Mathematical Foundations of Manufacturing Science: Theory and Implications

by

Steven Hyung Kim

B.S., Columbia University (1977)
S.M., Massachusetts Institute of Technology (1978)
M.S., Columbia University (1980)
S.M., Massachusetts Institute of Technology (1983)

Submitted to the Sloan School of Management
in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy in
Management and Mechanical Engineering

at the

Massachusetts Institute of Technology

June 1985

Signature of Author . . . . . .
Sloan School of Management, 3 May 1985

Certified by . . . . . .
Professor Nam P. Suh, Thesis Supervisor

Accepted by . . . . . .
Professor Arnoldo C. Hax, Chairman of Interdepartmental Committee

Accepted by . . . . . .
Chairman of Doctoral Committee, Sloan School of Management

Mathematical Foundations of Manufacturing Science: Theory and Implications

by

Steven Hyung Kim

Submitted to the Sloan School of Management on
3 May 1985 in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Management and Mechanical Engineering

Abstract

A mathematical architecture for manufacturing science has been constructed by building on the foundations of mathematics. The contributions in this thesis may be outlined more specifically as follows: (1) *Conceptual.* Frameworks are presented for a systematic study of manufacturing systems as well as for a mathematical architecture for manufacturing theory. (2) *Technical.* This category consists of three types of results: (a) Algebraic structures are shown to be appropriate mathematical structures for the analysis phase. (b) Symbolic logic is used to formalize the synthesis phase. (c) Both the analysis and synthesis phases may be grounded on the foundations of mathematics, namely set theory and logic.

A conceptual framework is presented which classifies the study of manufacturing systems along three dimensions in terms of phase (analysis versus synthesis), quantification (qualitative versus quantitative), and formality (informal versus formal). A science of manufacturing, if it is to be truly useful, must address both the analysis and synthesis of manufacturing systems. Analysis deals with the study of the components of a system and their interactions, in order to determine the nature and limits of their behavior, while synthesis deals with rational principles for designing manufacturing systems. Informal theories relate to assortments of facts and ideas, while formal theories provide rigorous, coherent systems. Moreover qualitative models deal with concepts while quantitative theories incorporate equational relationships.

Starting from set-theoretic concepts, the notion of an algebraic structure is formalized, then used as a uniform base for specialized structures such as lattices, graphs and groups. These constructs are in turn used to develop an infrastructure for the study of manufacturing systems. Algebraic and set-theoretic structures are shown to be appropriate bases for system representations such as matrices, graphs, and state spaces. Moreover they provide (1) a precise symbolism for specifying qualitative systems concepts such as linkages and hierarchies, (2) a uniform framework for more specialized theories such as automata theory and control theory, and (3) a base on which to build quantitative theories.

A typology of system models is also presented, classifying models into black, grey and beige boxes. In particular, the models of automata theory are shown to fall into the class of grey boxes, while those

of control theory and information transmission are beige boxes. Models such as these provide the bases for a rigorous study of the nature of manufacturing systems. The utility of algebraic structures for the analysis phase is demonstrated by an investigation of information characteristics and their impact on manufacturing performance.

In addition symbolic logic is used to formalize the Design Axioms, a set of decision principles which were previously available only informally. The implications of such formalization are carried forward to two levels, in terms of theoretical as well as operational consequences. The results are as follows: (1) *Theoretical.* The relationships among the Axioms and their corollaries can be studied more rigorously. An unexpected result is that the propositions which had previously been considered to be corollaries of the Function and Information Axioms may, in fact, be divided into two categories consisting of direct and indirect consequences. The class of indirect consequences may be further partitioned into those which follow from the Axioms plus some weak assumptions, versus those that require strong assumptions. (2) *Practical.* The long-term goal of axiomatics research is to establish concrete decision rules and techniques to enable computer systems to design manufacturing and engineering systems. Stating the Axioms in symbolic logic makes it clear how they may be written as clauses in a logical programming language such as Prolog. Generalized structures are presented for encoding procedures and data in Prolog, then illustrated through incorporation into a program called the Computerized Axiomatic System (CAS). The operating modes of CAS are discussed, as is the overall architecture for a full-fledged expert system.

The mathematical groundwork for the analysis and synthesis phases must ultimately be unified to form a coherent basis for manufacturing science. The way in which this may be accomplished is demonstrated by taking the Information Axiom as a vehicle for discussion. The consequences of modularity and the interpretation of an automaton in the context of information minimization are also addressed.

Finally, the conceptual frameworks are shown to be useful not only for organizing the diverse constructs and theories discussed so far, but for providing directions for future research.

Thesis supervisor: Nam P. Suh

Title: Professor of Mechanical Engineering

Acknowledgments

The last two years of my doctoral program have been among the happiest and most fulfilling periods of my life. This is due in large measure to the various people with whom I have had the pleasure to associate. I am grateful to:

* My thesis advisor, Professor Nam Suh, for the opportunity to share his vision for a science of manufacturing. Through both a master's and a doctoral program, he has been a constant source of inspiration in his role as a teacher, researcher, and principled human being.

* My committee chairman, Professor Arnoldo Hax, for helping me establish a course of study and for his efforts in overseeing it.

* The other members of the committee, Professors Gabriel Bitran and Tony Wong, for their counsel in finding directions for research and for guiding me through the labyrinths of an interdepartmental program.

* Fellow researchers in the axiomatics effort, Professor Susan Finger, Professor Jang Lee, and Jack Smith, for many hours of thoughtful discussion and advice. In particular, the theorems on functional coupling (see Appendix E) were developed in collaboration with Dr. Lee during his sabbatical stay at MIT.

* My office mates, Farid Kaymaram, Kyriakos Komvopoulos, Dai Gil Lee, Dae Eun Kim, Chongnam Chu, Charlie Sweet, and Steve Chan, for a lively atmosphere and many lunches together.

* My friends Elias Towe, Cyrus Bamji, Bill Song, Costas Bardjis, and Spiros Kinnas for their participation in the Eigen Club and the sorties to Paradise.

The work reported in this thesis was supported in part by a special fellowship from Emhart Corporation and a research grant from Xerox Corporation.

Once upon a time not too long ago, a beauty pageant winner was asked, after her year-long reign as the titleholder, how she felt about the feminists' charge that beauty pageants exploit women. "If this is exploitation", she replied, "then every woman should be exploited like this."[1]

---

[1] A disclaimer: Use of this story has no bearing whatsoever on my views on feminism. As it happens, I used to cheer for feminism in believing that every woman of able mind should pursue a career. Now that I am less young, however, I subscribe to a philosophy of contingency: "You must follow your dreams to be truly free", whether such dreams lie within or without the home.

And if my experiences over the past two years were indicative, I would wish that every person could have the opportunity to be a graduate student.

To my Parents

and the memory of

my Paternal Grandfather

Table of Contents

Table of Figures

Notation

SYMBOL	DESCRIPTION
&	And.
$\vee$	Or.
~	Not.
$\exists$	There exists.
$\forall$	For all.
->	1. Implies. 2. Graphic representation of ordered pair: a -> b means <a,b> is in a given relation.
<-	1. Implied by. 2. Graphic representation of ordered pair: b <- a means a -> b.
$\leftrightarrow$	If and only if.
$\models$	Consistent (logical consequence of): $W \models$ W means that W is true in all true interpretations of W.
$\vdash$	Derivable: $W \vdash$ W means that W is derivable from the set W of well-formed formulas.
$\in$	Member or element of.
$\subseteq$	Subset of.
$\subset$	Proper subset of.
$\supseteq$	Contains.
$\supset$	Properly contains.
$\cup$	Union.
$\cap$	Intersection.
$\star$	Concatenation operator.
-	1. Subtraction operator. 2. Difference operator (of sets).
*	1. Multiplication operator. 2. Dummy variable. 3. Kleene star operator: Σ^* means set of all finite concatenations of elements in Σ.
$\leq$	1. Less than or equal to. 2. Precedes or equal to.
$<$	1. Less than. 2. Precedes.
$<_i$	Immediately precedes.
$\geq$	1. Greater than or equal to. 2. Succeeds or equal to.
$>$	1. Greater than. 2. Succeeds.
$>_i$	Immediately succeeds.

