \) and \( t' \), elements of \( T \), such that \( t' = t + 1 \),
the rules for computing the next state of RR are as follows:

\[
\begin{align*}
crr(t) = 0 & \land xrr(t) \neq \text{stop-sign} \implies crr(t') = 0 \\
crr(t) = 0 & \land xrr(t) = \text{stop-sign} \implies crr(t') = 1 \\
crr(t) = 1 & \land xrrbbv(t) > 28 \implies crr(t') = 1 \\
crr(t) = 1 & \land xrrbbv(t) = 28 \implies crr(t') = 2 \\
crr(t) = 2 & \land xrrbbv(t) > 21 \implies crr(t') = 2 \\
crr(t) = 2 & \land xrrbbv(t) = 21 \implies crr(t') = 3 \\
crr(t) = 3 & \land xrrbbv(t) > 13 \implies crr(t') = 3 \\
crr(t) = 3 & \land xrrbbv(t) = 13 \implies crr(t') = 1 \\
crr(t) = 1 & 9 < xrrbbv(t) < 28 \implies crr(t') = 1 \\
crr(t) = 1 & xrrbbv(t) = 9 \implies crr(t') = 3 \\
crr(t) = 3 & 0 < xrrbbv(t) < 9 \implies crr(t') = 3 \\
crr(t) = 3 & xrrbbv(t) = 0 \implies crr(t') = 4 \\
crr(t) = 4 \implies crr(t') = 4
\end{align*}
\]

In other words, RR starts in state 0 and switches
states in the order 0, 1, 2, 3, 1, 3, 4 as velocity decreases. The output rules are:

\[
\begin{align*}
crr(t) = 0 \implies yrrbbp(t) = \text{pedal} & \land yrrbbl(t) = \text{null} \\
& \land yrrbbr(t) = \text{null} \\
crr(t) = 1 & \land xrrbbv(t) > 9 \implies yrrbbg(t) = \text{null}
\end{align*}
\]
& yrrbbl(t) = null & yrrbbr(t) = null
crr(t) = 2 -> yrrbbp(t) = null & yrrbbl(t) = brake & yrrbbr(t) = null
crr(t) = 3 -> yrrbbf(t) = null & yrrbbl(t) = brake & yrrbbr(t) = brake
crr(t) = 1 & 0 < xrrbv(t) < 9 -> yrbf(t) = pedal & yrrbbl(t) = brake & yrrbbr(t) = down
crr(t) = 4 -> yrrbbp(t) = stand & yrrbbl(t) = null & yrrbbr(t) = null

That is, when in mode 0, the rider pedals, when in 1 the rider coasts, in mode 2 the rear brake is applied, in mode 3 both brakes are applied, in mode 3, the rider pedals and downshifts while continuing rear braking (if velocity is less than 13 fps but greater than 0 fps), and in mode 4 the rider releases both brakes and stands. rr, an element of RR, is presented in Table 3. rr corresponds in terms of input to s C S.

BB may then be defined as follows:

ABBRRP = { null, pedal, stand }
ABBRL = { null, brake }
ABBRRR = { null, brake, down }
BBBRRG = { 1, 2 }
BBBD = DISTANCE
BBBV = VELOCITY
CBB = DISTANCE x VELOCITY x BBBRRG
BB : CBB x (XBBRRP x XBBRL x XXBBRRR) ->
(YBBD x YBBV x YBBRRG)

XBBRRP is the rider's pedal input to BB, XBBRL is left hand input, and XBBRRR is right hand input (both shifting and braking). YBBD is distance output, YBBV is
<table>
<thead>
<tr>
<th>t (crr(t))</th>
<th>xrr(t)</th>
<th>xrrbbg(t)</th>
<th>xrrbbv(t)</th>
<th>yrrbbf(t)</th>
<th>yrrbbl(t)</th>
<th>yrrbbbr(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>null</td>
<td>2</td>
<td>29</td>
<td>pedal</td>
<td>null</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>stop-sign</td>
<td>2</td>
<td></td>
<td>null</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>null</td>
<td>28</td>
<td></td>
<td></td>
<td>brake</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td>27</td>
<td></td>
<td></td>
<td>brake</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>19</td>
<td></td>
<td></td>
<td>brake</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
<td>10</td>
<td></td>
<td>pedal</td>
<td>down</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>9</td>
<td></td>
<td>null</td>
<td>brake</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
velocity output, and YBBRRG is the gear BB is in. Distance, velocity, and gear are also components of BB’s state.

Let P be a set of statements about BB and let CRIT be a function defined:

\[ \text{CRIT} : P \rightarrow \{0,1\} \]

\[ (\forall p \in P) \text{CRIT}(p) = 1 \iff p \]

That is, CRIT(p) takes on a value of 1 only when p is true. The constructive specification for BB is presented below in the form of predicates assumed to be true for BB:

\[ (\forall ((\text{cbbd}, \text{cbbv}, \text{cbbg}), (\text{xbbrrp}, \text{xbbrrl}, \text{xbbrr}), (\text{ybbd}, \text{ybbv}, \text{ybbrrg})) \text{GBB}) \]

\[ (\forall t, t' \in T \rightarrow t' = t + 1) \]

\[ \text{cbbd}(t') = \text{cbbd}(t) - \text{cbbv}(t) \]

\[ \text{cbbv}(t') = \max(0, \text{cbbv}(t) - \text{CRIT}(\text{xbbrrp}(t) = \text{null}) \]

\[ - \text{CRIT}(\text{xbbrrl}(t) = \text{brake}) \]

\[ - \text{CRIT}(\text{xbbrrr}(t) = \text{brake}) \]

\[ \text{cbbg}(t') = 1 \iff \text{cbbg}(t) = 1 \vee \text{xbbrrr}(t) = \text{down} \]

\[ \text{cbbg}(t') = 2 \iff \text{cbbg}(t) = 2 \& \text{xbbrrr}(t) \neq \text{down} \]

\[ \text{ybbd}(t) = \text{cbbd}(t) \]

\[ \text{ybbv}(t) = \text{cbbv}(t) \]

\[ \text{ybbrrg}(t) = \text{cbbg}(t) \]

Current distance is the last distance less the last velocity. Current velocity is last velocity decreased by one foot per second for each of the following conditions: rider not pedaling, rider applying the front brake, or rider applying the rear brake. The gear is changed by a down movement only. Outputs are direct from the state components. BB, a behavior of BB is presented in Table 4. It corresponds precisely to RR, the behavior of RR, in that
<table>
<thead>
<tr>
<th>t</th>
<th>cbbd(t)</th>
<th>cbbv(t)</th>
<th>xbbrrp(t)</th>
<th>xbbrrl(t)</th>
<th>xbbrrr(t)</th>
<th>ybbd(t)</th>
<th>ybbv(t)</th>
<th>ybg(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>298</td>
<td>29</td>
<td>pedal</td>
<td>null</td>
<td>null</td>
<td>298</td>
<td>29</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>269</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>269</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>240</td>
<td>29</td>
<td>null</td>
<td>&quot;</td>
<td>&quot;</td>
<td>240</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>211</td>
<td>28</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>211</td>
<td>28</td>
<td>&quot;</td>
</tr>
<tr>
<td>4</td>
<td>183</td>
<td>27</td>
<td>&quot;</td>
<td>brake</td>
<td>&quot;</td>
<td>183</td>
<td>27</td>
<td>&quot;</td>
</tr>
<tr>
<td>5</td>
<td>156</td>
<td>25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>156</td>
<td>25</td>
<td>&quot;</td>
</tr>
<tr>
<td>6</td>
<td>131</td>
<td>23</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>131</td>
<td>23</td>
<td>&quot;</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>21</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>108</td>
<td>21</td>
<td>&quot;</td>
</tr>
<tr>
<td>8</td>
<td>87</td>
<td>19</td>
<td>&quot;</td>
<td>brake</td>
<td>&quot;</td>
<td>87</td>
<td>19</td>
<td>&quot;</td>
</tr>
<tr>
<td>9</td>
<td>68</td>
<td>16</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>68</td>
<td>16</td>
<td>&quot;</td>
</tr>
<tr>
<td>10</td>
<td>52</td>
<td>13</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>52</td>
<td>13</td>
<td>&quot;</td>
</tr>
<tr>
<td>11</td>
<td>39</td>
<td>10</td>
<td>pedal</td>
<td>&quot;</td>
<td>down</td>
<td>39</td>
<td>10</td>
<td>&quot;</td>
</tr>
<tr>
<td>12</td>
<td>29</td>
<td>9</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>29</td>
<td>9</td>
<td>&quot;</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>8</td>
<td>null</td>
<td>brake</td>
<td>null</td>
<td>20</td>
<td>8</td>
<td>&quot;</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>12</td>
<td>5</td>
<td>&quot;</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>7</td>
<td>2</td>
<td>&quot;</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
<td>0</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
all corresponding inputs and outputs are equal.

The decomposition of S into RR and BB has given us a bit more insight into the problem of stopping a bicycle. If we are willing to live with the assumptions about action sequences and the events that trigger them, specifically sighting a stop sign and changes in velocity, we have a better description of a bicycle and rider, one in which the rider detects events and performs actions based on them. There are still problems, though. We still cannot explain why the particular events are significant to the rider. We do not know precisely how the rider makes decisions, plans ahead, and organizes actions. The new structure that we have constructed is more complex than S and provides more explanatory power than S. In short, it is probably better for organizing knowledge about bicycles and riders in general and our example in particular. However, it is not really good enough and does not satisfy our need for an interpretation of the contents of Chapter 3.

4.4 The Bicycle and Rider as a Human-Machine System

DATA, S, RR, and BB have provided simple mechanistic descriptions of the bicycle and rider. None of them really take into consideration the purposiveness, organizational activities, or the coordination techniques of the bicycle's rider. Since we are interested in how the humans in various systems behave, this is a serious defect.

To remedy this, we must develop another structure to organize what we know about bicycles and riders. We will define the pair as a human-machine system, a system which can be described in terms of goals and procedures. Therefore we have:

\[ BR = (S, \text{GOALS, PROCEDURES}) \]

S has already been defined both as a partly specified system and as a system composed of subsystems RR and BB. Both can be described in terms of goals. The latter can be further decomposed to produce a procedural description. We must now define GOALS and PROCEDURES.
4.4.1 Goals

We will use a very simple set of goals to define BR:

\[
\text{GOALS} = \{ \text{RIDE}, \text{STOP} \}
\]

\[\text{RIDE} = \{ s \mid t', t'' \in T \wedge t' < t'' \]

\[\wedge s = (c, x, (y_d, y_v)) \in S \]

\[\wedge (\forall v \in \text{VELOCITY} \implies v > 0)(\forall t \in T) \]

\[y_v(t) = v \}
\]

\[\text{STOP} = \{ s \mid t', t'' \in T \wedge t' < t'' \]

\[\wedge s = (c, x, y) \in S \]

\[\wedge y_v(t'') = 0 \}
\]

RIDE is a goal in which all behaviors exhibit a constant velocity greater than zero. STOP is a collection of behaviors in which the final velocity is zero. For example, we have:

\[s \in \text{RIDE} \]
Therefore, we can describe S in terms of GOALS.

4.4.2 Procedures and Associated Subsystems

Our intent is to show that we can form a procedural description of S by replacing RR with a collection of procedure processors and a few additional subsystems to connect processors and convert outputs. The procedures to be used are:

\[
\text{PROCEDURES} = \{\text{executive, plan, go, coast, downshift, front-brake, rear-brake}\}
\]

The executive procedure (procedural processor PE) is the central organizing procedure. It senses inputs, determines when an event has occurred and activates and deactivates all other procedures. It uses an additional subsystem, memory (MM) to store procedural plans. That is, MM contains entries which indicate, for a given event, what procedure is to be made active. When the executive determines that an event has occurred, in this case that the rider has detected the stop-sign or has noted a specific change in velocity, MM is queried by the executive to determine if a procedure is to be activated. The executive maintains a list of all active procedures.

The plan procedure (procedure processor PP) adds entries to memory. These entries constitute a sequence of actions to stop the bicycle.
go (PG) is a procedure to pedal the bicycle.

coast (PC) is the complement of go. It causes pedaling to stop and and the rider to wait for the bicycle to stop.

The downshift (PD) procedure causes the bicycle to be downshifted.

front-brake (PF) causes the front brake to be applied, through the rider's right hand.

rear-brake (PR) causes the rear brake to be applied through the rider's left hand.

The bicycle subsystem of S, BB, is used without modification. Demultiplexers DMG, DM, and DMV distribute visual input and input from BB to various procedure processors. Multiplexers MXP, MXR, and MXL convert the outputs of procedure processors into inputs to BB.

The organization of the procedure processors and associated subsystems is diagrammed in Figure 7. Detailed definitions of these structures follow.

Recall that \{inactive, executing, suspended\} is a component set of a procedure processor. The set PROCSTATES = \{ia, ex, sd\} will be used as a substitute set for these to abbreviate. ia corresponds to inactive, ex corresponds to executing, and sd corresponds to suspended.

4.4.2.1 Procedure - executive -

The executive procedure is the central procedure. Its procedure processor controls those of all other procedures.

Recall that a procedure was defined:

\[ \text{procedure} = (\ p\text{name}, p\text{string}, P) \]

where pname is the procedure name, pstring is a string of symbols, and P is a procedure processor, a system in which the symbols contained in pstring form one component of the state object and PROCSTATES constitutes another.
Figure 7
Procedural Representation of S
For *executive*, we then have:

\[
\text{executive} = ( E, \text{pstringe}, \text{PE} )
\]

where \( \text{pstringe} = \text{test-event} \)
- query
- activate
- waitq
- waita
- waitd

\( \text{PE} \) is defined on the sets:

- \( \text{APE} = \{ \text{null,stop-sign} \} \)
- \( \text{APEBBV} = \text{VELOCITY} \)
- \( \text{APEMM} = \{ \text{null} \} \cup \text{PROCNAMES} \)
  - \( \text{PROCNAMES} = \{ \text{E,P,G,C,D,F,R} \} \)
- \( \text{APEPP} = \text{APEPG} = \text{APEPC} = \text{APEPD} = \text{APEPR} = \text{APEPF} = \{ \text{null,done} \} \)
- \( \text{BPEMM} = \text{APE} \cup \text{VELOCITY} \)
- \( \text{BPEPP} = \text{BPEPG} = \text{BPEPC} = \text{BPEPD} = \text{BPEPR} = \text{BPEPF} = \{ \text{enable,disable} \} \)
- \( \text{CPE} = \text{PROCSTATES} \times \text{PSTRINGE} \times \text{BPEMM} \times \text{TP} \times \text{PROCNAMES} \)
- \( \text{TP} = \{ 0,0.25,0.5,0.75,1,...,15.75,16 \} \)

and \( \text{PE} \) itself is defined:

\[
\text{PE} : \text{CPE} \times (\text{XPE} \times \text{XPEBBV} \times \text{XPEMM} \times \text{XPEPP} \times \text{XPEPG} \\
\times \text{XPEPC} \times \text{XPEPD} \times \text{XPEPR} \times \text{XEP}) \rightarrow \\
(\text{YPPEMM} \times \text{YPPEPP} \times \text{YPPEPG} \times \text{YPPEPC} \times \text{YPPEPD} \\
\times \text{YPPEPR} \times \text{YPPEPF})
\]

where the following notational convention holds:

\[
\text{XPE...} = \{ x \mid x : \text{TP} \rightarrow \text{APE...} \}
\]
YPE... = \{ y \mid y : TP \rightarrow BPE... \}

XPE corresponds to RR's visual input (only the time set is different), XPEBBBV is velocity input from BB, the bicycle, and XPEMM is input from MM (memory). XPEPP, XPEPG, XPEPC, XPEPD, XPEPR, and XPEPF are inputs from the other procedure processors. Each processor tells the executive, through these input channels, when it is done.