$\sqsubset$	Part of.
$\rhd$	Acts on.
$\lhd$	Acted upon.
$\prec$	Subsystem of.
$+_a$	Association operator.
$\lvert * \rvert$	1. Cardinality of *. 2. Absolute value of *.
	3. Matrix of elements given by *.
∂/∂	Partial derivative operator.

$::=$	Defined as.
$\langle *,* \rangle$	Ordered pair.
$\langle *,*,\ldots,* \rangle$	Ordered list, n-tuple or vector.
$(*)$	Delimiters for *.
$[*]$	Delimiters for *.
$\{*\}$	1. Delimiters for *. 2. Set of objects *.
$\{*\lvert C\}$	Set of objects * such that condition C holds.

β	Behavior.
δ	Relative deficiency (of information).
Δ	Difference in: ΔX means difference in X.
η	Efficiency.
μ	Manufacturing system.
π	Process.
Π	1. Product operator. 2. Partition.
σ	1. Element of Σ. 2. System.
σ_0	Null input (of automaton).
$\sigma\lvert_k$	Object obtained from σ when only links of kind k are considered.
Σ	1. Set of systems. 2. Set of input symbols (of automaton).
	3. Summation operator.
$\Sigma\lvert_k$	Set of all objects $\sigma\lvert_k$ of type k.
τ	Subset of time instants T.
ω	Element of Ω.
ω_0	Null output (of automaton).
Ω	1. Set of all objects. 2. Set of output symbols (of automaton).

B	(Structural) bondage.
C	Coupling matrix.
Co	Composition.
D	1. Domain. 2. Data.
DP	Design parameter.
e	Event in state space.
E	1. Edge set (in graph). 2. Effectiveness.
E_p	Permissible event space.

En	Environment.
F	Set of final states (in automaton).
$\mathcal{F}$	Set of feasible solutions.
FR	Functional requirement.
h	History (of state space).
h(y|x)	History of y under influence of x.
I	Information measure.
In	Input.
k	Element of K.
K	1. Set of kinds of bonds. 2. Knowledge.
lb	Binary logarithm: log to base 2.
Lb	Logical bondage.
M	Input transition function.
$\mathcal{M}$	Set of nxn matrices.
N	Output transition function.
$\mathcal{N}$	Set of natural numbers.
Out	Output.
p	Probability.
P	1. Property. 2. Performance index.
R	1. Relation. 2. Set. 3. Redundancy. 4. Reangularity.
R	Set of relations.
$\mathcal{R}$	Set of real numbers.
s	1. State. 2. Element in set S.
s_0	Initial state.
S	1. Set (in general). 2. Set of states. 3. Set of social relations. 4. Semangularity. 5. Segmentation (of interval).
S_c	Conceivable state space.
S_p	Permissible state space.
St	Structure.
t	1. Element of set S. 2. Instant of time.
T	1. Set of time instants. 2. Set of transformation relations.
V	Vertex set (in graph).
x	1. Variable. 2. Object. 3. Cartesian product operator.
X	1. Variable. 2. State function.
y	1. Variable. 2. Object.
Y	Variable.

Chapter One

Introduction

1.1 General Purpose

A science of manufacturing must be based on general principles concerning elementary structures and their interrelationships. Moreover these concepts must be describable in a form concrete enough for mathematical expression. In fact, a coherent mathematical foundation on which to build specialized subtheories may well be more important in the long run than an assortment of disjointed facts.

The field of macroeconomics recognizes the fact that an inappropriate sequence of development in the economic and technological arenas of a society can lead to disruptive problems at a later stage. The area known as *appropriate technologies for developing countries*, for example, has come into existence as a direct recognition of this phenomenon. In particular, a developing economy must possess infrastructures such as utilities, roadways and telecommunications before it can properly make use of advanced technology. A washing machine, for example, is of little use in a society composed of hamlets lacking plumbing and electrification.

In a similar way, if the field of manufacturing is to become a science, it is imperative to approach this area systematically. We must have a foundation upon which to build more elaborate structures, so that *manufacturing science* will become a synergistic system of useful results rather than a rabble of facts and opinions.

Models and paradigms are central to the development of scientific endeavors. To quote one influential philosopher of science [Kuhn 1970: 15-19]:

> In the absence of a paradigm ... fact-gathering is a far more nearly random activity than the one that subsequent scientific development makes familiar .... When the individual scientist can take a paradigm for granted, he need no longer, in his major works, attempt to build his field anew.

The current work is based on the additional premise that mathematical models are central to the development of a science of manufacturing. This presupposes the belief that a phenomenon which is sufficiently well-understood can be described in mathematical terms [Kneebone 1963: 332]:

> There are indeed no *a priori* restrictions whatever on the kind of 'experience' to which mathematics may properly be applied, and any domain in which entities are sufficiently well defined and relations sufficiently stable for a clear structural pattern to be discernible is a legitimate field of applications for mathematics.

In particular, set theory and symbolic logic are proposed to be appropriate mathematical foundations for a science of manufacturing.

To summarize, this dissertation is based on the following premises:

1. Quantum advances in industrial productivity can only result from technological and organizational innovation [cf. Appendix A].

2. Systematic innovation requires a scientific theory for design and manufacturing.

3. A scientific theory must be grounded on a mathematical foundation.

4. The foundations of mathematics--namely, set theory and symbolic logic--must ultimately serve as the mathematical bases for manufacturing science.

1.2 Previous Research

The analytic phase refers to statements about the structure and behavior of manufacturing systems, while the synthetic phase refers to decision rules for designing superior systems. A set of general decision rules for the synthesis phase has been set forth in the form of manufacturing axiomatics [Suh et al. 1978a]. These principles have been elaborated on in the intervening years [Rinderle and Suh 1982; Nakazawa and Suh 1984; Wilson 1981]. To date, however, the design principles have been available only in informal form.

In the matter of providing mathematical formulations, the literature has been less productive. The limitations of the analytic approach as the mathematical basis for a science of manufacturing has

prompted some interest in an algebraic approach [Bjorke 1973]. However, no systematic research has been conducted in these areas. The following subsections pursue these ideas further.

1.2.1 Analysis

Traditional engineering fields such as mechanics, electronics, fluids, communications, transport phenomena, and control theory are often amenable to description by linear, continuous models. Hence analytic tools such as calculus, differential equations, and linear algebra are eminently suited for this purpose.

For example, the analytic approach is appropriate for describing the algorithm for the economic order quantity in inventory control. Moreover continuous models can be useful for describing the kinematics and dynamics of physical systems, such as the description of a lathe cutting tool under closed-loop control.

Studies in systems and control theory have relied on the application of vector space theory to the analysis and synthesis problems in a large spectrum of estimation and control applications. This approach has yielded exact or approximate solutions for some linear and nonlinear problems, such as local controllability conditions for nonlinear systems [Willsky 1973: 18]. However, the analytic methods are ineffectual for most nonlinear problems.

In the field of manufacturing, continuous models are applicable only to specialized phenomena comprising a small part of the entire manufacturing process. The traditional mathematical tools are of little use for describing the full spectrum of manufacturing phenomena including discrete, nonlinear processes. Hence new mathematical tools and models must be sought.

Moreover, continuous models are inapplicable to the description of software or hybrid systems. For example, the changes in state of a transaction log program cannot be described by differential equations. As more and more subsystems come to rely on digital processing units, this problem will become even more acute in the future.

1.2.2 Synthesis

In contrast to the relative plenitude of tools and techniques applicable to the analysis phase, the synthesis phase is characterized by want. Such disparity springs from the elusive character of the

creative effort: it is difficult to determine the general principles involved in design, or even to pinpoint the steps involved.

This problem has been addressed by a set of decision rules for design [Suh et al. 1978a]. The two design principles may be stated informally as

Axiom 1 (Independence). Maintain the independence of functional requirements.

Axiom 2 (Information). Minimize information subject to the fulfillment of functional requirements.

These Axioms also give rise to a number of associated corollaries.

The axiomatic methodology is based on the premise that the proper design of products and processes has the greatest impact on manufacturing productivity. To the extent that design axiomatics simplifies and rationalizes the process of engineering design (the cost of which represents an average of 15% of a product's selling price) and manufacturing (40%), it offers the potential for effecting significant reductions in the overall sales price [See Appendix A].

To date, research in axiomatics has yielded two classes of results:

1. *Qualitative.*

   * From the initial set of proposed axioms, two have been identified as the key principles.

   * The usefulness of the axiomatic approach [Rinderle and Suh 1982; Suh et al. 1978a; Suh and Rinderle 1982] has been demonstrated in various manufacturing fields ranging from producing lenses to processing polymers. An example is the molding of the thermal protection system on the external tanks of the space shuttle [McCree 1983]. The development of a simple vented compression molding technique saved NASA about $10 to $40 million, of which 60% was due to reduced waste materials and the remainder to lower labor and capital requirements.

   * The concept of functional hierarchies has proved useful for applying Axiom 1 [Yasuhara 1980].

* The applicability of the information axiom to process planning has been demonstrated [Nakazawa and Suh 1984].

2. *Quantitative.* Metrics for functional independence [Rinderle and Suh 1982] and information content [Suh et al. 1978b; Wilson 1981] have been proposed. These metrics are discussed in greater detail in Appendices C and D.

However, this theory has to date been available only as an informal set of decision rules. A major result of the work reported in this thesis is the demonstration of symbolic logic as a formal construct for the synthesis phase of manufacturing.

The long-term objectives of the axiomatic research effort are to develop a science base for design and to develop intelligent software systems. Despite the success of computer systems in handling ancillary tasks such as graphics generation, number crunching, and database management, computers are currently unable to support primary decision making tasks related to design. By incorporating the axiomatic principles, however, it should be possible to enable the computer to support primary functions such as determining whether certain designs meet their functional requirements, or to what extent they do not.

1.3 Conceptual Framework for Manufacturing Science

A conceptual framework for manufacturing is essential not only to house the final architecture of specialized theories, but to serve as a reference frame for guiding research during the developmental stages. A theory of manufacturing and design may be categorized along 3 dimensions, each with 2 nominal categories:

1. Phase
    a. Analysis
    b. Synthesis
2. Formality
    a. Informal

b. Formal

3. Quantification

a. Qualitative

b. Quantitative

The *analysis* of manufacturing systems refers to the study of the way components behave in isolation and in concert, and of the investigation of the limits of their behavior. *Design*, on the other hand, refers to the way in which systems should be generated or created. Obviously these terms are convenient distinctions made for the sake of discussion and research: it is difficult to conceive of a good design procedure, for example, that does not include some facets of analysis.

When a candidate solution to a set of functional requirements is generated, it must be analyzed to ensure consistency with the original requirements. Hence any methodology which enhances the analysis phase also tends to improve the synthesis effort. This relationship is depicted in Figure 1-1.

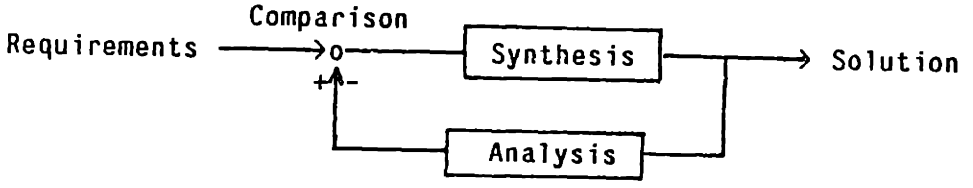


Figure 1-1:Synthesis and analysis phases of developing solutions to requirements.

For both the analysis and synthesis phases, we may have *informal* or conceptual models as well as *formal* or mathematically rigorous theories. The latter type deal with the same objects as the former, but is distinguished by the fact that it superimposes mathematical constructs on the conceptual models.

Formal theories may be further subdivided into those relating to general principles or to quantitative metrics for certain attributes. For example, symbolic logic may be used as a formal description of the manufacturing design principles embodied in the form of the Axioms [Suh et al. 1978a], while the metrics for information [Suh et al. 1978b] and functional independence [Rinderle and Suh 1982] are measures of the two key attributes.

A *qualitative* theory deals with concepts and general principles relating to a given field. It is prerequisite to a *quantitative* theory which involves the construction of analytic models. The current state of art in manufacturing has available isolated quantitative models such as robot kinematics, but lacks a systematic methodology which unifies either the qualitative or quantitative subtheories for analysis.

These 3 dimensions may be viewed as a set of axes, each with two classifications, as shown in the figure below.

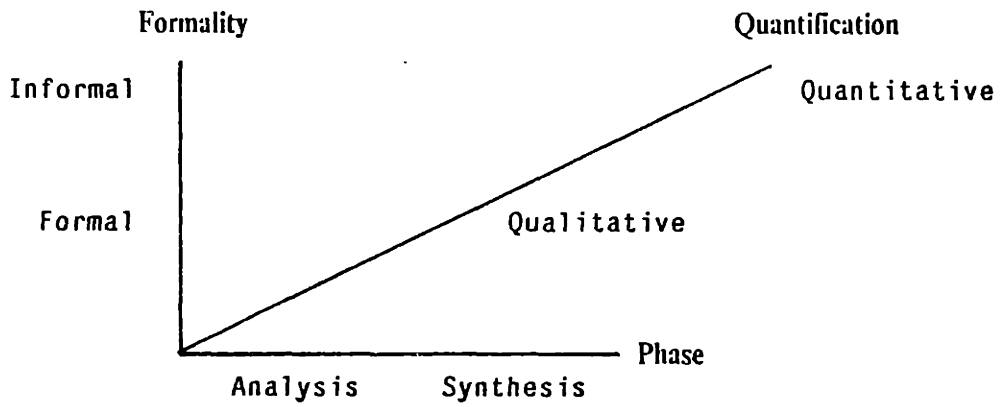


Figure 1-2:General conceptual framework for manufacturing science.

These classes partition the 3-dimensional volume into 8 regions.

It is obviously cumbersome to display a 3-dimensional object on a 2-dimensional page. Hence for the purposes of discussion, we will consider the qualitative and quantitative aspects separately as subframeworks of the general conceptual framework.

1.3.1 Qualitative Aspect

Qualitative theories may be of the formal or informal variety. Both varieties, of course, deal with the same subject matter and differ only in the manner of presentation. The figure below shows the subframework for the qualitative aspect. The second row of entries pertain to mathematical constructs which may be used to formalize the concepts in the first row.

The automated factory will be largely coordinated by computers and run by robots. The computational hardware will take the form of a hierarchy of mainframes, minis and micros, including

	PHASE	
FORMALITY	Analysis	Synthesis
Theories	Conceptual extensions of automata, computability, & systems theory	Design Axioms and corollaries
Mathematical constructs	Algebraic structures	Symbolic logic

Figure 1-3:Subframework for qualitative aspect.

the processors incorporated in robotic sensing devices. The software hierarchy will consist of algorithmic procedures, such as those for data collection and inventory control, as well as knowledge-based systems for such tasks as situation recognition and crisis management.