YPEMM is output to MM, memory. When an event occurs, memory is queried via this output. YPEPP, YPEPG, YPEPC, YPEPD, YPEPR, and YPEPF are outputs to each of the other processors. The executive enables and disables the other procedure processors by these outputs.

The state object of PE consists of four components. The first is the required PROCSTATES component. The second is a component consisting of the symbols in pstringe. The third component keeps track of events. Whenever an input is received (via XPE or XPEBBBV) which is different from the current state, this component is updated. Finally, the fourth component of PE's state is a set consisting of all active procedures. Each active procedure, p, is enabled via its YPEPp output.

The constructive specification for PE will be given in the form of predicates which are held to be true about PE. The formal statements will be interspersed with narrative to explain them.

\[
(\forall ( ((cpep',cpep"),cpee,open),
(xpe,xpebbv,xpemm,xpepp,xpepg,xpepc,xpepd,xpepr,xpepf),
(ypeemm,ypepp,ypepg,ypepc,ypepd,ypepr,ypepf)) \in PE
\]

\[
cpep"(0) = \text{test-event}
\]

\[
open(0) = \{ G \}
\]

PE always starts with the "test-event" symbol in pstringe. Procedure go, whose name is G, is always initially active. In other words, the rider is always pedaling initially.

\[
(\forall t,t' \in TP \mid t' = t + 0.25)
\]
When on the test-event symbol, cpee(t) is updated if any new inputs have been detected. If the current velocity is different from the last, cpee(t') takes on the new velocity. Any updating by a new velocity is always overridden by a stop-sign input, however. When an event is detected, PE goes to the query symbol in pstringe, otherwise, it goes to the waitq symbol. Any procedure processors that have signalled that they are done are removed from cpen(t).

\[ \text{cpee}(t) = \text{test-event} \Rightarrow \]
\[ \text{cpee}(t) \neq \text{xpe}(t) = \text{stop-sign} \Rightarrow \]
\[ \text{cpee}(t') = \text{query} \land \text{cpee}(t') = \text{stop-sign} \]
\[ \land \text{open}(t') = \text{open}(t) - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ \land \text{cpee}(t) \neq \text{xpebbv}(t) \land \text{xpe}(t) \neq \text{stop-sign} \Rightarrow \]
\[ \text{cpee}(t') = \text{query} \land \text{cpee}(t') = \text{xpebbv}(t) \]
\[ \land \text{open}(t') = \text{open}(t) - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ \land \text{cpee}(t) = \text{xpebbv}(t) \land \text{xpe}(t) \neq \text{stop-sign} \Rightarrow \]
\[ \text{cpee}(t') = \text{waitq} \land \text{cpee}(t') = \text{cpee}(t) \]
\[ \land \text{open}(t') = \text{open}(t) - \{ n \mid \text{xpepn}(t) = \text{done} \} \]

PE goes immediately from query to activate.

\[ \text{cpee}(t) = \text{query} \Rightarrow \]
\[ \text{cpee}(t') = \text{activate} \land \text{cpee}(t') = \text{cpee}(t) \]
\[ \land \text{open}(t') = \text{open}(t) - \{ n \mid \text{xpepn}(t) = \text{done} \} \]

\[ \text{cpee}(t) = \text{activate} \Rightarrow \]
\[ \text{cpee}(t') = \text{waitd} \land \text{cpee}(t') = \text{cpee}(t) \]
\[ \land \text{xpemm}(t) = \text{null} \Rightarrow \text{open}(t') = \text{open}(t) \]
\[ - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ \land \text{xpemm}(t) = \text{P} \Rightarrow \text{open}(t') = \text{open}(t) \lor \text{P} \land \text{C} \]
\[ - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ \land \text{xpemm}(t) = \text{D} \Rightarrow \text{open}(t') = \text{open}(t) \lor \text{D} \land \text{G} \]
\[ - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
\[ \land \text{xpemm}(t) \neq \text{null} \land \text{xpemm}(t) \neq \text{P} \]
\[ \land \text{xpemm}(t) \neq \text{D} \Rightarrow \text{open}(t') = \text{open}(t) \lor \text{null} \land \text{xpemm}(t) \]
\[ - \{ n \mid \text{xpepn}(t) = \text{done} \} \]
In the activate state the list of active procedures is updated according to responses received from memory as a result of a query sent in the previous state (query). If a null response is received, no new procedures need activated. If a response of P (plan) is received, P and C (coast) are activated. If a response of D (downshift) is received, D and G (go) are activated. Otherwise, the response is added to the active list. The next state is always waitd. Procedures that have finished are always removed from open(t).

\[
cpep''(t) = \text{waitq} \Rightarrow \\
  cpep(t') = \text{waita} \& cpee(t') = cpee(t) \\
  \& \ \text{open}(t') = \text{open}(t) - \{ n \mid xpepn(t) = \text{done} \}
\]

\[
cpep''(t) = \text{waita} \Rightarrow \\
  cpep''(t') = \text{waitd} \& cpee(t') = cpee(t) \\
  \& \ \text{open}(t') = \text{open}(t) - \{ n \mid xpepn(t) = \text{done} \}
\]

\[
cpep''(t) = \text{waitd} \Rightarrow \\
  cpep''(t) = \text{test-event} \& cpee(t') = cpee(t) \\
  \& \ \text{open}(t') = \text{open}(t) - \{ n \mid xpepn(t) = \text{done} \}
\]

waitq, waita and waitd are dummy states in which nothing happens except the removal of done procedures from the active list. They are used to delay the executive when no event has occurred so that it remains synchronized with BB and the other procedures.

Statements of the output predicates will be provided in a similar manner.

\[
\text{ypemm}(t) = \text{cpee}(t) \iff cpep(t) = \text{query}
\]

\[
\text{ypemm}(t) = \text{null} \iff cpep''(t) \neq \text{query}
\]

Memory is queried with the name of an event when one is detected. Otherwise, output to memory is null.

\[
\text{ypepp}(t) = \text{enable} \iff P \& \text{open}(t)
\]
\[
\text{ypepg}(t) = \text{enable} \iff G \& \text{open}(t)
\]
ypepc(t) = enable <=> C \in\subseteq cpen(t) \& G \notin\subseteq cpen(t)  
ypepd(t) = enable <=> D \notin\subseteq cpen(t)  
ypepr(t) = enable <=> R \in\subseteq cpen(t) \& D \notin\subseteq cpen(t)  
ypepf(t) = enable <=> F \notin\subseteq cpen(t)  

In general, a procedure is enabled if it is active. Two exceptions exist. Since go and coast both require the rider's feet, they cannot execute simultaneously so go is given priority over coast. Similarly, downshift and rear-brake both require the right hand, so downshift is given priority.

4.4.2.2 Associated Subsystem - Memory -

Memory is used to store procedural plans to aid the executive in determining which procedures should be active. Memory (MM) is defined on the following sets:

\[ \text{AMMPE} = \{ \text{null, stop-sign} \} \cup \text{VELOCITY} \]
\[ \text{AMMPPE} = \text{AMMPE} \]
\[ \text{AMMPPP} = \{ \text{null} \} \cup \text{PROCNAMES} \]
\[ \text{BMMPE} = \text{AMMPPP} \]
\[ \text{CMM} = \text{TT}(\text{AMMPE} \times \text{PROCNAMES}) \times (\{ \text{null} \} \cup \text{PROCNAMES}) \]

and MM is defined:

\[ \text{MM} : \text{CMM} \times (\text{XMMPE} \times \text{XMMPPE} \times \text{XMMPPP}) \rightarrow \text{YMMPE} \]

where:

\[ \text{XMM...} = \{ x | x : \text{TP} \rightarrow \text{AMM...} \} \]
\[ \text{YMMPE} = \{ y | y : \text{TP} \rightarrow \text{BMMPE} \} \]
MM stores a procedural plan in its state object. Each entry consists of an event name (a velocity or the value stop-sign) and a procedure name. XMMPE is input from the executive procedure, and can be either a value of null or the name of an event. XMMPPPE and XMMPPP are inputs from the plan procedure processor used for adding new entries to the plan. YMMPE is an output to the executive for responding with procedure names when queried with an event.

The constructive specification is given as for PE, above:

$$(\forall (\text{cmmp,cmmn}),(\text{xmmpe,xmmpppe, xmmppp}),\text{ymmpe}) \in M$$

$$\text{cmmp}(0) = \{(\text{stop-sign}, P)\}$$
$$\text{cmmn}(0) = \text{null}$$

There is always an initial entry in MM's plan: an entry specifying that the plan procedure is to be activated when the stop-sign is detected.

$$(\forall t,t' \in TP \Rightarrow t' = t + 0.25)$$

$$\text{xmmpe}(t) = \text{xmmpppe}(t) = \text{null} \Rightarrow \text{cmmp}(t') = \text{cmmp}(t)$$
$$\text{xmmpe}(t) \neq \text{null} \Rightarrow \text{cmmp}(t') = \text{cmmp}(t)$$

$$ - \{ (e,p)| e = \text{xmmpe}(t) \land (e,p) \in \text{cmmp}(t)\}$$

$$\text{xmmpppe}(t) \neq \text{null} \Rightarrow \text{cmmp}(t') = \text{cmmp}(t) \cup$$

$$\{(\text{xmmpppe}(t), \text{xmmppp}(t))\}$$

$$\text{cmmn}(t') = p \Leftrightarrow \text{xmmpe}(t) = e \land (e,p) \in \text{cmmp}(t)$$

$$\text{cmmn}(t') = \text{null} \Leftrightarrow$$

$$\neg(\exists (e,p) \in \text{cmmp}(t) \Rightarrow \text{xmmpe}(t) = e)$$

$$\text{ymmpe}(t) = \text{cmmn}(t)$$

When the executive queries memory via XMMPE with an event, memory searches its procedural plan to see if there is an entry corresponding to that event. If so, it places the name of the associated procedure in cmmn(t') and removes the entry from the plan. Any procedure name in
cmmn(t) is sent to the executive via YMMPE so that the procedure can be executed.

Entries are added to the plan by the plan procedure. If xmmppe(t) has a value other than null, then the entry (xmmppe(t),xmmppp(t)) is added.
A standard format will be used to describe the remaining procedures and minimal explanation of the various details of each procedure will be given.

Suppose that mproc is a procedure. Then mproc may be defined:

\[
\text{mproc} = (\text{pnamem}, \text{pstringm}, \text{PM})
\]

where pnamem is the name of mproc, pstringm is a string of symbols, and PM is a procedure processor, a system of which the symbols in pstringm form a component. Both pnamem and pstringm will be defined.

PM is defined on a number of sets. APMSS is an input alphabet of symbols PM can receive from a source designated by SS and BPMDD is an alphabet of symbols that PM can send to a destination designated by DD (both SS and DD will be defined for each procedure processor). CPM is a set of states: CPM = PROCSTATES x PSTRINGM. PM may then be defined:

\[
\text{PM} : \text{CPM} \times (\text{XPMPE} \times \text{XPMSS}) \rightarrow (\text{YPMPE} \times \text{YPMDD})
\]

\[
\text{XPMPE} = \{ x \mid x : TP \rightarrow \{\text{enable}, \text{disable}\} \}
\]

\[
\text{XPMSS} = \{ x \mid x : TP \rightarrow \text{APMSS} \}
\]

\[
\text{YPMPE} = \{ y \mid y : TP \rightarrow \{\text{null}, \text{done}\} \}
\]

\[
\text{YPMDD} = \{ y \mid y : TP \rightarrow \text{BPMDD} \}
\]

XPMPE is an input from PE, the executive procedure processor, and can take on the values enable or disable as functions of time. This channel controls the PROCSTATES component of CPM. XPMSS is an input channel from a source such as another procedure processor or an associated subsystem. YPMPE is an output channel to PE. Through this channel, PM tells PE when it is done. YPMDD is an output channel to another procedure processor or an associated subsystem.

A constructive specification consisting of a set of rules for computing next state and output will be given for each procedure.
All procedure processors have identical control structures defined by the first component (PROCSTATES) of the state object and state transitions defined on it. Specifically:

\[
(\forall ((\text{cpm}',\text{cpm}''),(\text{xpmpe},\text{xpmss}),(\text{ypmpe},\text{ypmdd}) \subseteq \text{PM})
(\forall t,t' \in \text{TP} \Rightarrow t' = t + 0.25)
\]

\[
\text{cpm}'(t) = \text{ia} \& \text{xpmpe}(t) = \text{enable} \Rightarrow \text{cpm}'(t') = \text{ex}
\]

\[
\text{cpm}'(t) = \text{ex} \& \text{xpmpe}(t) = \text{disable} \Rightarrow \text{cpm}'(t') = \text{sd}
\]

\[
\text{cpm}'(t) = \text{sd} \& \text{xpmpe}(t) = \text{enable} \Rightarrow \text{cpm}'(t') = \text{ex}
\]

otherwise, \( \text{cpm}'(t') = \text{cpm}'(t) \)

and:

\[
\text{cpm}'(t) \neq \text{ex} \Rightarrow \text{cpm}''(t') = \text{cpm}''(t)
\]

For output we have:

\[
\text{cpm}'(t) \neq \text{ex} \Rightarrow \text{ypmpe}(t) = \text{ypmdd}(t) = \text{null}
\]

\[
\text{cpm}''(t) = \text{stop} \Rightarrow \text{ypmpe}(t) = \text{done}
\]

That is, if the procedure processor is in the inactive state, the only way that it can transition to the executing state is if the executive procedure processor enables it. Once active, the executive can switch it back and forth between executing and suspended. Whenever the processor is not executing, the other component of state cannot change.

The procedure processor can produce output (other than null, which is to be interpreted as no output) only when executing. When the processor enters the stop state (the final state in processing), it outputs a value of done to the executive.

The constructive specification given for each procedure processor will be based on the assumption that the processor is in the executing state. These concepts allow the executive to control the execution of procedures. It can "turn on" and "turn off" the procedure processors, suspending them and resuming them at will.
The above is common to all procedures and will not be explicitly mentioned in the sections that follow. It does hold for each procedure, though.