In this milieu, the natural mathematical models would seem to be those of automata theory and computability theory. Automata theory [Ginzberg 1968] and computability theory [Davis 1958; Lewis and Papadimitriou 1981] have yielded a number of remarkable results concerning the nature and limits of computations. An example lies in the unsolvability of the halting problem.

A different perspective is found in general systems theory, which provides a framework of concepts and terminology for the study of all types of systems [Bunge 1979; Gheorge 1981; Mesarovic 1970, 1975; M'Pherson 1973]. The power of systems theory lies in the applicability of its methods and conclusions to all systems. At the same time, its weakness lies in its blindness to domain-specific knowledge; but a key lesson from artificial intelligence is that knowlege is the source of logical power. Hence systems theory by itself is obviously inadequate for a science of manufacturing.

But here lie the building blocks for a new edifice. By adapting automata theory to the manufacturing context and by tailoring systems theory, it should be possible to construct a coherent framework on which to build a theory of manufacturing systems. This objective is pursued at length in Chapter 3.

In summary, a set of concepts and principles for the synthesis phase has been proposed in the form of the Design Axioms. The models and concepts for the analysis phase may be adapted from existing fields such as automata theory and system theory, then extended to fit the manufacturing environment. A basic premise of the current work is that logical and algebraic structures are

appropriate formal constructs for presenting these ideas rigorously.

1.3.2 Quantitative Aspect

Much of the quantitative research in manufacturing has been in terms of theories for the analysis phase. Having been developed by different investigators at different times, these theories are informal in the sense that they are not based on a unifying mathematical substructure.

	PHASE	
FORMALITY	Analysis	Synthesis
Theories	Special subtheories; Data on individual systems; Standard performance metrics	Data on sets of designs
Mathematical constructs	Algebraic structures; Measurement theory	Symbolic logic; Functional independence & information metrics

Figure 1-4:Subframework for quantitative aspect.

It is difficult to envision a formal quantitative methodology, such as calculus, which would integrate the various subtheories for the analysis of manufacturing systems. Certain concepts, such as the emergence of instability in a system composed of stable subsystems, are better handled by qualitative constructs. In fact, a conclusion of the current work is that algebraic structures can provide the formal infrastructure on which to base the specialized subtheories.

1.3.3 Discussion on Quantification

The traditional wisdom in the physical sciences maintains that quantification is the only road to enlightenment. This viewpoint is certainly appropriate for such areas as physics, where the emphasis is on breaking down systems into smaller pieces and modeling their behavior.

In many newer fields such as computer science, molecular biology and engineering, analysis is only part of the game. The other part relates to synthesis, the construction of systems superior to--or

simply different from--their precedents. This is especially true of the field of manufacturing.

Hence it is necessary to generalize the old adage that "numbers make the theory good" to the newer formula, "rigor makes the theory good". And rigor may be found in both qualitative and quantitative attire.

Hence formal mathematical structures are appropriate for both the analysis and synthesis phases, as well as qualitative and quantitative aspects. Moreover, a rigorous qualitative infrastructure is a logical prerequisite to the development of a unified quantitative theory.

1.3.4 Architecture for the House of Manufacturing Science

The relationships among the various concepts discussed so far are depicted in Figure 1-5. The rectangles in the figure imply a partial ordering among the ideas, from bottom to top, in terms of logical precedence. For example, algebraic structures are a logical prerequisite to the mathematical representations of systems.

The justification for this figure will follow from a more detailed examination of the various topics. The purpose of this thesis is to investigate mathmatical constructs as they relate to the foundations, infrastrucuture, and general theories. The category of "special theories" will be left for future research on the particular topics involved.

1.4 Scope of Thesis

This chapter has argued for the need for a mathematical infrastructure for a science of manufacturing. A conceptual framework has been presented, based on a 3-dimensional architecture spanned by formality (informal vs. formal), phase (analysis vs. synthesis), and quantification (qualitative vs. quantitative). This framework serves as a general structure in which to house different facets of manufacturing science, as well as to provide guidelines for present and future research.

Chapter 2 presents the mathematical foundations on which subsequent chapters build. Symbolic logic is the paradigm of rational thinking while set theory constitutes the primitive objects of mathematical reasoning. In particular, set theory provides the background for an explicit definition of an algebraic structure. The latter in turn is shown to be the basis for a spectrum of basic

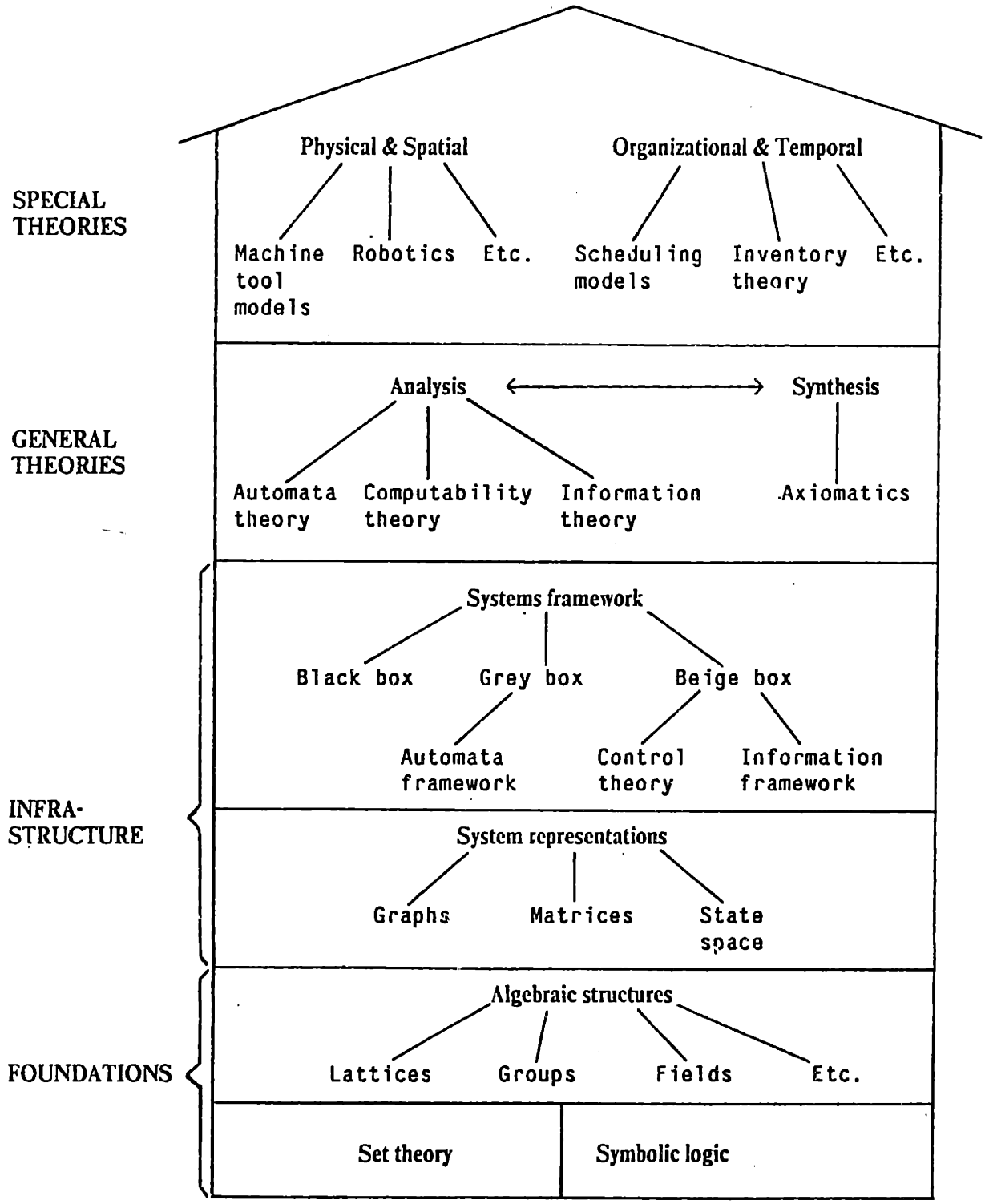


Figure 1-5: Architecture for the House of Manufacturing Science.

mathematical constructs such as lattices, groups and fields.

Chapter 3 demonstrates how the set-theoretic and algebraic structures from the preceding chapter may be used to (1) express with precision a variety of important qualitative concepts such as hierarchies, (2) provide a uniform framework for more specialized theories such as automata theory and control theory, and (3) provide the groundwork for quantitative theories. This is achieved by presenting systems concepts in set-theoretic language and by using mathematical structures as the basis for system representations and models.

The infrastructure of the preceding chapters is used in Chapter 4 to develop a precise qualitative model of manufacturing systems. The utility of these structures is demonstrated by applying them to an investigation of information characteristics and their impact on manufacturing performance.

Chapter 5 shows how symbolic logic may be used to give rigorous expression to the design principles of manufacturing axiomatics. The relationships among the Axioms and the so-called corollaries are clarified; a number of propositions are seen to be derivatives of the Axioms and some mild assumptions, while others would require a major extension of the set of Axioms. The application of symbolic logic also makes it clear how the Axioms may be encoded in a logical programming language such as Prolog. A system architecture for an expert system for design axiomatics is also presented here.

Chapter 6 deals with the integration of the analysis and synthesis phases. The Information Minimization Axiom is used as a vehicle to demonstrate how this integration may be achieved.

Chapter 7 concludes the thesis by summarizing the results of the previous chapters (in somewhat greater detail than the discussion here). In addition, the conceptual framework of Chapter 1 is invoked to explore directions for future research.

The precedence ordering among the chapters and their subjects is depicted in Figure 1-6.

A number of topics related to the main current of the thesis are discussed in the appendices. Appendix A presents a short introduction to the sources of industrial productivity and their implications for research in manufacturing. It asserts that increases in labor efficiency have only limited potential for enhancing industrial productivity, and that significant progress can ensue only from technological and organizational innovation. Appendix B contrasts and compares the axiomatic

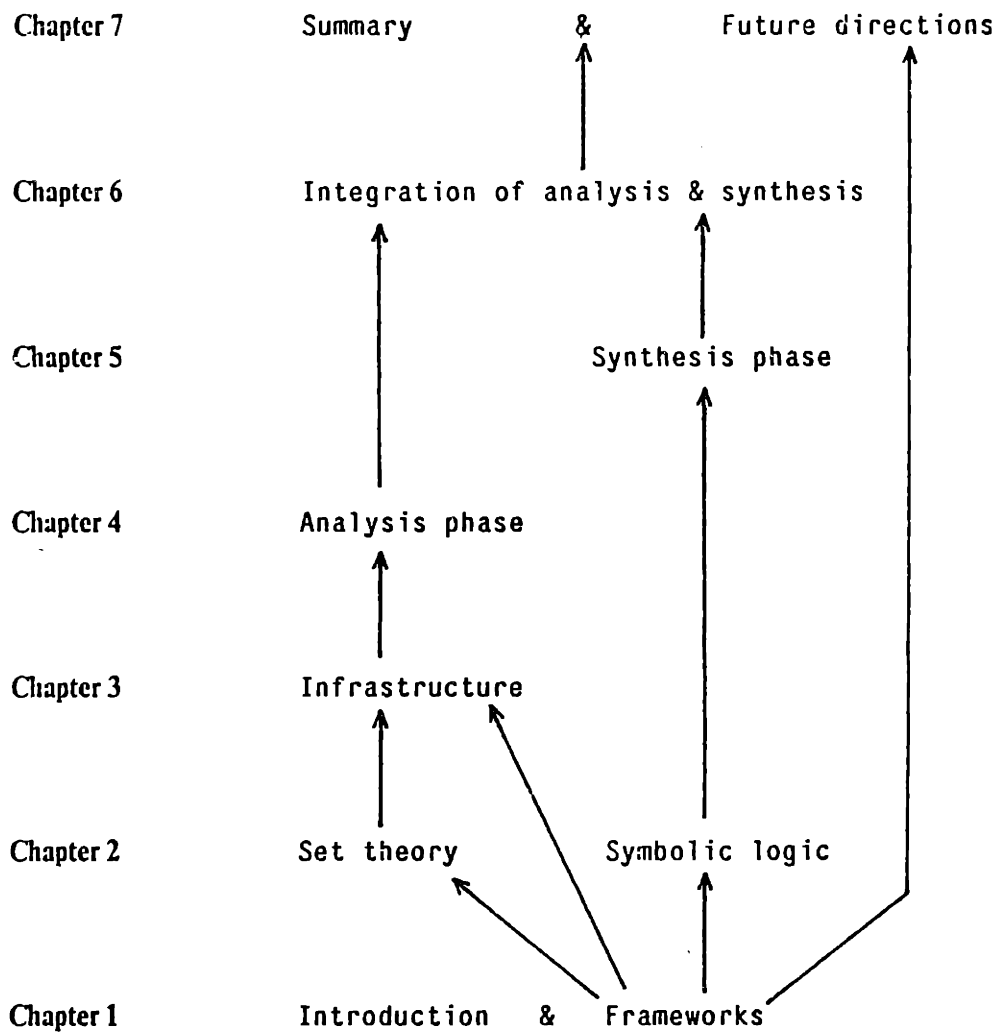


Figure 1-6:Precedence ordering among the chapters and their subjects.

methodology with the satisficing and behavioral models of decision making.

The subsequent attachments are somewhat more specific and technical. Appendix C presents background material on measures of functional coupling in terms of reangularity and semangularity. Appendix D presents an informal introduction to the traditional measure of information, followed by a more formal treatment. The relationship of information to entropy is clarified, and the application of the information measure to tolerance specifications is discussed.

Appendices C and D may be helpful in reading Chapters 3, 4 and 5. They are also instrumental for Appendix E, which presents a number of theorems relating to the Function and Information Axioms. The modularity and invariance properties of the independence measures are proved, and the relationship between information and coupling is clarified. Finally, Appendix F offers background information on the logical programming language of Prolog which is discussed in Chapter 5.

Chapter Two

Mathematical Foundations

This chapter presents the foundations of mathematics, symbolic logic and set theory. The concepts of set theory are then used to give a precise definition of an algebraic structure, which in turn is used as the basis for more specialized constructs such as graphs, lattices, groups and fields. While these particular concepts are widely available in the literature, the unified presentation given here is believed to be novel.

Strictly speaking, symbolic logic should be used to present set theory and algebraic structures formally. For example, Subsection 2.1.2.3 illustrates how group theory may be expressed in symbolic logic as axioms G1 and G2. The main objective of this chapter, however, is on explanation rather than formalization. Therefore in the interest of readability, the presentation tends to be informal.

The material in this chapter forms the basis for the infrastructure discussed in Chapter 3, which in turn is the supporting structure for Chapter 4. Moreover symbolic logic in conjunction with the principles of design axiomatics are the bases for Chapter 5.

2.1 Symbolic Logic

People use logic as a matter of course in everyday life. For example, a person who knows that a falling barometer level indicates the approach of foul weather, will draw the right conclusion from observing a slide in atmospheric pressure.

Unfortunately we humans are prone to error when it comes to complex problems in reasoning. For example, careful studies of legal documents--which are intended to be paragons of logical precision--often result in the excavation of internal contradictions.

Whether by design or by inculcation, we are often careless in stating assumptions explicitly and sloppy in performing deductions. Although our long-term memory capacity seems to be infinite in practical terms, our short-term working memory can handle only about 7 items concurrently. Further, although our minds are capable of occasional leaps of intuition (which may or may not turn

out to be justified), they are remarkably slow in making even vanilla-flavored deductions.