Following the formal definition of each procedure, a brief explanation will be given.

4.4.2.3 Procedure - plan -

\[ \text{plan} = (P, \text{pstringp}, PP) \]

\[ \text{pstringp} = \text{planr} \]
\[ \text{planf} \]
\[ \text{pland} \]
\[ \text{stop} \]

BPPMME = \{ null, stop-sign \} U VELOCITY

BPPMMP = \{ null \} U PROCNAMES

CPP = PROCSTATES x \{ planr, planf, pland, stop \}

PP : CPP x XPPPE \rightarrow (YPPPE x YPPMME x YPPMMP)

\((\forall ((cpp', cpp''), xpppe), (ypppe, yppmme, yppmmp)) \in PP)\)
\((\forall t, t' \in \mathbb{R} \ni t' = t + 0.25)\)

\[ \text{cpp''}(t) = \text{planr} \Rightarrow \text{cpp''}(t') = \text{planf} \]
\[ \text{cpp''}(t) = \text{planf} \Rightarrow \text{cpp''}(t') = \text{pland} \]
\[ \text{cpp''}(t) = \text{pland} \Rightarrow \text{cpp''}(t') = \text{stop} \]
\[ \text{cpp''}(t) = \text{stop} \Rightarrow \text{cpp''}(t') = \text{planr} \]

\[ \text{cpp''}(t) = \text{planr} \Rightarrow \text{ypppe}(t) = \text{null} \land \text{yppmme}(t) = 28 \land \text{yppmmp}(t) = R \]
\[ \text{cpp''}(t) = \text{planf} \Rightarrow \text{ypppe}(t) = \text{null} \land \text{yppmme}(t) = 21 \land \text{yppmmp}(t) = F \]
\[ \text{cpp''}(t) = \text{pland} \Rightarrow \text{ypppe}(t) = \text{null} \land \text{yppmme}(t) = 13 \land \text{yppmmp}(t) = D \]

plan is a procedure to add entries to the procedural plan in M'. plan works in the following manner. PP is generally in the inactive state. When enabled by PE, it
transitions to the \textit{planr} state and outputs the values 28 and R to MM, which causes an entry of (28,R) to be added to the procedural plan in MM's state. PP then transitions to \textit{planf} and outputs 21 and F to cause an entry of (21,F) to be added to the plan. It then transitions to \textit{pland} and causes an entry of (13,D) to be added to the plan, after which it transitions to stop. In the stop state, it sends a value of done to PE then transitions back to \textit{planr}.

4.4.2.4 Procedure - go -

\begin{align*}
go &= (G, pstringg, PG) \\
pstringg &= \text{pedal} \\
&\quad \text{stop} \\
APG &= \{ \text{null, stop-sign} \} \\
APGBBG &= \{ 1, 2 \} \\
BPGBBP &= \{ \text{null, pedal} \} \\
CPG &= \text{PROCSTATES} \times \{ \text{pedal, stop} \} \\
PG : CPG \times (XPGPE \times XPG \times XPGBBG) &\rightarrow (YPGPE \times YPGBBP) \\
(\forall((cpg', cpg''), (xpge, xpg, xgg), (ypge, ypgbb)) \in CPG) \\
(\forall t, t' \in TP \rightarrow t' = t + 0.25)
\end{align*}

\begin{align*}
cpg''(t) &= \text{pedal} \& \ xpg(t) = \text{stop-sign} \\
&\quad \text{v} \ xpgbbg(t) = 1 \Rightarrow cpg''(t') = \text{stop} \\
cpg''(t) &= \text{pedal} \& \ xpg(t) \neq \text{stop-sign} \\
&\quad \& \ xpgbbg(t) = 2 \Rightarrow cpg''(t') = \text{pedal} \\
cpg''(t) &= \text{stop} \Rightarrow cpg''(t') = \text{pedal} \\
cpg''(t) &= \text{pedal} \Rightarrow ypgpe(t) = \text{null} \& \ ypgbbp(t) = \text{pedal} \\
cpg''(t) &= \text{stop} \Rightarrow ypgpe(t) = \text{done} \& \ ypgbbp(t) = \text{null}
\end{align*}

\textit{go} is a procedure to pedal until the \textit{stop-sign} is detected or until the bicycle is in first gear. \textit{XPG} is external visual input \textit{XPGBBG} is gear input, and \textit{YPGBBP} is
output to the bicycle

4.4.2.5 Procedure - coast -

\[
\text{coast} = (C, \text{pstring}_c, PC)
\]

\[
\text{pstring}_c = \text{coast} \quad \text{stand} \quad \text{stop}
\]

\[
\text{APCBBV} = \text{VELOCITY}
\]

\[
\text{BPCBBP} = \{ \text{null}, \text{pedal.stand} \}
\]

\[
\text{CPC} = \text{PROCSTATES} \times \{ \text{coast}, \text{stand}, \text{stop} \}
\]

\[
\text{PC} : \text{CPC} \times (\text{XPCPE} \times \text{XPCBBV}) \to (\text{YPCPE} \times \text{YPCBBP})
\]

\[
(\forall ((xpc', xpc''), (xpcpe, xpcbbv), (ypcpe, ypcbbp)) \in \text{PC})
\]

\[
(\forall t, t' \in \text{TP} \to t' = t + 0.25)
\]

\[
\begin{align*}
\text{cpc}''(t) &= \text{coast} \& \text{xpcbbv}(t) \neq 0 \Rightarrow \text{cpc}''(t') = \text{coast} \\
\text{cpc}''(t) &= \text{coast} \& \text{xpcbbv}(t) = 0 \Rightarrow \text{cpc}''(t') = \text{stand} \\
\text{cpc}''(t) &= \text{stand} \Rightarrow \text{cpc}''(t') = \text{stop} \\
\text{cpc}''(t) &= \text{stop} \Rightarrow \text{cpc}''(t') = \text{coast} \\
\text{cpc}''(t) &= \text{coast} \Rightarrow \text{ypcpe}(t) = \text{null} \& \text{ypcbbp}(t) = \text{null} \\
\text{cpc}''(t) &= \text{stand} \Rightarrow \text{ypcpe}(t) = \text{null} \& \text{ypcbbp}(t) = \text{stand} \\
\text{cpc}''(t) &= \text{stop} \Rightarrow \text{ypcpe}(t) = \text{done} \& \text{ypcbbp}(t) = \text{null}
\end{align*}
\]

coast is a procedure to coast (no pedaling) until the bicycle stops and then to stand. \text{XPCBBV} is velocity input from the bicycle. \text{YPCBBP} is pedal output to the bicycle.
4.4.2.6 Procedure - downshift -

\[
\text{downshift} = (D, p\text{stringd}, PD)
\]
\[
p\text{stringd} = \text{shift}
\]
\[
\text{stop}
\]
\[
\text{APDDBG} = \{ 1, 2 \}
\]
\[
\text{BPDDBR} = \{ \text{null, down} \}
\]
\[
\text{PD} : \text{CPD} \times (\text{XPDPF} \times \text{XPDDBG}) \rightarrow \text{YPDPF} \times \text{YPDDBR})
\]
\[
(\forall (\text{cpd}', \text{cpd}''), (\text{xpdpf}, \text{xpdbbg}), (\text{ypdpf}, \text{ypdbr})) \in \text{PD}
\]
\[
(\forall t, t', t' \in \text{TP} \rightarrow t' = t + 0.25)
\]
\[
\text{cpd}''(t) = \text{shift} \land \text{xpdbbg}(t) = 2 \Rightarrow \text{cpd}''(t') = \text{shift}
\]
\[
\text{cpd}''(t) = \text{shift} \land \text{xpdbbg}(t) = 1 \Rightarrow \text{cpd}''(t') = \text{stop}
\]
\[
\text{cpd}''(t) = \text{stop} \Rightarrow \text{cpd}''(t') = \text{shift}
\]
\[
\text{cpd}''(t) = \text{shift} \Rightarrow \text{ypdpf}(t) = \text{null} \land \text{ypdbr}(t) = \text{down}
\]
\[
\text{cpd}''(t) = \text{stop} \Rightarrow \text{ypdpf}(t) = \text{done} \land \text{ypdbr}(t) = \text{null}
\]

downshift is a procedure to shift the bicycle to first gear. XPDBBG is input from the bicycle, indicating which gear (1 or 2) it is in. YPDBBR is output to the bicycle's shift mechanism. Downshift, when enabled by PE, moves the shift mechanism down until it is in first gear. It then stops.

4.4.2.7 Procedure - front-brake -

\[
\text{front-brake} = (F, p\text{stringf}, PF)
\]
\[
p\text{stringf} = \text{brake}
\]
\[
\text{stop}
\]
\[
\text{APFBBV} = \text{VELOCITY}
\]
\[
\text{BPFBRR} = \{ \text{null, brake} \}
\]
\[
\text{CPF} = \text{PROCSTATES} \times \{ \text{brake, stop} \}
\]
PF: CPF x (XPFPE x XPFBBV) -> (YPFPE x YPFBBR)

$$\forall ((cpf', cpf''), (xpfpe, xpfbbv), (ypfpe, ypfbbbr)) \in PF$$
$$\forall t, t' \in TP \rightarrow t' = t + 0.25$$

$$cpf''(t) = \text{brake} \& xpfbbv(t) \neq 0 \Rightarrow cpf''(t') = \text{brake}$$

$$cpf''(t) = \text{brake} \& xpfbbv(t) = 0 \Rightarrow cpf''(t') = \text{stop}$$

$$cpf''(t) = \text{stop} \Rightarrow cpf''(t') = \text{brake}$$

$$cpf''(t) = \text{brake} \Rightarrow ypfpe(t) = \text{null} \& ypfbbbr(t) = \text{brake}$$

$$cpf''(t) = \text{stop} \Rightarrow ypfpe(t) = \text{done} \& ypfbbbr(t) = \text{null}$$

The front-brake procedure applies the front brake until the bicycle stops. XPFBBV is velocity input from BB, the bicycle. YPFBBR is front brake output to the bicycle.

4.4.2.8 Procedure - rear-brake -

rear-brake = (R, pstringr, PR)

pstringr = brake
stop

APRBBV = VELOCITY

BPRBBL = { null, brake }

CPR = PROCSTATES x { brake, stop }

PR : CPR x (XPRPE x XPRBBV) -> (YPRPE x YPRBBL)

$$\forall ((cpr', cpr''), (xprpe, xprbbv), (yprpe, yprbbbl)) \in PR$$
$$\forall t, t' \in TP \rightarrow t' = t + 1$$

$$cpr''(t) = \text{brake} \& xprbbv(t) \neq 0 \Rightarrow cpr''(t') = \text{brake}$$

$$cpr''(t) = \text{brake} \& xprbbv(t) = 0 \Rightarrow cpr''(t') = \text{stop}$$

$$cpr''(t) = \text{stop} \Rightarrow cpr''(t') = \text{brake}$$

$$cpr''(t) = \text{brake} \Rightarrow yprpe(t) = \text{null} \& yprbbbl(t) = \text{brake}$$

$$cpr''(t) = \text{stop} \Rightarrow yprpe(t) = \text{done} \& yprbbbl(t) = \text{null}$$

rear-brake applies the rear brake until the bicycle stops. XPRBBV is velocity input from the bicycle, YPRBBL.
is rear brake output to the bicycle.

4.4.2.9 Demultiplexers -

Three demultiplexers are used to provide single input channels to multiple procedure processors and to convert from $T$, the time set of $BB$ to $TP$, the time set of the procedure processors. $DMG$ converts $YBBRRG$, the gear output of the bicycle, $BB$, to $XPCBBG$, the gear input to procedure processor $PC$ (coast). $DM$ converts $XRR$, visual input (originally used for $RR$, the rider subsystem of $S$) to $XPG$ and $XPE$, visual input to $PG$ (go) and $PE$ (the executive). $DMV$ converts $YBBV$, the velocity output of $BB$, to $XPEBBV$, $XPCBBV$, $XPDBBV$, $XPFBBV$, and $XPRBBV$, velocity inputs to $PE$ (the executive), $PC$ (coast), $PD$ (downshift), $PF$ (front-brake), and $PR$ (rear-brake). These demultiplexers are defined:

$$DMG : YBBRRG \rightarrow XPCBBG$$
$$DM : XRR \rightarrow XPE \times XPG$$
$$DMV : YBBV \rightarrow XPEBBV \times XPCBBV \times XPDBBV \times XPRBBV \times XPFBBV$$

where:

$$(\forall (ybbrng,xpcbbg) \in DMG)(\forall (xrr,(xpe,xpg)) \in DM)$$
$$(\forall (ybbv,(xpebbv,xpebbv,xpdbbv,xprbbv,xpfbbv)) \in DMV)$$

$$(\forall t \in T)$$
$$(\forall t' \in TP \supset t'=t \lor t'=t+0.25 \lor t'=t+0.5 \lor t'=t+0.75)$$

$$xpebbg(t') = ybbrrg(t)$$
$$xpe(t') = xpg(t') = xrr(t)$$
$$xpebbv(t') = xpebbv(t') = xpdbbv(t') = xprbbv(t') = xpfbbv(t') = ybbv(t)$$

In other words, these three multiplexers just copy each second's input four times to each output, spreading the value at the beginning of a one second interval over four one-fourth second intervals.
4.4.2.10 Multiplexers -

Three multiplexers are used to convert the output of multiple procedure processors to single input channels to the bicycle, BB, and to convert from TP, the processors' time set, to T, the time set of BB. MXP converts YPGBBP and YPCCBBP, pedal output of PG (go) and PC (coast) to XBBRRP, pedal input to BB. MXR converts YPDBBR and YPFBBR brake/downshift output of PD (downshift) and PF (front-brake) to XBBRRR, corresponding input to BB. MXL converts YPRBBL, brake output of PR (rear-brake) to XBBRRL, input to BB. They are defined:

\[
\begin{align*}
MXP & : YPGBBP \times YPCCBBP \rightarrow XBBRRP \\
MXR & : YPDBBR \times YPFBBR \rightarrow XBBRRR \\
MXL & : YPRBBL \rightarrow XBBRRL \\
\end{align*}
\]

and:

\[
\begin{align*}
(V \ mxp = ((ypgbbp,ypcbbp),xbbrrp) \in MXP) \\
(V \ mxr =((ypdbbr,ypfbbbr),xbbrrr) \in MXR) \\
(V \ mlx = (yprbbl,xbbrrl) \in MXL) \\
(V \ t \in T) \\
\quad xbbrrp(t) = \text{null} \iff ypgbbp(t) = ypcbbp(t) = \text{null} \\
\quad xbbrrp(t) = \text{pedal} \iff ypgbbp(t) \neq \text{null} \& ypcbbp(t) \neq \text{null} \\
\quad xbbrrr(t) = \text{null} \iff ypdbbr(t) = ypfbbbr(t) = \text{null} \\
\quad xbbrrr(t) = \text{down} \iff ypdbbr(t) = \text{down} \\
\text{otherwise, } xbbrrr(t) = \text{brake}
\end{align*}
\]
In other words, PG (go) is given precedence over PC (coast) in output to BB and PD (downshift) is given precedence over PF (front-brake).

4.4.3 A Procedural Representation of S

Now we want to show that by combining the procedure processors and associated subsystems defined above, we can develop a procedural representation, call it P, of the bicycle and rider system, S. Specifically, we want to compute a behavior, p, of P, that is identical, in terms of input and output to s, the behavior of S we have been talking about. To do this we must perform the following:

1. Define P.
2. Identify the component behaviors of the procedure processors and associated subsystems to be computed.
3. Identify the interconnections between the procedure processors and associated subsystems.
4. Identify initial conditions for all procedure processors and associated subsystems.
5. Compute the behaviors.
6. Compare the resulting behavior, p, with s.

4.4.3.1 Definition of P -

P is formed by connecting BB, the bicycle subsystem of S, with the procedure processors, demultiplexers, multiplexers, and MM, the memory subsystem, defined above. That is, P is comprised of the following components:

The executive procedure processor:
PE : CPE x (XPE x XPEBBV x XPEMM x XPEPP x XPEPG x
XPEPC x XPEPD x XPEPR x XPEPF) ->
(YPEMM x YPEPP x YPEPG x YPEPC x
YPEPD x YPEPR x YPEPF)

The memory subsystem:

MM : CMM x (XMMPE x XMMPP x XMMPPP) -> YMMPE

The plan procedure processor:

PP : CPP x XPPPE -> (YPPE x YPPMME x YPPMMP)

The go procedure processor:

PG : CPG x (XPBPE x XPG x XPGBBG) -> (YPGPE x YPGBBP)

The coast procedure processor:

PC C : CPC x (XPCPE x XPCBBV) -> (YPCPE x YPCBBP)

The downshift procedure processor:

PD : CPD x (XPDPE x XPDBBG) -> (YPDPE x YPDBBR)

The front-brake procedure processor:

PF : CPF x (XPFPE x XPBBV) -> (YPFPE x YPFBBR)

The rear-brake procedure:

PR : CRR x (XPRPE x XPRBBV) -> YPRPE x YPRBBRL

The demultiplexers:

DMG : YBBRRG -> XPCBBG
DM : XRR -> XPE x XPG
DMV : YBBV -> XPEBBV x XPCBBV x XPDBBV x XPRBBV
x XPBBBV

The multiplexers:

MXP : YPGBBP x YPCBBP -> XBBRRP
MXR : YPDBBR x YPBBR -> XBBRRR
MXL : YPRBBBL -> XBBRRL
The bicycle subsystem:

\[ \text{BB} : \text{CBB} \times (\text{XBBRRP} \times \text{XBBRL} \times \text{XBBRR}) \rightarrow (\text{YBBD} \times \text{YBBV} \times \text{YBBRRG}) \]

P may then be defined:

\[ \text{P} : \text{CP} \times \text{XP} \rightarrow (\text{YPD} \times \text{YPV}) \]

where:

\[ \text{CP} = \text{CPE} \times \text{CPP} \times \text{CPG} \times \text{CPC} \times \text{CPD} \times \text{CPF} \times \text{CPR} \times \text{CMM} \times \text{CBB} \]

\[ \text{XP} = \text{XRR} \]

\[ \text{YPD} = \text{YBBD} \]

\[ \text{YPV} = \text{YBBV} \]

4.4.3.2 Identification of Behaviors -

Next we need to identify the behaviors to be computed. This simply means giving arbitrary names to the behaviors that will be computed so that we can talk about them later. The behaviors are:

\[ \text{pe} = ((\text{cpep, cpee, cpem}), (\text{xpe, xpebbv, xpemm, xep, xpepg, xpepc, xpepd, xpepr, xpepf}), (\text{ypemm, ypepp, ypepg, ypepc, ypepd, ypepr, ypepf})) \subseteq \text{PE} \]

\[ \text{mm} = ((\text{cmmp, commn}), (\text{xmmpe, xmmppe, xmmppp}), (\text{yymmpe})) \subseteq \text{M} \]

\[ \text{pp} = (\text{cpp, xpppe}, (\text{yppmme, yppmme, yppmmp})) \subseteq \text{PP} \]

\[ \text{pg} = (\text{cpg, (xpgpe, xpg)}, (\text{ypgpe, ypgbbp})) \subseteq \text{PG} \]

\[ \text{pc} = (\text{cpc, (xpcpe, xpcbbv)}, (\text{ypcpe, ypcbbp})) \subseteq \text{PC} \]

\[ \text{pd} = (\text{cpd, (xpdpe, xpdbbg)}, (\text{ypdpe, ypdbbr})) \subseteq \text{PD} \]
pf = (cpf,(xpfpe,xpfbbv),(ypfpe,ypfbbp)) ∈ PF
pr = (cpr,(xprpe,xprbbv),(yprpe,yprbbl)) ∈ PR
dmg = (ybbrrg,xpobbg) ∈ DMG
dm = (xrr,(xpe,xpg)) ∈ DM
dmv = (ybbv,(xpebbv,xpobbv,xpdbbv,xprbbv,xpfbbv)) ∈ DMV
mxp = ((ypgbbp,ypcbbp),xbbrrp) ∈ MXP
mxr = ((ypdbbr,ypfbrbr),xbbrrr) ∈ MXR
mxl = (yprbbl,xbbrrl) ∈ MXL
bb = ((obbd,obbv,obbg),(xbbrrp,xbbrrr,xbbrrl),
(ybbd,ybbv,ybbrrg)) ∈ BB

4.4.3.3 Identification of Interconnections -

The description of S through the use of procedure processors depends not on the behaviors of the individual processors but on the behavior, p, of P, the procedural representation of S. It is therefore necessary to indicate, based on the behaviors identified in the previous section, which outputs are connected to which inputs and vice versa. These connections are as follows:

pe:

ypemm = xmmpe PE → M PE query channel to memory
ypepg = xpppe PE → PP Executive control of plan
ypepe = xpgpe PE → PG Executive control of go
ypepc = xpcpe PE → PC Executive control of coast
ypepd = ypdpe PE → PD Executive control of downshift
ypepr = xprpe PE → PR Executive control of rear-brake
ypepf = xpfpe PE → PF Executive control of front-brake

mm:

ymmpe = xppm M → PE Memory response to executive
pp:
ypppe = xpepp PP -> PE plan done channel to executive
yppmme=xmmpppe PP -> MM plan event name channel to memory
ypppmp=xmmppp PP -> MM plan procedure channel to memory

pg:
ypgpe = xpepg PG -> PE go done channel to executive

pc:
ypcpe = xpepe PC -> PE coast done channel to executive

pd:
ypdpe = xpepd PD -> PE downshift done channel to executive

pf:
ypfpee = xpepf PF -> PE front-brake done channel to exec

pr:
yprpee = xprpe PR -> PE rear-brake done channel to exec

The reader is urged to refer again to Figure 7 to better understand the various procedure processor and associated subsystem interconnections.

4.4.3.4 Identification of Initial Conditions -

In order to compute behaviors for the procedure processors we must identify initial conditions. These values are derived from converting rr's inputs, above, and through the restrictions placed on the processors in their definitions. The initial conditions, along with their interpretations are as follows:

pe: executive
cpep(0) = ex,test-event executive on first symbol
cpee(0) = 29 Last input received was 29 fps
xpe(0) = null  
Stop sign not seen yet
xpebbv(0) = 29  
Initial velocity = 29 fps
xpemm(0) = null  
No response from memory
xpepp(0) = null  
plan not done
xpepg(0) = null  
go not done
xpepc(0) = null  
coast not done
xpdpe(0) = null  
downshift not done
xpepr(0) = null  
rear-brake not done
xpepf(0) = null  
front-brake not done

ypemm(0) = null  
No query to memory
yppepp(0) = disable  
plan inactive
yppepg(0) = enable  
go active
yppepc(0) = disable  
coast inactive
tpepd(0) = disable  
downshift inactive
yppepr(0) = disable  
rear-brake inactive
yppepf(0) = disable  
front-brake inactive

mm: memory

cmmmp(0) = {(stop-sign,P)}  
Prepared to start plan procedure
cmmn(0) = null  
No response to executive

xmmpe(0) = null  
No query from executive
xmmpp(0) = null  
No entry from plan to deposit
xmmmp(0) = null  
No entry from plan to deposit

ymmpe(0) = null  
No response to executive

pp: plan

cpp(0) = ia,planr  
plan inactive
xpppe(0) = disable  
Disabled by executive

yppe(0) = null  
plan not done
ypmmpe(0) = null  
No entry to deposit in memory
ypmmmp(0) = null  
No entry to deposit in memory

pg: go

cpg(0) = ex,pedal  
go initially pedaling
xpgpe(0) = enable  
Enabled by executive
xpg(0) = null  
Stop sign not yet seen

ypgpe(0) = null  
go not done
ypgbbp(0) = pedal  
go initially pedaling
**pc: coast**

- `cpc(0) = ia,coast`
  - coast initially inactive
- `xpcpe(0) = disable`
  - Disabled by executive
- `xpcbbv(0) = 29`
  - Velocity is 29 fps
- `ypcpe(0) = null`
  - coast not done
- `ypcbbp(0) = null`
  - No output

**pd: downshift**

- `cpd(0) = ia,shift`
  - downshift initially inactive
- `xpdpe(0) = disable`
  - Disabled by executive
- `xpdbbg(0) = 2`
  - Bicycle initially in 2nd gear
- `ypdpe(0) = null`
  - downshift not done
- `ypdbbr(0) = null`
  - No output

**pf: front-brake**

- `cpf(0) = ia,brake`
  - front-brake initially inactive
- `xpfpe(0) = disable`
  - Disabled by executive
- `xpfbbv(0) = 29`
  - Velocity is 29 fps
- `ypfpe(0) = null`
  - front-brake not done
- `ypfbbr(0) = null`
  - No output

**pr: rear-brake**

- `cpr(0) = ia,brake`
  - rear-brake is currently inactive
- `xprpe(0) = disable`
  - Disabled by executive
- `xprbbv(0) = 29`
  - Velocity is 29 fps
- `yprpe(0) = null`
  - rear-brake not done
- `yprbbl(0) = null`
  - No output

**bb: bicycle**

- `cbbd(0) = 298`
  - 298 feet from stop sign
- `cbbv(0) = 29`
  - 29 feet per second
- `cbbg(0) = 2`
  - in high gear
- `xbbrrp(0) = pedal`
  - rider is pedaling
- `xbbrrl(0) = null`
  - rear brake not applied
- `xbbrrr(0) = null`
  - front brake not applied
ybbd(0) = 298
ybbv(0) = 29
ybbrrg(0) = 2

4.4.3.5 Computation of Behaviors -

The individual behaviors may now be computed with the information provided previously, using the following algorithm:

1. \( t = 0.25 \)
2. Compute PE state at time \( t \)
3. Compute PE output at time \( t \)
4. Compute all other processor and subsystem states at time \( t \)
5. Compute all other processor and subsystem outputs at time \( t \)
6. Compute PE input at time \( t \)
7. \( t = t + 0.25 \)
8. If \( t < 16 \) go to step 2, else continue
9. Stop

Tables 4-13 show the results of the use of this algorithm. Table 4 gives the behavior, bb, of BB. Tables 5-12 give a detailed listings for time 0-4 seconds of pe, the executive procedure processor (Tables 5-7), mm, the memory subsystem (Table 8), pp, the plan procedure processor (Table 9), pg, the go procedure processor (Table 10), pc, the coast procedure processor (Table 11), and mxp, one of the multiplexers (Table 12). Table 13 gives a second-by-second account of the overall behavior of p, concentrating on input and output. State may be determined from states of the individual components. Figure 8 is a procedure profile of s as represented by p. In Figure 8, time is represented by the horizontal axis.
### TABLE 5

pe & PE - State

<table>
<thead>
<tr>
<th>$t$</th>
<th>cep($t$)</th>
<th>cpee($t$)</th>
<th>open($t$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ex,test-event</td>
<td>29</td>
<td>{G}</td>
</tr>
<tr>
<td>0.25</td>
<td>ex,waitq</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>ex,waita</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>ex,waitd</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>ex,test-event</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>ex,query</td>
<td>stop-sign</td>
<td>Ø</td>
</tr>
<tr>
<td>1.5</td>
<td>ex,activate</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.75</td>
<td>ex,waitd</td>
<td>&quot;</td>
<td>{P,C}</td>
</tr>
<tr>
<td>2</td>
<td>ex,test-event</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>ex,query</td>
<td>29</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>ex,activate</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.75</td>
<td>ex,waitd</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>ex,test-event</td>
<td>&quot;</td>
<td>{C}</td>
</tr>
<tr>
<td>3.25</td>
<td>ex,query</td>
<td>28</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.5</td>
<td>ex,activate</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.75</td>
<td>ex,waitd</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>4</td>
<td>ex,test-event</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>$t$</td>
<td>$xpe(t)$</td>
<td>$xpebbv(t)$</td>
<td>$xpmem(t)$</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>null</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>stop-sign</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>P</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>null</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;</td>
<td>28</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>t</td>
<td>ypepp(t)</td>
<td>ypepc(t)</td>
<td>ypepd(t)</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>0</td>
<td>null</td>
<td>disable</td>
<td>enable</td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>stop-sign</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.5</td>
<td>null</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>enable</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>disable</td>
</tr>
<tr>
<td>2.25</td>
<td>29</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>null</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;</td>
<td>disable</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.25</td>
<td>28</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.5</td>
<td>null</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>3.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>t</td>
<td>cmmpe(t)</td>
<td>cmmpn(t)</td>
<td>xmmpe(t)</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>0</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>&quot;</td>
<td>stop-sign</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.5</td>
<td>Ø</td>
<td>P</td>
<td>null</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>null</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>28</td>
</tr>
<tr>
<td>2.25</td>
<td>{(28,R)}</td>
<td>&quot;</td>
<td>29</td>
</tr>
<tr>
<td>2.5</td>
<td>{(28,R),(21,F)}</td>
<td>&quot;</td>
<td>null</td>
</tr>
<tr>
<td>2.75</td>
<td>{(28,R),(21,F),(13,D)}</td>
<td>&quot;</td>
<td>null</td>
</tr>
<tr>
<td>3</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>t</td>
<td>cpp(t)</td>
<td>xpppe(t)</td>
<td>ypppe(t)</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>1.5</td>
<td>ia,planr</td>
<td>disable</td>
<td>null</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>enable</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>ex,planr</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>ex,planf</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.75</td>
<td>ex,stop</td>
<td>&quot;</td>
<td>done</td>
</tr>
<tr>
<td>3</td>
<td>ia,planr</td>
<td>disable</td>
<td>null</td>
</tr>
<tr>
<td>t</td>
<td>cpg(t)</td>
<td>xpgpe(t)</td>
<td>xpg(t)</td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>ex, pedal</td>
<td>enable</td>
<td>null</td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>ex, stop</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.5</td>
<td>ia, pedal</td>
<td>disable</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>t</td>
<td>cpc(t)</td>
<td>xpcpe(t)</td>
<td>xpcbbv(t)</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>0</td>
<td>ia,coast</td>
<td>disable</td>
<td>29</td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>ex,coast</td>
<td>enable</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
### Table 12

mxp & MXP

<table>
<thead>
<tr>
<th>t</th>
<th>ypgbbp(t)</th>
<th>ypebbp(t)</th>
<th>yrrbbp(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>pedal</td>
<td>null</td>
<td>pedal</td>
</tr>
<tr>
<td>0.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>0.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.25</td>
<td>null</td>
<td>&quot;</td>
<td>null</td>
</tr>
<tr>
<td>1.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>1.75</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.25</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2.5</td>
<td>&quot;</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
### TABLE 13

<table>
<thead>
<tr>
<th>t</th>
<th>xp(t)</th>
<th>ypd(t)</th>
<th>ypv(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>null</td>
<td>298</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>stop-sign</td>
<td>269</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>null</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>211</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>183</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>156</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>131</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>108</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>87</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>68</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>52</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>39</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>29</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Event-causing inputs

Procedures
rear-brake
front-brake
down-shift
coast
go
plan
executive

Time (seconds)

Figure 8
Procedure Profile of S
Procedures are depicted above the time axis, also horizontally, and their states as functions of time are represented as follows. No line means the corresponding procedure is inactive. A solid line means the procedure is executing. A dashed line means the corresponding procedure is active yet suspended. Specific inputs which cause events are represented by vertical dashed lines which are labeled at the top with the event name. The tables and figure are explained chronologically in the following paragraphs.