Mathematical logic was developed to address the problems of faulty reasoning and ambiguity in everyday language. The basic premise is that correct procedures for reasoning or deductive thinking can be formalized and used profitably, independently of the specific domain of discourse.

This section first gives an informal introduction to the notation and terminology of symbolic logic [Bittinger 1982; Hamilton 1978; Kneebone 1963; Kowalski 1979]. Among all fields of study, it is only appropriate that mathematical logic be presented formally: this is the second item on the immediate agenda.

2.1.1 Informal Introduction

2.1.1.1 Propositional Logic

A *proposition* is a declarative statement or sentence. A particular proposition may be either *true* or *false*. To illustrate, let H denote the statement "Horses have wings", and D the statement "Ducks can swim". Then H is a false proposition while D is a true proposition.

There is little that we can do with an assortment of unrelated propositions except to assign truth values (i.e. *true* or *false*). To enable us to say more interesting things, we introduce the notion of functors and connectors; these concepts are defined and illustrated below.

A *functor* is a function which assigns a new truth value to one or more propositions. The simplest example of a functor is *not*, also called the *non* functor and denoted by the symbol ~. Given a proposition P, we take ~P (or "not P") as true if P is false, and false if P is true.

A *connector* is a functor which makes a compound statement out of two simpler statements. The *and* connector, also called *conjunction* and symbolized as &, results in the compound statement being true if each of the referent statements is true, and is false otherwise. Using the example from above, the compound statement (H & D) is false, since H is false.

The *or* connector, also called *disjunction* and symbolized as V, results in the value *true* of at least one of its substatements is true, and is false if both are false. From the preceding example, (H V D) is true, since D is true.

The *implication* connector is also called *if-then* and is denoted by a right arrow, ->. The compound statement (P -> Q), read "P implies Q", is false if P is true and Q is false, and is true in all other cases. In the example above, (H -> D) is true since H is false, while (D -> H) is false since D is true and H false. The left-hand-side P is called the *antecedent* or *premise* while the right-hand-side Q is the *consequent* or *conclusion.*

The connector *implied-by*, also called the *if* connector and symbolized as a left arrow <-, is the converse of implication. In other words Q <- P is true if and only if P -> Q is true.

The *equivalence* connector, also called *if-and-only-if* or simply *iff*, is symbolized as ↔. P ↔ Q is defined to be true if P -> Q and Q -> P are both true, and is false otherwise.

As a matter of convenience and convention, we observe the following precedence ordering among functors and connectors, in increasing scope of influence:

1. ~

2. &

3. V

4. ->, <-, ↔

For example, the sentence (P & ~Q -> R) is taken to mean [(P & (~Q)) -> R]. Of course the precedence ordering may be superceded by the explicit use of delimiters such as parentheses.

A *truth table* is a table of combinations of truth assignments to propositions, whether of the elementary or compound variety. An example is given in the figure below.

P	Q	(P -> Q)	~P	(~P V Q)
t	t	t	f	t
t	f	f	f	f
f	t	t	t	t
f	f	t	t	t

Figure 2-1:Example of a truth table.

The third line, for example, says that when P is false and Q is true, then the sentences P -> Q, ~P, and ~P ∨ Q are all true. A comparison of the third and fifth columns shows that they are the same. Since P -> Q and ~P ∨ Q take on identical truth values for all combinations of values for P and for Q, we conclude that they are equivalent; that is, (P -> Q) ↔ (~P ∨ Q).

This observation highlights the fact that the functors and connectors are not independent of each other. For example, the functors ->, <-, and ↔ may each be defined in terms of ~, ∨, and &.

2.1.1.2 Predicate Logic

The calculus of propositions is useful but limited in scope. For example, there is no way to say "If an object x is human, then x is mortal" for any arbitrary object x. So we generalize the concept of a proposition to include one or more arguments.

A *predicate* is a function that assigns a truth value to some set of objects which serve as its arguments. To illustrate, let H(x) be the predicate that takes on the value *true* if x is human, and is *false* otherwise. In a similar way, we let M(x) define the predicate "mortal". Then the statement "If x is human, then it is mortal" may be written in predicate logic as H(x) -> M(x).

We would also like the capability to say whether a given predicate is true for *some* object or *all* objects. This is implemented through the use of quantifiers.

The *existential quantifier*, symbolized as ∃, has the following interpretation: (∃xP(x)) is taken to be true if there is some object x which makes the predicate P(*) true, and is false otherwise. For example, let M'(x) denote the predicate "x is a man", W(x) the predicate "x is a woman", and L(x,y) the predicate "x loves y". Then ∃x M'(x) is true iff there is an object which is a man. Moreover, the statement ∃x∃y(M'(x) & W(y) & L(y,x)) says "There is a man who is loved by some woman".

The *universal quantifier*, symbolized as ∀, has the following interpretation: ∀xP(x) is true iff P(x) is true for every object x. For example, ∃x∀y(M'(x) & W(y) & L(x,y)) says that there is a man who loves all women.

As in the case of propositional calculus, quantifiers may be defined in terms of each other plus the ~ functor. Thus ∃xP(x) is logically identical to ~∀(x)~P(x). For example, "There exists some bird" is equivalent to "It is not the case that for all objects, each is not a bird".

This brings to an end the informal orientation to symbolic logic, and we turn now to a more rigorous presentation.

2.1.2 Formal Presentation

A logical system can be defined by two equivalent approaches, the *syntactic* view and the *semantic* view. The former regards axioms and theorems in terms of their sentence structure, while the latter focuses on the meaning of the sentences. Both viewpoints are based on the definition of a language: a set of symbols and of rules for constructing well-formed formulas. The most common language used is that of *first-order predicate calculus*, in which quantification occurs over variables. This is in contrast to *second-order predicate calculus*, in which the predicates may themselves be quantified.

2.1.2.1 Syntax of the Language

The language consists of primitive symbols and rules for combining them into larger objects called *well-formed formulas*. The primitive symbols are defined as follows.

Definition. A *primitive symbol* is one of the following:

1. A constant symbol, usually denoted by a letter from the beginning of the alphabet.

2. A variable symbol, usually denoted by a letter from the end of the alphabet.

3. A function symbol, such as f, g, or h.

4. A predicate symbol, usually denoted by an upper case letter.

5. One of two logical functors: ~ denoting *negation* and -> denoting *implication*.

6. The universal quantifier, symbolized by ∀ and read as "for all".

The statements of the language concern objects called terms.

Definition. A *term* is one of the following:

1. A constant.

2. A variable.

3. A function of the form $f(t_1,\ldots,t_n)$ where f is a function symbol of n arguments and $t_1,\ldots,t_n$, are terms.

No other objects are terms.

Terms serve as the arguments to atomic formulas.

Definition. An *atomic formula* is an object of the form $P(t_1,\ldots,t_n)$, where P is a predicate symbol which takes n arguments (for $n \geq 0$) and $t_1,\ldots,t_n$, are terms.

When the number and identity of the arguments are immaterial, an atomic formula $P(t_1,\ldots,t_n)$ will be written simply as P.

Definition. A *proposition* P is an atomic formula $P(t_1,\ldots,t_n)$ for which $n = 0$.

A *literal* is an atomic formula, $P(t_1,\ldots,t_n)$, or its negation, $\sim P(t_1,\ldots,t_n)$.

Well-formed formulas are built up from atomic formulas, parentheses, logical connectives, and the universal quantifier.

Definition. A *well-formed formula*, or *wff*, is one of the following:

1. An atomic formula.

2. ~P, where P is a wff.

3. P -> Q, where P and Q are wffs.

4. (∀x)P, where P is a wff and x is a single variable.

5. (P), where P is a wff.

The only wffs are those constructible by finitely many applications of rules 1 through 5.