Time 0 - The initial velocity is 29 fps, procedure go is executing, and an output value of pedal is produced by MXP.

Time 0.25 - No event was detected at time 0 so the executive transitions to the waitq state. No other changes have occurred.

Time 0.5 - The executive transitions to the waita state.

Time 0.75 - The executive transitions to the waitd state.

Time 1 - An input value of stop-sign is received (the stop sign is seen).

Time 1.25 - The executive goes to the query state and queries memory to see if a procedure needs to be activated when the stop sign is detected. go, also having detected the stop sign, transitions to the stop state and sends a value of done to the executive.

Time 1.5 - Since (stop-sign,P) was an entry in memory's state object, memory responds to the executive's stop-sign query with P. P and C are added to the executive's active procedure list in its state object. go, having signalled that it is done, is removed from the list and transitions to the inactive state.

Time 1.75 - The executive transitions to the waitd state. At this time, plan and coast are the only procedures executing besides the executive.

Time 2 - The executive transitions to the waitd state.

Time 2.25 - plan deposits an entry of (28,R) in memory to cause rear-brake to be activated when velocity drops to 28 fps.
Time 2.5-3 - plan deposits (21,F) and (13,D) in memory then stops. Velocity drops to 28 fps at time 3.

Time 4-6 (Table 13, Figure 8) - rear-brake begins executing.

Time 7 - Velocity drops to 21.

Time 8-9 - This change causes front-brake to begin executing.

Time 10 - Velocity drops to 13 fps.

Time 11 - 12 - This causes downshift and go to execute and front-brake and coast to be suspended.

Time 13 - downshift and go being finished, front-brake and coast are resumed and continue executing.

Time 14-16 - Velocity drops to 0. The bicycle stops 5 feet from the stop sign.

4.4.3.6 Comparison of Behaviors p and s -

Now all that remains is the comparison of the behavior of p with that of s. By comparing Table 13 with Table 2 we see that:

\[(\Psi t \in T)\]
\[
x_p(t) = x(t) \quad \text{(visual input)}
\]
\[
y_{pd}(t) = y_d(t) \quad \text{(distance)}
\]
\[
y_{pv}(t) = y_v(t) \quad \text{(velocity)}
\]

Therefore, the procedure processors and associated subsystems form an equivalent description of S in the case of s and therefore P is a procedural representation of S and p is a procedural representation of s.
4.4.4 Discussion

In describing the bicycle and rider as a human-machine system we have attempted to form an interpretation of the terms and axioms of Chapter 3. We have clearly used the terminology. The interpretation of the axioms has also been accomplished because of the following.

BR, the bicycle/rider human-machine system, uses procedural plans. These plans are entries in MM's state object. For that reason, MM may be considered a knowledge base.

Not only are procedural plans stored, they are used. Procedures are made active according to the entries stored in MM's state object.

Procedure competition exists in the description since multiple procedure processors produce output which is intended for a single output channel. This competition is mediated to a certain extent by the multiplexers MXP and MXR.

Not all procedures can execute simultaneously. This is built into PE's output functions.

The executive procedure controls the activation, execution, and suspension of all procedures.

Based on the foregoing, a number of points must be raised. First, it must be emphasized that BR is just a structure to organize knowledge about stopping a bicycle under very constrained conditions. We are not saying that all (maybe even any) bicycle riders actually execute these procedures, but the structure presented provides an interesting way to look at the organization of activities to get a bicycle stopped.

Some more specific points need considered as well.

In this description, the executive was given full control of all other procedures. This might not always be the case. It might be more useful, especially in more complex, realistic models, to divide the executive's responsibilities among other procedures. For example, much external input (visual and velocity) was handled by the executive. It might have been more interesting to have included a procedure responsible for all input and output, which provided the interconnection with the world and the
bicycle.

In this example, memory was capable of handling only one procedure per event - the activation of multiple procedures after the occurrence of an event was taken care of through the executive's state transition function. A more general case would allow memory to respond to a query with any number of procedures to activate.

In the cases where multiple procedures produced output for a single external output channel (downshift/front-brake and go/coast), a simple priority scheme to arbitrate was built into the executive and into the output arbitrators. A more general, dynamic priority scheme could readily be conceived.

Finally, all procedure processors in this example used the same time set. There is no technical reason why this is necessary. Different procedures could use different time sets assuming that some method for determining the appropriate relationships could be defined.

4.5 Structures for Organizing Knowledge About Bicycles and Riders

Chapter 4 has been a stepwise regression from a high level of abstraction, in which we described a bicycle and rider in terms of some numbers, to a low level of abstraction, in which we tried to explain these numbers in terms of things like goals and procedures. What has resulted is a collection of structures for organizing knowledge about bicycles and riders stopping at stop signs. Each subsequent structure has provided more explanatory power than the last, but at the expense of additional assumptions.

Assumptions are necessary to pose structures, but these assumptions themselves pose a significant danger. We must be aware of them and be sure they are valid in the sense that the resulting structures tell us what we need to know in a manner that is consistent with what we already are sure of.

The final test of any structure used to organize knowledge about any subject domain is in whether or not the structure adds insight to the problems in the domain and whether or not it raises significant and interesting
issues. Chapter 5 will explore these questions.

4.6 Summary

In Chapter 4 an interpretation of the terms and axioms of Chapter 3 was presented. The case of a rider stopping a bicycle at a stop sign was explored from several levels of abstraction. First the situation was described informally and a method of applying a qualitative structure to it was suggested. Next, a more structured approach was used and the pair was described as a simple system and as a system composed of bicycle and rider subsystems. The remainder of the chapter was devoted to describing the bicycle and rider as a human-machine system. Goals and procedures were defined and the behavior of the bicycle and rider were explained in those terms.
CHAPTER 5

Some Implications of the Structure

The terms and axioms presented in Chapter 3 and the interpretation discussed in Chapter 4 provide a rather different way to describe human-machine systems. They immediately raise a number of issues that might not have arisen otherwise and in other cases provide a different perspective on problems that are already of some concern. This fact is at least in part justification for the research that produced the structure for that was really one of its major objectives, to raise issues and to present them in such a way that they can be addressed in a systematic manner.

The purpose of this chapter is to identify some of the issues and implications raised by the structure, to show that they are at least potentially addressable and to discuss some opportunities for research provided by the issues. In addition, we will attempt to show that the formal approach used in the development provides additional insight into certain issues, insight that may not be provided by less formal approaches.

5.1 Issues Raised by the Structure

Of all the general issues raised by the structure, possibly the most fundamental is one of existence. A number of entities were proposed in Chapter 3, such as goals, procedures, an executive procedure, and so forth, that may be interpreted as mere speculation. Consequently, the existence of these entities in real human-machine systems or the validation of the concepts should be of some concern, especially to psychologists and to others interested in explanation as opposed to just performance prediction.
One way to address this problem is through the psychological literature. There seems to be some indication in the literature that things like goals or executive procedures do exist, but it may not be the case that they exist in the form postulated here. It may be necessary to re-conceptualize a few things to produce a sufficient degree of harmony.

Another way to go about validating existence is through experimentation. The search for knowledge about how human operators organize their activities, while already well underway, might benefit from the concepts put forth here.

Whether or not a structure such as a procedure or a knowledge base "exists" or can be validated may be less important than whether or not the concept is useful. For example, whether or not human operators in manual control tasks actually have Kalman filters in their heads is really of less importance than the fact that they do perform something corresponding to state estimation and that optimal control theory has in many cases been able to make some rather interesting statements about human performance. Similarly, whether or not people actually execute procedures in all of their activities is less important than the fact that we can describe such consistent and repeatable behavior as being directed by procedures.

Artifacts of descriptive structures, things that don't "exist" but which form important parts of these structures, can be justified on a number of grounds. If they allow us to make fairly accurate predictions about how a system will perform, then they are useful to the designer and the planner. If they provide a degree of explanation for how a system operates, then they can be useful to the scientist. On the other hand, if they merely provide a quality of consistency, if they "fill in the gaps," so to speak, they are of some justification to the mathematician or the aestheteic.

Given that an entity like a goal or a procedure exists or that it is useful to describe a human-machine system as if that entity existed, its representation raises another issue in two areas. First, we must be able to refer to it in order to talk about it, and second we may wish to talk about how it is represented and stored in a human-machine system.

In either case, we must consider the representation of these entities in terms of symbology, syntax, and
semantics. Symbology refers to the actual symbols that we use to describe or talk about something and it is determined in part by the media of storage and communication. In the simple case of Ph.D. dissertations, the medium is (at least at the time of writing) printed text, so the alphabet of a natural language, supplemented possibly by some technical, musical, or mathematical notation, comprises the symbology. When such material is to be stored by machines like computers and word processors, another symbology, isomorphic to the first, is used, but this symbology, consisting of the states of semiconductor devices and the magnetic characteristics of metallic oxide surfaces, is not directly interpretable by the humans for whom it is intended. A translation process intervening between the devices and the humans must be present.

The syntax of representation relates to how the symbols may be arranged, on paper, in computer memory, or in our own minds. The particular syntax used may be the result of design, accident, or tradition, but without it, the information that is to be conveyed will be unintelligible, for without syntax there is, by definition, no agreement about how that material is to be physically or temporally arranged.

The semantics of representation are the rules about how groups of symbols relate to the things that they represent. Here again there must be agreement about semantics or communication is impossible.

So whether we are talking about what a procedure is or about how human-machine systems use procedures, representation of that procedure is an important issue. While just talking about a procedure seems quite easy, the manner in which it is stored in some human-machine system's knowledge base is probably a mystery. There seems ample opportunity for research (and it is certain that much has already been undertaken) into how information is stored in human-machine systems.

5.1.1 Some Issues Raised by the Terminology

The definition of systems put forth in the previous chapters are not really all that different from conventional definitions of systems as input/output entities. Consequently, the approach used here does not of
itself raise a lot of new issues concerning systems. However we can conceive of a number of implications that result from such a perspective.

For example, a taxonomy of systems, based on input/output characteristics might be of some use, especially to human-machine system designers. Van Cott and Kinkade (1972), among others, have already started such an undertaking, but further work could be attempted with respect to humans. For example, it might be possible and useful to write human operator job descriptions in terms of the input and output characteristics of the activities required of the operator.

More fundamental, though, is the question of whether or not systems really exist. The term is used a lot in the sciences and engineering, but it seems that systems really only exist in our thinking. They are just conceptual structures that we use to organize knowledge, but they are valuable to us just for that reason.

The concept of goals raises a number of interesting issues. First, we again must wonder about their existence. Of course the idea of purposiveness of HMS's is of little controversy in the field of psychology, but the exact nature of a goal or an objective is open to dispute and the definition of a goal as given in Chapter 4 is by no means the only one that might be postulated. However, the concept of a goal as a collection of acceptable behaviors seems general enough to account for many other definitions, but whether or not human-machine systems actually implement goals in this form needs to be validated.

Assuming that they do, the representation and storage of goals becomes of major interest. If a human-machine system is to develop some plan of action (a procedural plan) based on a goal, there must be some way of representing that goal to the HMS and of storing that representation at least while the plan is developed. Of course a set of mission objectives, written, for example on a briefing sheet, is one implementation, but it is by no means a universal one. It seems certain that human operators are capable of personally encoding information about goals and storing them in their memories. This process should be of some interest, for the accurate and efficient storage of such information seems crucial to the satisfaction of HMS objectives. Here again there is ample opportunity to continue research in an area already of concern, that of human memory, but with a particular effort in mind, based on the structure put forth here.
If goals do exist, they must come from somewhere, although that somewhere was not stated in Chapter 3. If we speak of their creation as goal formulation, there is opportunity to discover the nature of the process of formulating goals. Decomposition of high level goals into simpler subgoals seems one possible avenue, but that does not answer the question of where the original goal came from. (Was it the result of the decomposition of still another? And what about that one?) This may be an inquiry into some of the fundamentals of human behavior, but it certainly has practical implications in the design of human-machine systems and the planning of their activities. We may also ask about how a goal becomes known to an HMS. Of course some goals are imposed on the system from outside, such as mission objectives and so on, but others come from within the system as well, such as subgoals that result from the decomposition of goals specifying mission objectives. It seems likely that system communication and knowledge bases play an important role here, but this issue is much too complex to dismiss with just a few paragraphs.

Often goals are inherently contradictory, as when a pilot must look outside to get a visual fix on an approaching runway while simultaneously monitoring engine instruments. One way to explain the arbitration that must take place is in terms of priority. It is true (at least in private aviation) that an engine that got the airplane to final approach will most likely function long enough and well enough to get the airplane onto the runway. Therefore, knowledge of engine condition is of less importance than where the runway is while landing. We say that during final approach, visual acquisition of the runway has a higher priority than does knowledge of engine conditions.

This is again a rather simple example, at least under most normal circumstances, but the general idea of priority and the assigning of priorities to conflicting goals should be of major concern. There is much that could be done to investigate and to understand the role of priority in goal formulation and achievement, and how those priorities change with time and with changing circumstances.

The concept of a procedure presented in this work, although having some unique characteristics, seems rather similar to a number of other conceptual devices developed in the fields of psychology and artificial intelligence. For example, Miller et al (1960) used the concept of TOTE (Test-Operate-Test-Exit) units as a response to the then-popular view that human behavior could be described
largely in terms of simple reflex actions. A TOTE unit is a semi-autonomous action unit that processes inputs to produce outputs, but feedback plays an important role in the action of the unit. Hierarchies of TOTE units were used to describe complex human activities.

In the field of artificial intelligence, the concept of productions is an important one. A production, in the context of a computer program or the description of human behavior in terms of computer programs, is a simple action rule that is invoked when a certain condition is detected, much as procedures are invoked by events in the structure presented and illustrated in Chapters 3 and 4.

Newell and Simon (1972) developed the idea of operators to assist them in their description of human problem solving. An operator is a conceptual device used to reduce the difference between a current state and a desired state. Like productions and procedures, operators are invoked to handle certain circumstances.

A production is usually a very simple action rule. To explain (in psychology) and to produce (in the field of artificial intelligence) more complex behaviors, the concept of demons has been advanced (Winston, 1977; Boden, 1977). A demon can be described as a sort of sentinel that watches for a particular condition. When that condition is detected, the demon performs some action which may be quite complex and may involve the invoking of other demons.

This is very similar to the concept of an expert. Chandrasekaran et al (1979) developed a computer program to aid in the diagnosis of liver disease. They used modules called experts to deal with particular subproblems of the diagnosis task. When an expert detected a situation with which it could not deal, it passed the problem along to an expert with more specific knowledge about that particular situation.

Although these concepts are somewhat similar to that of a procedure in that they utilize specialized modules to deal with particular situations, one major difference exists, that of the control structure. Each approach requires some sort of control apparatus to invoke the modules, but none employ a mechanism quite like the executive procedure. Nevertheless, it would be interesting to further investigate the similarities, possibly through a formal, set-theoretic analysis of defining objects.
The existence of something like procedures, as defined in Chapter 3, seems almost indisputable, for it is clear that linguistic structures are used to guide system activities, as when aircraft crews use checklists and other textual aids to assist them in their duties. We can, however, argue about the exact nature of procedures and whether or not procedures defined as before do in fact exist.

Probably more controversial, though, is the issue of the universality of procedures. In other words, it is one thing to say that an aircraft crew using a checklist prior to final approach is executing a procedure, it is another thing entirely to say that a kid balancing on a unicycle is executing a procedure. There are at least two ways to address this issue. One is to hypothesize that the kid really is subconsciously executing a procedure that is stored in his/her brain using some symbology, syntax, and semantics that we don't yet understand. This claim demands some substantiation by way of discovering that symbology, syntax and semantics. Another way to deal with this problem is simply to say that we can describe the kid as behaving AS IF he/she were executing a procedure. This presents fewer pragmatic difficulties, but it does leave wide open the question of structural validity, a correspondence between the description and what is being described.

Assuming that we do come to the conclusion that HMS's really do use procedures or at least that we can describe their behavior in terms of procedures, we come to the problem of detection. How do we KNOW when a system is executing a procedure, when that procedure is active, and when it is suspended? One possible operational way to tell if a procedure is active is to require that the execution of ALL procedures begin with some readily observable action and terminate with another readily observable action. Then, if we have observed the initial action but not the final one, we can say that the procedure is active. We might, on the other hand be able to infer procedure activation and execution by monitoring brain activity with an electroencephalograph. Both of these are pure speculation, though, and considerable research would need to be done before they could even be tried in any realistic situations.

The issue of procedure representation and storage has already been alluded to and there may be nothing unique about how procedures are stored in knowledge bases, whether they be paper, magnetic disks, or human memory. However,
how procedures are accessed, and the processing mechanism that accesses them do pose some interesting questions. It is tempting to use the concept of the digital computer as an analogy for the processing of procedures by human-machine systems. We can interpret the human brain or a handbook containing a procedure as the computer memory where a program is stored. Then instructions are accessed sequentially by means of one or more registers serving as pointers, and the instruction is decoded and processed. The analogy is bound to break down, though, since there are clearly restrictions on the computer that do not exist for the human, such as limited capacity, and it is reasonably clear that the storage mechanisms are quite different. Nevertheless, the analogy could prove useful if its limitations are recognized.

The idea of procedures, either as explanatory or just descriptive entities, seems useful. Possibly the next step in the study of procedures is the categorization of them for specific classes of systems, based on things like time needed to complete, resources required for execution, skill level required, and the classes of goals which they are likely to be useful in achieving. A taxonomy of procedures could prove to be very useful to the HMS designer, for it seems that the design of procedures, entities that organize system activities, are at least as important as the hardware that make such systems up.

Goal priority was mentioned above as an issue, similarly, priority of procedures seems important. The most straightforward approach is to directly relate a procedure's priority with that of the goal which it is to achieve. More will be said about procedure priority in the next section.

A correctly functioning computer does not make "mistakes" in executing programs (we will ignore for the moment the fact that cosmic rays have been shown to alter bits in memory) but human operators are prone to mistakes, even in well-rehearsed activities under conditions of low stress. By considering these activities to be guided by procedures, it might be possible to identify in particular procedures just where mistakes or errors are likely to occur. If errors like these can be foreseen, there may be opportunity to avoid them through preparation or redundancy.

Coordination of activities is a very interesting matter in HMS design and analysis and there seems to be a number of ways that the concept of a procedural
representation could account for it. For example, since the executive procedure controls all others, a certain amount of coordination could be explained in terms of that control. Additionally, it seems reasonable to advance the notion of procedure processor communications. That is, two procedures which required coordination could communicate via the inputs and outputs of their respective processors.

In a way it is tempting to define generalized procedure processors instead of specialized ones, which, like computers capable of executing many programs, could execute multiple procedures. An advantage of this would lie in the economy of the procedural representation. On the other hand, though, this economy would be purchased at the price of increased complexity of the generalized processors.

One final issue that procedures raise relates to skill. Can we define a motor skill as a procedure? It seems clear that a skilled athlete, musician, or operator does not consciously follow procedures. Nevertheless, it may be possible to describe these skills as if they were procedures.

It is obvious that whether we call them knowledge bases, data bases, or something else entirely, there are entities for information storage in all human-machine systems, so their existence is really not an issue. There are, however, a number of other issues that do bear some investigation. We can first cite the problems of media and organization. That is, how is information stored in a particular knowledge base and how is it organized? In the case of knowledge bases like handbooks and computer memories, these problems are more straightforward, but in cases of human memory they are very significant. Again, we can consider the digital computer analogy, thinking of the study and design of data bases where terms like "linked lists" and "associative data bases" are important. It would be very interesting to determine if the analogy holds for such concepts as these.

A taxonomy of knowledge bases might be a useful undertaking, in which type of information stored, form of storage and access, and other factors would be of concern.

In any given human-machine system, it seems unlikely that all stored information is kept in a single knowledge base. Consequently, the issue of distributed knowledge bases is raised. There is area for study here concerning the number and kinds of knowledge bases a particular HMS
with a particular set of goals should possess and of the function and characteristics of each knowledge base. The issue of redundancy is raised here, too. Redundancy is not always bad, especially where security and survivability are concerned, but its implications must be understood. For instance it must be understood that redundancy has both syntactic and semantic aspects. That is, the arrangement of particular symbols may differ in two knowledge bases, but the meaning of the information therein may be identical (i.e. redundant). This may prove to be wasteful or it might be useful, based on the prevailing circumstances. (Consider, for example factual and procedural knowledge in human memory. Procedural knowledge about golf swing is essential to the golf pro in competition, but factual knowledge is more useful when training novices, even though it may be considered redundant.)

Implicit all along has been the concept that a knowledge base is not always a dedicated entity. That is, a resource may be a knowledge base and at the same time may serve other functions as well.

5.1.2 Some Issues Raised by the Axioms

Possibly more significant than the issues raised by the terms of Chapter 3 are those raised by the axioms that result from combining and giving additional interpretation to the terms. To introduce some of these issues, let us briefly recap the axioms by summarizing them.

A human-machine system develops procedural plans to achieve goals and follows these plans in activating and executing procedures. Procedures compete for output channels and, consequently not all active procedures can execute simultaneously. The executive procedure controls the activation and execution of procedures.

The first issue that this raises is at what level of description of an HMS these axioms apply. That is, based on the time-sharing literature this all seems reasonable when applied to a single operator, but does it also apply to multi-operator HMS's and, if so, where does the control structure in such a multiple human system reside? If we are proponents of any one of a number of Oriental views of the universe, we really have no trouble here for we believe that the distinctions between things in general and between people in particular are purely artificial and not
particularly healthy to maintain. Even from a more Western perspective, it seems obvious that in any multi-operator HMS there is a good deal of coordination and cooperation among the operators in deciding what is to be done, what the priorities are, and what particular activity should be pursued at a given moment. It is even possible that in the near future we will see machines taking a more active role in this process as well, which is another issue in itself.

It is well understood by all of us that planning is an important part of human activity. It is also clear that the concept of a procedural plan as discussed in Chapter 3 is not nearly general enough to embrace the entire concept. It is, however, a reasonable compromise, linking the concept of a set of goals to sets of procedures to achieve those goals. The existence of plans really needs no validation, but the existence of procedural plans as defined does. Again, though, this may be best addressed by claiming that even though procedural plans may not exist, the concept is useful for organizing knowledge about the purposive and directed behavior of human-machine systems.

If procedural plans do exist, there must be a way of representing and storing them in knowledge bases, for access by the mechanism controlling the HMS's actions. It seems clear that representing and storing an event depends in part on being able to develop a predicate, indicating what state transitions constitute an event. Representation and storage of a procedure is more straightforward.

If HMS's do use procedural plans, then there must be a mechanism to create them. For the following discussion, we will postulate that there exist one or more procedures in an HMS for creating procedural plans.

Any mechanism or procedure to construct procedural plans from goals must be capable of goal decomposition for, in general, there will not exist pre-defined procedures for achieving these goals. The decomposition of a goal may result in two classes of structures, a time decomposition or a subsystem decomposition. A time decomposition of a goal is a set of subgoals that are to be achieved sequentially, the achievement of which constitute the achievement of the original goal. Subsystem decomposition, on the other hand results in a set of goals which must be achieved essentially simultaneously but by different subsystems of the system for which the original goal was intended. In general, the decomposition of a goal will result in a set of goals best described as a combination of the above.
In any case we must consider the question of whether the achievement of all subgoals of a given goal is a necessary condition for the achievement of the original goal. In the case of redundant subgoals, the answer seems to be no, but where no redundancy exists, it seems likely that we must require this condition.

Having accomplished the decomposition of a goal, a procedure to develop a procedural plan must be capable of selecting or synthesizing procedures to achieve the subgoals. Here again the concept of an internal model seems appropriate, for the ability to select a suitable procedure from a number of alternatives must depend to a large extent on being able to predict the results of the execution of each of those alternative procedures.

Any procedure to develop procedural plans must be capable of defining the events that are to trigger the activation of the procedures in the plan. This seemingly must involve the prediction of a chain of future events (again, possibly derived through an internal model) and the recognition of key events that signal the need for a particular procedure to achieve a goal or subgoal. These events then need to be encoded and stored as part of the procedural plan, a process which should be of some interest.

It seems likely that all procedural plans must be subject to change based on changing goals and conditions. The dynamic nature of procedural plans and of the planning process seems to be an important issue.

Procedure competition, while here couched in different terminology, is a subject that has been of interest to psychologists and human factors engineers for some time, for the operator's ability to perform often conflicting activities simultaneously is of great importance in complex human-machine systems. We may wonder first if there isn't some limitation on the number of procedures that can be active in a given HMS before confusion causes a breakdown in performance. We also may wonder if there aren't procedures inherently conflicting in nature, whose simultaneous active states will produce such a breakdown. The answers to these questions, while seeming to be in the affirmative, need more investigation to gain more insight into the control mechanism.

Capacity limitations have been of considerable interest for a long time, but usually with respect to sensory and motor activities. Here we have postulated that
the capacity for a human-machine system to execute procedures is limited. Like some of the concepts surrounding the sensory-motor limitations we might consider capacity to be described as channel capacity or pool capacity. Were we to develop a channel theory of procedure capacity, we might say that each procedure required the exclusive use of particular input and output channels of a particular system and any other procedure requiring any of those channels could not execute simultaneously. This approach seems consistent with our definition of systems. A pool theory of procedure capacity, on the other hand, might state that each procedure required a certain amount of a system's processing capability and once that capability was used up, no other procedures could execute.

In any case, there would seem to be a number of factors influencing the capacity of an HMS to execute procedures. In humans these factors could include innate ability, level of stress, physiological condition, fatigue, and level of training. If we interpret our own personal experiences in terms of goals and procedures, we know that our ability to execute procedures changes with time and possibly as a function of these and other factors. Consequently there should be some advantage in being able to predict capacity limitations and the problems that might result from them. If such a predictive capability could be developed for specific HMS's, it might be possible to at least avoid situations in which conditions of overload occurred.

Although the fact that there is some mechanism or mechanisms by which a human-machine system organizes its activities, the existence of an executive procedure may be the most controversial and difficult to validate claim made by the axioms. Even if there is such a procedure, there must be a great deal of uncertainty about its nature, characteristics, and in what system or systems of an HMS it executes. If we assume for the moment that such a procedure exists, or at least that this perspective provides us with a useful way to describe HMS's, a number of issues are raised.

First we must speculate that the executive procedure, to keep things organized, stores information about goals, procedures, and events in knowledge bases, and uses this information to coordinate its activities. The issue of how this information is stored has already been alluded to, but the manner in which the executive stores, accesses, and utilizes the information poses additional questions.
Since the executive is responsible for activating procedures based on the occurrence of events, it must be capable of detecting these events. It seems likely that such an activity could be best explained in terms of some sub-procedure or sub-procedures responsible for event detection. This raises the question of the significance of false alarms and missed events. If an HMS is to follow a procedural plan by activating procedures based on events, what is the result of an event which occurs but is missed by the executive? Of course this type of problem has been studied for a long time under the name of signal detection theory, but this study has been directed primarily at the detection of specific sensory events such as targets on radar screens, not at the mechanisms which control human and system behavior.

Once the executive has a number of active procedures to execute and given that there is not enough capacity in the system for them all to execute at once, the executive must determine which procedures execute and which are suspended. One way to describe this situation is in terms of priorities and requirements. Priorities may be assigned to procedures based upon their relative immediate importance in terms of the goals that are active in the HMS, and these priorities may be subject to alteration according to changing goals and conditions. In addition, each procedure, as discussed before, may have certain requirements, such as the requirement for certain resources, certain input and output channels, and certain amounts of system processing capacity. The executive may well determine which procedures execute based on priority. That is, the procedure with the highest priority and for whom all required resources, systems, channels, and processing capacity are available is allowed to execute. The procedure with the next highest priority is then considered for execution and so on until all resources and capacities have been exhausted or until no more active yet suspended procedures exist. Each time an event is detected, this process is repeated.