From the primitive connectives and the universal quantifier, we may define abbreviations as follows:

1. *Conjunction*: P & Q denotes ~(P -> ~Q).

2. *Disjunction*: P ∨ Q denotes (~P -> Q).

3. *Equivalence*: P ↔ Q denotes (~P ∨ Q)&(P ∨ ~Q).

4. *Existential quantifier*: ∃xP denotes ~(∀x ~P).

We also need the idea of variables being free or not in a given expression (wff).

> **Definition.** A *closed* wff is one in which all variables are quantified; that is, no variable is free. A wff which is not closed is called *open*.

Much of the work in mathematical logic deals with closed wffs. To prove that a formula P is a theorem in predicate logic, it is sufficient to show that its universal closure, cl(P)--which denotes P universally quantified on all its free variables--is also a theorem.[1] In propositional logic every wff may be considered closed, since there are no free variables.

2.1.2.2 Semantics of the Language

In the semantic approach to logic, each expression or formula is viewed to be true or false. Since variables are not allowed in propositional calculus, an interpretation is simply an assignment of truth values to atomic formulas. These truth values may then be used to determine the truth values of more involved wffs.

The definition of interpretation must be modified for first-order predicate calculus, which does allow for variables. This may be done in terms of a domain and an assignment of truth values to the symbols in a wff, whether they represent constants, functions, or predicates.

> **Definition.** An *interpretation* of a wff consists of a nonempty domain D and an assignment of "values" to each constant or function symbol in the wff in the following way:
>
> 1. Each constant symbol is assigned an element in D.
>
> 2. Each n-place function symbol is assigned a mapping from D^n to D, where D^n is

[1] The situation is different in logic programming. There, queries are submitted in the form of open wffs in which free variables serve as "hooks" for answers by being instantiated to specific constants.

defined as the nth order Cartesian product of D [cf. Subsection 2.2]. In other words,

$$D^n = \{\langle x_1,\ldots,x_n\rangle \mid x_i \in D \text{ for } i = 1,\ldots,n\}$$

3. Each predicate symbol is assigned a mapping from D^n to $\{t,f\}$, where t represents *true* and f represents *false*.

Thus a predicate symbol is a special kind of function symbol whose codomain consists of exactly two objects, t and f.

An open wff, by definition, contains free variables. Hence such a wff may be true or false depending on the specific assignments of constants to variables. To obtain categorical or absolute results rather than conditional ones, we work with closed wffs in the semantic approach to logic. In this way a wff is either true or false in a given interpretation, regardless of the values assigned to its variables.

Definition.

1. An interpretation of a set $W = \{W_i\}$ of wffs is called a *model* of W iff every W_i is true in that interpretation.

2. A wff W is a *logical consequence* of a set of wffs W iff W is true in all models of W. In this case, we write

   $W \models W$

3. A wff W is *satisfiable* if it has a model; otherwise it is *unsatisfiable.*

4. A wff W is *valid* if it is true in all possible interpretations; otherwise it is *invalid.*

Thus if a wff is true in some interpretations but not all, it is satisfiable but invalid.

2.1.2.3 Production Rules in Syntactic Viewpoint

Certain formulas are necessarily valid due to the meaning of the logical symbols, as exemplified by the expression $x \leftrightarrow x$. Formulas such as these are called *logical axioms.* Other axioms which are valid by hypothesis are called *proper* or *nonlogical axioms.*

To illustrate, we present the modern axiom system for groups, called the elementary theory of groups

and designated G. G has exactly one nonlogical symbol, the binary functor or operator *, and two nonlogical axioms:

G1. $\forall x \forall y \forall z\, [(x*y)*z = x*(y*z)]$
G2. $\exists x\, [\forall y\, (x*y = y)\ \&\ \forall y \exists z\, (y*z = x)]$

The first axiom specifies the associative property, while the second axiom stipulates the existence of an identity element (x) as well as all inverse elements (z).

Predicate calculus consists of logical axioms and two production rules. These rules of inference are

1. *Modus Ponens*: From P and (P -> Q), infer Q.

2. *Generalization*: From P, infer $(\forall x)P$.

Inference rules may be used to deduce theorems from axioms. A first-order theory *T* is a formal system consisting of the logical and nonlogical axioms as well as the production rules.

When a theorem W is deducible from the axioms of a theory *T*, we write

$$\vdash_T W$$

A wff W is a consequence of a set of wffs *W* if W is derivable from the axioms of theory *T* and formulas in *W*. In this case we write

$$W \vdash_T W$$

A model of *T* is by definition an interpretation in which all axioms of *T* are true. A nonobvious property of a set of axioms *T* is that a theorem derivable from *T* will be true in every model of that set.

2.1.2.4 Soundness versus Completeness

The semantic approach focuses on the validity of a formula, while the syntactic approach deals with the construction of proofs of formulas. These viewpoints, however, differ only superficially, as they can be shown to be equivalent. This equivalence is known as the completeness and soundness results.

$$(W \models W) \quad \underset{\text{Soundness}}{\overset{\text{Completeness}}{\rightleftarrows}} \quad (W \vdash W)$$

Completeness refers to the fact that if a formula W is true in all models of the set of axioms *W*, then W can be proved from *W* viewed as a set of hypotheses. On the other hand, *soundness* pertains to the

fact that if the formula W is derivable from a set of axioms *W*, then W is true in all models of *W*.

2.1.2.5 Theorem Proving

One can ask whether a given wff is a theorem of a particular theory. To this end, one could apply syntactic deduction rules (Modus Ponens and Generalization) to axioms in all possible ways. If the wff is in fact a theorem, or if the number of possible deductions is finite, then the procedure will eventually terminate. But if the wff is not a theorem, then the procedure may run forever. Hence this problem is said to be *semi-decidable.*

The soundness and completeness results, however, imply the following. A wff W is a theorem of a given set of axioms *T* iff the union of ~W and *T* is unsatisfiable. In other words, if ~W and *T* derive the empty formula denoted by □, then they are unsatisfiable; and the original formula W is in fact a theorem of *T*.

A procedure in which the empty formula is derived from ~W and the axioms *T*, is called a *refutation* procedure. The most widely-known refutation procedure is based on the *resolution principle* [Robinson 1965]. This procedure is applied after the wffs are put into a canonical format called the *disjunctive normal form*, or *clause form.* In this format, each factor--consisting of a literal or collection of literals bundled together by conjunctions--is separated from other factors by disjunctions. Moreover, all variables in the expression are considered to be universally quantified.

For propositional calculus, the resolution principle takes the form of a simple cancellation law. If P, Q, and R are literals, then the pair of clauses {P ∨ Q, ~Q ∨ R} together derive the result P ∨ R:

$$\frac{\begin{array}{l} P \vee Q \\ \sim Q \vee R \end{array}}{P \vee R}$$

The literals P, Q, ~Q, and R are called *resolved literals*, while the resulting clause P ∨ R is called the *resolvent.*

To illustrate further, consider the set of axioms *T* = {P ∨ Q, ~Q}. To see whether P is a theorem of *T*, we form the extended set incorporating ~P, to give {P ∨ Q, ~Q, ~P}. Applying the resolution principle twice derives the empty clause, □. Hence P is seen to be wff of *T*. The refutation procedure may also be depicted as a tree of cancellations, as shown in the figure below.

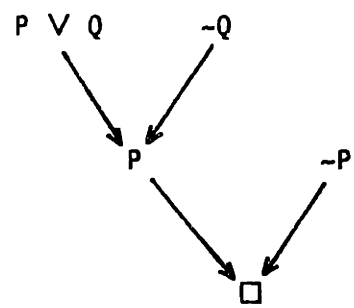


Figure 2-2:Example of a refutation tree.

The refutation principle applied to first-order predicate calculus is similar but requires one prior step, since we must deal with atomic formulas containing terms. This is handled by first performing a pattern-matching step called *unification*. The procedure involves binding variables together or instantiating constants for variables in order to obtain a uniform structure where possible.