If such a situation holds, then we must expect that procedures that are executing are occasionally preempted by other, higher-priority procedures. For example, a pilot flying straight and level, alerted by a subsystem failure detection indicator, must interrupt his/her normal duties to determine what malfunction has occurred. In procedure terminology, a straight and level flight procedure is executing when an event, the sounding of a buzzer or the flashing of a light, causes the pilot's executive procedure to activate the fault detection procedure. Since this
procedure has a much higher priority than the flight procedure, and since it requires some of the same resources (eyes, hands, etc.) it begins executing and the flight procedure is suspended for a time. Once it is determined, for instance, that the failure is not in any critical system, the fault detection procedure's priority decreases so the pilot can resume flying again.

In some cases, as above, suspended procedures can be resumed at the point of suspension, but this is certainly not always the case. Consider for example what might happen if the above failure occurred during final approach to landing. If the criticality of the failure could not be determined in time, it might be necessary for the landing to be aborted and a go-around to be made, after which the landing sequence could be repeated. In other words, the landing procedure is interrupted by the fault detection procedure and it remains suspended while the missed-approach procedure executes. Then at an appropriate time, following an appropriate event (such as arrival at a particular landmark on the downwind side of the runway), the landing procedure is "backed up" and resumed. In a case like this, pilots are trained to expect such situations and, assuming no major failure has taken place, the landing can be made safely. However, we must wonder what, in general, the penalties are for interrupting and resuming the execution of procedures. Certainly this occurrence provides ample opportunity for confusion, errors, and subsequent failures to take place, and a better understanding of the phenomenon could be of great help in planning HMS activities.

If it is accurate to view human-machine systems as being controlled by executive procedures or if it is at least useful to describe them in those terms, we would be well-advised to consider the possibility of enhancing the operation of the executive procedure. There is some evidence (Gabriel and Burrows, 1968) that time-sharing performance can be improved by training. It seems possible then that the process of executive control, which provides a somewhat more general explanation of activity organization, could also be enhanced by training. It might be possible, for example, to train operators to quickly prioritize procedures and then allocate their attention to the highest priority procedures. Some enhancement might also be possible through the use of aids, such as dynamic, computer-generated displays of active procedures, listed according to priority. Finally, it may be possible to improve HMS performance by selecting operators based in part on their abilities to handle the procedure control
process efficiently and successfully.

Human errors in the operation of HMS's are of extreme concern to researchers, designers, and analysts. It would be very helpful if the structure developed here could provide some insight into this important problem. This capability does seem present for a number of reasons. First, a procedural representation of a human-machine system can make mistakes in a manner similar to that of humans. If an unexpected event occurs or if an expected event occurs at an unexpected time, either no procedure to deal with it may be invoked or an inappropriate procedure may be invoked. Additionally, certain limitations could be built into the executive in a procedural representation, limitations which would allow the executive to become overloaded and confused, or limitations which would allow it to miss events. Such error-prone behavior may help us to understand how and when humans make mistakes.

5.1.3 Potential for Integration

As was pointed out in Chapter 2, most existing theories of human-machine systems, such as manual control theories and discrete event theories, are too restrictive to provide a complete description and understanding of HMS's. It was hoped that the structure developed in this dissertation would provide a framework in which the existing theories could be integrated so that breadth in scope yet precision and detail in description could be achieved.

Two very important issues raised by the research, then, are whether or not the resulting structure is consistent with existing theories, and whether or not it can serve to integrate some of them into a useful, unified whole. It is our opinion that such a potential does exist. Specifically, the structure is capable of accommodating certain manual control theoretic, decision theoretic, and information theoretic descriptions as substructures. Important aspects of the structure are consistent with and can be described in terms of queueing theoretic descriptions of HMS's. Finally, the structure is consistent (or at least not contradictory) with discrete event theories of HMS's.

To substantiate and illustrate these claims, consider an HMS consisting of an aircraft and a single pilot. We
could go about describing this system in a number of ways based on existing HMS theories. For example we could describe the pilot as an optimal controller with respect to controlling the aircraft's heading, pitch, and yaw. We might use utility theory to describe the pilot's decision making activities with respect to selecting routes and strategies. We could also describe the pilot as an information processor. Queueing theory could be used to describe the pilot's attention to particular activities and we could write a discrete event simulation of the system in which the pilot's activities were triggered by the occurrence of specific events.

With so many possible perspectives, is it possible to bring these approaches to describing the system together into a systematic, unified structure? Although the final answer to this question must wait on specific research, the potential for an affirmative response is high. For instance, if we describe this HMS in terms of goals, procedures, and dynamic procedure control, as put forth in the previous chapters, we can incorporate optimal control concepts into the description in the form of an optimal control procedure. Decision-making activity can be described as another separate but (possibly) simultaneously active procedure. Either or both of these procedures could be described in terms of the information processing that must occur in any control or decision-making activity. We could describe the processing of procedures by the HMS in terms of procedures queueing for service (execution) and the queueing theory concepts of priorities, and multiple servers might be used to an advantage. We could also describe the situation in terms of discrete events where event occurrence produces not an activity, but instead the activation of a procedure which, when executed, brings about the activity.

5.2 Measures, Performance, and Workload

An issue that must be raised in almost any discussion of human-machine systems is the one concerning measures and their use in HMS research, analysis and design. One of the primary purposes of HMS research is to produce better, more efficient, and more effective HMS's. This process, involving the comparison of alternative HMS's and alternative designs, almost necessitates the measurement of system characteristics and performance. In this section we will discuss measures, especially as they relate to HMS
performance, and we will propose some concepts, based on the terms and axioms of Chapter 3, that may be useful in predicting HMS performance.

5.2.1 Measures

Informally speaking, a measure is a means for describing a (possibly) complex object or phenomenon in terms of some simpler kind of structure having rather well defined mathematical properties. Commonly used measures allow us to classify physical objects according to size, weight, or volume. Other measures are claimed to tell us in a very concise manner how intelligent a person is or what shape our country's economy is in.

More formally, a measure may be defined as a relation on two sets. Let $O$ be a set of objects. The elements of $O$ may be physical, like metal disks or cats, or they may be more abstract, like concepts or the behaviors of systems that we have been talking about. Let $V$ be a set of values. $V$ is usually linearly ordered, but, as we shall see, this is not always the case. A measure, $M$, may then be defined:

$$M \subseteq O \times V$$

In other words, a measure associates one or more values with an object. We usually interpret the values assigned to the object in terms of some units, inches or centimeters for example.

To illustrate, let us consider a measure of height. Let BLOCKS be a set of blocks defined:

$$\text{BLOCKS} = \{ b_1, b_2, b_3 \}$$

We can then define $M$ to be a measure of height:

$$M \subseteq \text{BLOCKS} \times \mathbb{R}^+$$
Where \( R^+ \) is the set of non-negative real numbers (including 0, of course), and the measure is interpreted in millimeters. Practically speaking, we can measure the height of the blocks \( b_1, b_2, \) and \( b_3 \) by means of a meter stick or a metric ruler. Consider the following elements of \( M \) that deal with block \( b_1 \):

\[
\begin{align*}
(b_1,9.95) & \in M \\
(b_1,10) & \in M \\
(b_1,10.05) & \in M
\end{align*}
\]

Depending on who measures the height of \( b_1 \), when it is done, or what instrument is used for the measurement, different values of its height may be found. Such an indeterminate situation is not highly desirable so we usually wish to define a measure as a function. For example:

\[
\begin{align*}
M : BLOCKS & \to R^+ \\
M(b_1) & = 10 \\
M(b_2) & = 20 \\
M(b_3) & = 30
\end{align*}
\]

Here just one value is associated with a particular block. This can be enforced by requiring that the measure be made at one particular time for each block, by the particular individual, or on one particular instrument to insure uniqueness.

Note that the measure is one-dimensional and that \( R^+ \) is linearly ordered by the relation \(<\). We can also define multi-dimensional measures. Formally we might have:

\[
M : O \to V \times V \times \ldots \times V \\
\quad 1 \quad 2 \quad \ldots \quad n
\]

where \( V, V, \ldots, V \) are linearly ordered value sets. Formally we might have:

\[
M : O \to V \times V \times \ldots \times V \\
\quad 1 \quad 2 \quad \ldots \quad n
\]

In this case, however, the cartesian product is not
generally linearly ordered, but we can define a function on it which maps to a linearly ordered set. For example, determining the dimensions of a block (height, width, and length) in centimeters is a multi-dimensional measure, mapping the set of blocks to the cartesian product $\mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+$, which can then be mapped to $\mathbb{R}^+$ and interpreted as volume.

Measures do several things for us. They provide us with a structure, defined by the value set and any relations defined in turn on it, to organize our knowledge about some aspect or aspects of a group of objects, often providing an easier or a more convenient way of thinking about those objects. Measures help us simplify complex phenomena, maybe to get at some particularly important property, collapsing a lot of irrelevant information into a very concise form. We use measures to categorize things, to place them in classes or sets to be dealt with as a group rather than singly. Finally, measures aid us in deduction. We use the values provided to us by measures to reduce things to a common means of expression for purposes of comparison, computation, and, possibly, prediction.

The concept of a measure raises a number of issues, both formal and informal. Formally speaking, when we talk about a measure, we must define a number of sets, which is nearly always an exercise in careful consideration and precise description. We must first define the domain of the measure function. Practically speaking, this means that we must define what it is that we are measuring, an issue not as significant when measuring blocks as it is when measuring intelligence. Next we must define the codomain of the function, the value set, and the relations defined on it. More often than not, the value set is some subset of the set of real numbers, but there is no requirement that this be the case. Finally, we must define the function itself, or at least state some predicate that characterizes it.

One thing that we must be very careful about in defining measures is to be very clear about what properties exist in the domain (what is being measured), how these properties are translated by the function into the codomain (the value set), and what statements we might be led to make based on the properties present or presumed to be present in the codomain.

For example, we use IQ to "measure" a person's intelligence. Apart from the fact that the domain of this measure remains undefined, we can rate, say, one person's
intelligence at 50 and another's at 100. Does this mean that the second person is twice as intelligent as the first? By using a subset of the reals as the codomain for this function, we seem to imply that, but any number of people would be willing to argue strongly against that.

Another issue raised by measures is the problem of the inverse. Since, in general, a measure is not a bijection, the inverse of any value consists of a (quite possibly very large) set of objects. For instance we can refer to the set of all objects that have a mass of 100 grams, but that tells us very little else about the nature of the objects in that set. We must be very careful in using the values provided by measures to speak about the measured objects.

Defining a function is one thing. Implementing it is quite another, for to implement it we must in effect find a way to compute it. A measure of height is relatively easy to implement through the use of a very simple device called a meter stick. Measuring the gross national product of the US, on the other hand, is very difficult, not, like intelligence, because we don't know what we are measuring (although that can be argued) but because it is operationally impossible to determine and record every bit of financial activity that takes place in the country. Only estimates and approximations can be made.

5.2.2 Performance

A performance measure is a special kind of measure applied to the behavior of a system over time. Let \( S : C \times X \rightarrow Y \) be a system as defined before on the time set \( T \) and let \( V \) be a linearly ordered set. A performance measure may then be defined:

\[
PS : S \rightarrow V
\]

Even though we are measuring something more abstract than the blocks and such that we talked about before, the only really significantly different characteristic of a performance measure is that the objects being measured are defined on a time set. We use a performance measure to characterize behavior over time.
If we use an aircraft and its crew as a vehicle for discussion, we can think of a number of examples of performance measures, based on the observed behavior of this human-machine system. Such examples could include mean airspeed, mission completion time, number of targets hit, number of errors committed, and so on. These measures all map behaviors to subsets of the real numbers, but other value sets may be employed. For example, we may map system behaviors to the set $V = \{ \text{success, failure} \}$ where "is preferable to" is the ordering relation (success is preferable to failure).

It is of course not enough that we be able to classify behaviors according to performance, although that is useful in organizing knowledge about missions and so forth, but to design systems, select and train crews, and plan missions, it is necessary to be able to predict performance. That is, given a set of system parameters, such as crew size, experience, and equipment available, we want to be able to say, with a reasonable degree of confidence, how that system will perform, what the value of that performance will be, and consequently from what class of behaviors that value will be derived.

Other measures can be of help to us here, for we can often relate performance to other properties of system behavior as defined by simpler measures, but this is not always the case. We sometimes need to use additional structures to organize knowledge about systems in order to make statements about how we believe that they will perform.

5.2.3 Workload

As mentioned above, there is a great need in the area of human-machine system research to be able to predict the performance of human-machine systems. There are two major practical problems here. First, the hardware and software components of HMS's are rapidly becoming so complex that even their designers cannot always accurately predict how an individual component will perform under certain circumstances, much less how large systems of these components will behave. The second problem is of more concern to us here. It is very difficult to predict how human operators will perform under various circumstances. Some have been hopeful that by describing the operator as a servomechanism with delays and other infirmities or as a
decision maker with some notion of utility, prediction of the human component of HMS performance could be enhanced, and it has been to a certain extent.

Unfortunately, most of our past attempts have been subject to some severe limitations. Human operators seem to have a capacity to perform activities and when that capacity is exceeded, their performance deteriorates. The concept of mental workload was invented to deal with those issues.

Although several definitions of workload will be examined below, we can present an intuitive definition of the term here. Mental workload (or, simply, workload) can be regarded as a mental analog of the physical load resulting from applying force to the free end of a cantilever beam. Workload is said to result from requiring a human operator in an HMS to perform activities. In general, the more activities the operator must perform, the greater is his/her workload.

Like the cantilever beam analog, there seems to be an "optimal" level of workload. Too little load and the beam is overdesigned or underutilized. Too much load and the beam fails. Too little for the human to do and he/she becomes bored. Too much to do and system failure might result from an operator error.

We are interested in workload for just this reason. We wish to design systems in which the operators are not underutilized or bored, but in which the operators are never so loaded that system operation is jeopardized. We believe that a better understanding of operator workload can assist us in predicting operator and system performance, that a knowledge of its characteristics can help us predict and avoid performance breakdowns, and that such knowledge could lead to better system designs. In addition, an understanding of operator workload could lead to a better understanding of the operator him-/herself.

5.2.3.1 The Current State of Workload Research -

No attempt will be made here to thoroughly review the workload literature, for this has already been done by a number of authors. For example, Mental Workload, the proceedings of the 1977 NATO Symposium on the Theory and Measurement of Mental Workload, edited by Neville Moray (1977), provides a rather comprehensive picture of the
state-of-the-art with respect to workload. Also Gartner and Murphy (1976) have surveyed the literature adequately.

However, an introduction to some of the important concepts surrounding the term is in order. The following sections will present a brief overview of some of the currently used definitions of workload and some of the more common measurement techniques.

5.2.3.1.1 Definitions of Workload -

Some workload definitions do not go far beyond the intuitive cantilever beam analogy presented above. One definition of workload is simply how busy the operator is (Ogden, et al, 1979). Others go a bit farther in defining workload in relation to the frequency of occurrence of events (Schmidt, 1976).

A number of workload definitions are based on the operator's subjective feelings. Borg (1978), for example, defines workload as subjective intensity as perceived by the operator. Sheridan (1980) feels that workload must be defined as subjective experience of cognitive effort.

Many workload definitions are related to demands placed on the operator. Gartner and Murphy (1976) define pilot workload as how much a pilot must do to satisfactorily perform flight operations. An economic analogy is used by Teiger (1978) when he defines workload as the cost to the operator of employing psychophysiological functions to respond to task requirements. Chiles (1979) defines workload as the aggregate of occupational demands placed on the worker.

Another dimension is added to demand, that of capacity, by several authors. For example, workload can be defined as the mental effort an operator devotes to control and supervision relative to his/her capacity to expend mental effort (Curry, et al, 1977). Welford (1978) defines it as the relationship between task demand and the operator's mental and physical capacity.

Other authors treat workload as an inherently multidimensional entity. Johannsen (1977), surveying the work of several others, defines workload in terms of input loading, operator effort, and performance.
5.2.3.1.2 Workload Measures -

The objective of most workload research is to measure workload and relate it to HMS performance. Quite a number of measurement techniques have been developed, including primary and secondary task measures, subjective opinion ratings, the measurement of physiological variables, and task analysis. For each of these, a brief description and a reference treating the subject in some detail will be given below.

In primary task measurement techniques, an operator is asked to perform some activity in which the level of difficulty is varied. The operator's performance, such as error rate in discrete activities or RMS error in tracking experiments, is taken to reflect the level of workload. Williges and Wierwille (1979) discuss primary task measures of workload.

In secondary task measures of workload, the fundamental assumption is that the operator has a finite capacity to perform monitoring, decision making, and control activities and that workload is related to how much of this capacity is used up by a particular activity. To measure this, a subject is given a high priority activity which he/she is expected to perform as well as possible, such as continuous tracking. Spare mental capacity is measured by how well he/she performs a secondary, lower priority activity. Ogden et al (1979) describe secondary task measures in much more detail and give a number of examples. Some of the secondary tasks surveyed include mental arithmetic, tracking, choice reaction time, tapping (where the subject is asked to press a switch at as constant a frequency as possible and the regularity is measured), event detection, and piano playing.

Williges and Wierwille (1979) also discuss the use of subjective measures in measuring workload. In such cases, subjects perform monitoring and control activities under various conditions and are afterwards asked to rate the level of difficulty or intensity experienced.

Some researchers feel that by measuring certain physiological variables, on-line detection of workload levels can be achieved. Williges (1979) describes many of these efforts. Variables measured include heart rate, blood pressure, eye movement, and pupillary dilation. Other techniques include the analysis of respiration, body fluid, and speech patterns.
In task analysis measures of workload, described by Sheridan and Stassen (1977) and Soede (1977), a synthetic approach is used. First, the levels of difficulty or workload of very simple activities, such as event detection, instrument monitoring, or tracking are developed. Then complex activities are decomposed into these simple activities. An aggregate workload level is then synthesized.

5.2.3.1.3 Shortcomings of Current Methods -

The existing "workload" measures seem to focus on primary and secondary task performance and physiological variables. But is primary task performance workload? Is secondary task performance workload? Is heart rate workload? What is new or different about using one kind of performance to predict another? It seems clear that these measures do not really measure what their proponents say they that are interested in. And though those who advance the subjective and synthetic approaches cannot be immediately disqualified, it may only be because they have not truly defined what it is that they are measuring. It seems as though all of the workload measures we have briefly examined are, at best, exploratory efforts aimed at determining if there is some characteristic of human behavior that can provide a key to understanding and predicting performance. The researchers and analysts who design and use them cannot be faulted for that, but they should at least be careful about the claims that they make.

But still, the issue remains. Is the idea of workload a useful one? If clearly defined and utilized, can it give us any insight into HMS performance? Since the purpose of this chapter is to explore some of the issues raised by Chapter 3, perhaps a look at workload from a procedures perspective might be interesting.

5.2.3.2 Performance and Workload -

If a procedural representation of a human operator is a reasonable way to describe the way he/she organizes activities, then to use procedures in a definition of workload makes some sense. Perhaps one useful way to begin to define workload is in terms of procedural load, that is, in terms of the procedures that are active at a given time.
Let HMS = (S, GOALS, PROCEDURES) be a human-machine system as before with S defined on the time set T. Then let PLOAD be a map defined:

\[
PLOAD : S \times T \rightarrow \mathcal{T} \text{ PROCEDURES} \times \mathcal{T} \text{ PROCEDURES}
\]

where:

\[
(\Psi (s,t,(\text{EXEC},\text{SUSP})) \in PLOAD)
\]

\[
\text{proc } \in \text{EXEC } \iff \text{proc is executing in } s \text{ at time } t
\]

\[
\text{proc } \in \text{SUSP } \iff \text{proc is suspended in } s \text{ at time } t
\]

This map gives the active procedures at any time. It further distinguishes between executing and suspended procedures.

Although experimental validation would be necessary, it seems as though looking at procedural load could give an indication of the demands placed on a system at a given time. Executing procedures would logically place certain immediate demands on an operator but the presence of active yet suspended procedures could give an indication of how "busy" he/she was.

Of course the first practical limitation to this approach that comes to mind lies in our potential ability (or lack thereof) to implement the function PLOAD. That is, how do we detect if and when a procedure is active, let alone distinguishing between whether it is executing or suspended. A number of possibilities for dealing with this problem exist. First, in certain experimental or operational situations, it may be possible to require that the execution of each procedure begin and end with a readily observable action. We could then determine when these procedures were active. The recognition of executing procedures would be a bit more difficult, though. Possibly electroencephalography could be useful if it could be shown that certain types of brain activity are associated with the operator's attention to particular activities. The monitoring of eye movements might be helpful, especially when the execution of procedures was closely tied to certain displays or areas of an operator's visual range.
But even if we could determine what procedures were executing and suspended at a particular time, would that be enough? If we look at the cantilever beam analogy of workload, we see that by just considering the load placed on the beam, we cannot, in general, predict whether or not the beam will fail. It is necessary to know the load bearing capability of the beam as well, its capacity to sustain the load without failing.

It therefore seems reasonable to say that a concept of workload without consideration of workload "capacity" is meaningless. Procedural capacity could not tell us anything unless we knew something about procedural capacity.

Informally speaking, we might define procedural capacity as the ability of a system to have procedures executing and suspended without an accompanying deterioration of performance. What the formal requirements for such a concept are is not now clear, but it must be consistent with our definition. That is, there must be some way to compare a system's procedural capacity with its current or projected procedural load to be able to make statements about how the system will behave.

The fact that we are unable to answer all the questions about determining procedural load and the fact that we cannot even formally define procedural capacity at this time does not mean that the concepts are useless. The ideas of goals and procedures were useful in organizing knowledge about a bicycle and rider, perhaps procedural load and procedural capacity will be useful in organizing knowledge about demands placed on operators and their abilities to meet them. The area of design might benefit the most from this approach. By analyzing a proposed systems in terms of the procedures that it must execute and how those procedures must interrelate, insight might be gained into how the operators in the systems would perform. It seems as though the approach deserves additional investigation.

5.3 Summary

In this chapter, some of the implications led to by the structure described in Chapters 3 and 4 were briefly discussed. Issues raised by the terms and axioms were mentioned and the potential for the use of the structure to
integrate existing human-machine system theories was explored. A more detailed inquiry into the nature and purposes of measures was made with special attention given to performance measures. Finally, the concept of workload was explored to see if it might offer some assistance in HMS performance prediction.
CHAPTER 6

Summary, Discussion, and Conclusions

This final chapter is intended to summarize the research that has been done, to analyze it in relation to some of the broader concerns that have motivated it, and to offer some possible conclusions that may be drawn from it.

6.1 Summary

First, a brief summary of the major points will be covered to give an overview of the research.

6.1.1 Languages, Theories, Models, and Formalization

A language is a set of strings of symbols from an alphabet that are combined according to some set of rules (syntax) and which may have some particular meaning (semantics). These strings of symbols are generally called sentences in the language and, whether or not the language is natural (such as English or Chinese) or formal (like FORTRAN), it is usually possible to determine whether or not a given arbitrary string of symbols is a sentence in the language. In the case of formal languages, this is always true.

A language may have several classes of sentences. A statement is a kind of sentence to which is assigned a truth value (true or false) when interpreted in terms of some subject domain.

A theory is a collection of statements about some subject domain. It may be further broken down into constituent parts including primitives, defined terms, axioms, a logical calculus, and the theorems which may be
derived from the axioms using the logical calculus.

A model of a theory is a structure in which the statements of the theory are interpreted as true. Therefore, while we usually think of other physical and mathematical structures as models (which they may be) the subject domain of the theory is a model of that theory too.

Formalization, as considered in the current research, is a process of developing a theory from the primitive notions of sets, elements, and relations. The first step, called axiomatization, consists of identifying observation terms, primitive terms, defined terms, axioms, and a logical calculus. The second step involves the interpretation of the terms, axioms, and resulting theorems by models. Formalization provides a way to precisely and unambiguously describe a subject domain in a way that can lead to clearer understanding.

6.1.2 Human-Machine Systems

Informally speaking, a human-machine system is a set of interrelated components, some human, some machine, which function together to achieve objectives or goals that could not be achieved consistently, efficiently, or safely by any of the components individually. The types of human-machine systems of particular interest in this research are highly automated, and have, in general, multiple operators who function more as monitors and supervisors of the system than as moment-to-moment continuous controllers.

Examples of such human-machine systems include flight management systems (modern aircraft), air traffic control systems, and the control systems of large electric power generation and distribution systems.

These complex systems pose many important analysis and design problems. A number of theories have been developed to describe them in such a way as to address some of these problems. For example, using a control engineering perspective, the human operator has been described as a(n optimal) servomechanism which interfaces with the rest of the system through mechanical and electronic components. Information theorists have described the operator as an information channel while decision theorists treat him/her primarily as a decision maker in an uncertain environment. Various subsystems of the HMS may be described as queueing
systems. The entire HMS may be described in terms of the occurrence of and response to discrete events.

There are two major shortcomings of such current HMS theories. First, by being limited to restricted domains (such as the operator performing continuous control activities) they are too limited to address the global problems of HMS analysis and design. The other major problem is that those theories which attempt to describe a broader scope of HMS issues usually lack the precision and detail necessary to make meaningful and helpful statements. There exists a need for an HMS theory which is sufficiently broad in scope yet precise and unambiguous enough to avoid these shortcomings.

6.1.3 Formal Development

In Chapter 3 a collection of terms and axioms, intended to serve as the foundations for a new theory of human-machine systems was formally developed. The defined terms include systems, events, goals, procedures, and knowledge bases.

A system is a structure defined on a set of inputs, a set of outputs, and possibly a set of states. Systems may be connected to form more complex systems or may be decomposed into simpler subsystems.

An event is a set of possible state transitions. An event is said to occur if one of the state transitions in the set takes place.

A goal is a statement of what system behaviors are acceptable.

A procedure is defined by a name, a string of symbols (the steps or instructions of the procedure), and a procedure processor, a system whose state object includes the symbols in the symbol string. A system may be described in terms of procedures if a collection of procedure processors can imitate it.

A knowledge base is a special kind of system capable of storing procedures and other information for later retrieval.
A human-machine system may be defined formally by a system, a set of goals, and a set of procedures. Human-machine systems execute procedures to achieve goals by developing procedural plans and activating and executing procedures according to those plans. Since a human-machine system cannot execute all active procedures simultaneously, an executive procedure controls the activation and execution of the other procedures.

6.1.4 An Interpretation of the Terms and Axioms

In Chapter 4 an interpretation of the terms and axioms was presented. A system consisting of a bicycle and rider was chosen to illustrate the important concepts and to avoid certain complexity issues. A sequence of formal structures for describing the bicycle and rider stopping at a stop sign was presented, culminating in the description of the bicycle and rider as a human-machine system in which the rider was described as a processor of procedures.

6.1.5 Some Implications of the Structure

The potential value of a new structure like the one developed in Chapter 3 and interpreted in Chapter 4 can perhaps best be tested judged initially by examining some of the issues and implications that it raises. Chapter 5 presented a number of these issues and implications, both those raised by the terminology and those raised by the axioms. Then, taking a somewhat broader perspective, the formal approach used in the development of the terms and axioms was used to focus on measures, performance, and workload. Some observations based on this approach were made about current efforts in workload research.

6.2 Discussion

This dissertation and the structure it presents can be interpreted as a contribution in two areas. The first area, to which the dissertation was directed, is the area involving the design and analysis of human-machine systems. The second, and more general, area is that of scientific
inquiry.

To the field of HMS study, the dissertation has contributed a framework for describing human-machine systems. We have developed a set of terms, precisely defined on some very basic notions of input and output sets. By doing this, we have forced those who choose to use the structure to always speak precisely, using terms relating in well-defined ways to what can be observed in HMS behavior. We have used these terms to state some very simple axioms or assumptions about the way human-machine systems are organized and how they function. From these axioms, it is believed that theorems addressing important HMS issues can be stated and investigated. Operationally speaking, what has been produced is a formal structure which can be used to describe existing and proposed HMS's, define important HMS issues, pose research questions, devise experiments to investigate these questions, and develop models to predict performance.

The dissertation has made a more general contribution to the area of scientific inquiry, for it has been a demonstration of just one manner in which structures to organize knowledge can be developed. It has made some claims as to the attractiveness of a formal approach to theory development and has attempted to follow that approach and justify its claims. Whether or not the theory (or theories) which may result from these foundations will enjoy any kind of success is really of secondary importance to the pedagogical contribution that an account of the process provides.

6.3 Conclusions

Although these contributions can be claimed for the dissertation, it must be admitted (but with no real regret) that the material presented in these pages is not an end in itself. It could have been more rigorous—most mathematicians would argue that it is hardly rigorous at all. It certainly could have been more thorough, for many important concepts of HMS's were not directly addressed. Also, the ideas that were presented could have been more carefully and thoroughly developed.

However, the real value of this work is not in the results produced but in the approach that produced them. Inquiry is a process of structure development. We often
claim that through inquiry we search for truth but, as we can clearly see from the history of science, our ideas about what truth is are constantly changing. This process of change can be characterized as a constant flux of conceptual structures that we build, modify, destroy, and replace to bring a sense of order to what we experience individually and collectively, for without this order, a chaos of perceptions, impressions, and ideas would result.

It is really impossible for something written on paper to faithfully capture the dynamic nature of a process. However, it is hoped that the reader has seen through this account the development of ideas starting with those of languages, models, and theories, through the notions of sets and relationships, to the concept of a human-machine system, and how those ideas can be useful in ongoing research. Of course the actual process did not necessarily follow those systematic steps, it just seemed sensible to present the work in a logical order. After all, it is often advantageous to use structures to organize our knowledge about structures as well.
LIST OF REFERENCES