For example, the set of literals {P(x),P(a)} can be unified by the substitution of x for a, denoted by x <- a or by a/x. As another example, the literals

{P(x,y) V Q(x,a), ~Q(f(b),w)}

can be unified by taking the unifiers f(b)/x and a/w. After the unification step, the refutation procedure for first-order logic is identical to that for propositional calculus.

2.1.2.6 A Note on Terminology

According to the syntactic view of logic, all valid statements take the form of axioms or theorems. An axiom is a statement which is assumed to be true, while a theorem is a statement derived from a set of axioms and/or other theorems through rules of deduction.

In fields of mathematics other than symbolic logic, however, it is customary to use other names as a matter of convenience to distinguish informally among such statements. In particular, an axiom is also called a *postulate*, *hypothesis*, or *assumption*.

A *theorem* is usually interpreted as a major or significant result derivable from axioms or other valid results. A *corollary* is an easily-derived theorem. A *lemma* is a corollary which is used as a stepping stone to prove a major theorem. Of course, one person's lemma may be another person's theorem: there are certain lemmas which are more widely known than most theorems. A *proposition* is a

theorem yet to be proved, while a *conjecture* is a dubious proposition which is about as likely to be false as true.

The remainder of this thesis adopts this informal set of appellations.

2.2 Set Theory

The fundamental concepts of set algebra are encapsulated in this section [Bittinger 1982; Gellert et al. 1975: 320-332, 343-380, 678-680; Lewis and Papadimitriou 1981: 5-23]. We begin by defining some primitive ideas.

Definition.

1. A *set* S is a collection of objects. An object s in the collection S is called a *member* or an *element* of S. We also say that s is *in* S, written $s \in S$, or that S *contains* s. If an object r is not in S, we write $r \notin S$.

2. The number of occurrences of an object in S is immaterial, as is its relative order within the set. Two sets S and T are *equal*, written $S = T$, iff they have the same elements; otherwise they are *unequal* and denoted $S \neq T$.

3. A *singleton* is a set with exactly one element. The *empty* set, denoted $\emptyset$, is the (unique) set with no elements; any other set is called *nonempty*.

4. A set S is a *subset* of a set T, written $S \subseteq T$, iff every object in S is also in T. If $S \subseteq T$ but $S \neq T$, then we say that S is a *proper subset* of T, and write $S \subset T$.

Two sets may be combined through a set operation to form a new set.

Definition. Let S and T be sets.

1. The *union* of S and T, written $S \cup T$, is the collection of objects which are members of S or T or both.

2. The *intersection* of S and T, written $S \cap T$, is the collection of objects which are in both S and T. If the intersection of S and T is empty, then the two sets are said to be *disjoint*.

3. The *difference* of S and T, written S - T, is the collection of objects in S but not in T.

We now introduce the notions of power sets and partitions.

Definition. Let S be a set.

1. The *power set* of S, denoted 2^S, is the collection of all the subsets of S.

2. A *partition* Π of a nonempty set S is a subset of the power set 2^S such that (a) each member of Π is nonempty, (b) members of Π are disjoint, and (c) the union of members in Π equals S.

These ideas may be illustrated by considering the set $S = \{a,b,c\}$. The power set of S is given by

$$2^S = \{\emptyset,\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$$

One way to partition S is to take $\Pi = \{\{a\},\{b,c\}\}$.

2.2.1 Relations and Functions

An ordered list is a collection of objects whose relative position or order is of significance.

Definition.

1. Let $L = \langle a_1,\ldots,a_n\rangle$ be a collection of objects whose relative positions are fixed. Then L is called an *ordered list*, *n-tuple*, or *vector*.

2. If $n = 2$ in the ordered list L, then L is called an *ordered pair*.

3. Two ordered lists $L = \langle a_1,\ldots,a_n\rangle$ and $M = \langle b_1,\ldots,b_n\rangle$ are *equal* iff $a_i = b_i$ for $i = 1,\ldots,n$.

From the definition we see that two ordered pairs $\langle a,b\rangle$ and $\langle c,d\rangle$ are equal iff $a = c$ and $b = d$. This property is preserved if we transform an ordered pair $\langle x,y\rangle$ into the set $\{\{x\},\{x,y\}\}$. In other words, the ordered pair $\langle x,y\rangle$ is equivalent to the set $\{\{x\},\{x,y\}\}$.

We next define the notions of Cartesian products, binary relations, and functions.

Definition. Let S, S_1, S_2,...,S_n, and T denote sets.

1. The *n-fold Cartesian product* of sets S_1, S_2,...,S_n, denoted $S_1 x S_2 x \ldots x S_n$, is the set of

all ordered tuples $\langle s_1, s_2, \ldots, s_n \rangle$ such that $s_i \in S_i$ for all $i = 1, \ldots, n$. S_i is called the ith *coordinate* or *dimension* of the n-fold Cartesian product.

2. The *binary Cartesian product* of S and T, also called simply the *Cartesian product* and denoted SxT, is the set of all ordered pairs $\langle s,t \rangle$ such that s is in S and t is in T.

3. A *binary relation* R on S and T is a subset of the Cartesian product SxT. In other words,

$$R \subseteq SxT = \{\langle s,t \rangle \mid s \in S \,\&\, t \in T\}$$

We also write $\langle s,t \rangle \in R$ in (a) infix notation as sRt, or (b) prefix notation as $R(\langle s,t \rangle)$ or more simply R(s,t).

4. A *function* or *mapping* from S to T is a binary relation f such that for each s in S, there is at most one ordered pair in R whose first component is s. In other words,

$$f \subseteq \{\langle s,t \rangle \mid \langle s,t \rangle \in R \,\&\, (\langle s,t \rangle \in R \,\&\, \langle s,t' \rangle \in R \rightarrow t = t')\}$$

In this case we write f: S -> T. S is called the *domain* of f and T the *codomain* (or *range* or *image*) of f. The object t in the ordered pair $\langle s,t \rangle \in f$ may also be written as f(s), since it is unique. When s is given, t is called the *image of s under f.*

5. A function f: S -> T is called *one-to-one* or *injective* iff for distinct objects s, s' $\in$ S, we have $f(s) \neq f(s')$. f is said to be *onto* T or *surjective* iff each element of T is the image of f of some object in S. f is called a *bijection between S and T* iff it is both one-to-one and onto T.

An operation is a special kind of function that associates with each ordered pair $\langle s,t \rangle$ a third element r, where r, s, and t are members of some set S.

Definition. Let S be a set. A function $f \subseteq SxSxS$ is called an *operation.* In other words

f: SxS -> S. The symbol f is called an *operator.*

For example, ordinary multiplication is an operation on the set of real numbers ***R***. In this case S ::= ***R*** and f ::= *, where the compound symbol ::= means *is defined as.*

More specifically, let S = T = ***N***, the set of natural numbers. Then

$$R = \{\langle i,j \rangle \mid n, m \in N \,\&\, i < j\}$$

defines the "less than" relation on the Cartesian product $N \times N$. The function

$$f = \{\langle i,j\rangle \mid n, m \in N \,\&\, j = i^2\} \subset R$$

defines the "squaring" function or operation. The function f is a bijection between the two sets {1,2,3,...} and {1,4,9,...}.

The image of an operation is given by $f(\langle s,t\rangle)$ for s, t $\in$ S. For notational convenience, we often remove the inner set of delimiters and write simply f(s,t). Moreover, we may use infix notation sft rather than the prefix notation f(s,t). For example, the multiplication operation f ::= * on the natural numbers N is usually written as s*t rather than *(s,t).

We now turn to the notion of composing relations or functions.

Definition. The *composition* of two binary relations P and Q, denoted P★Q or more simply PQ, is given by

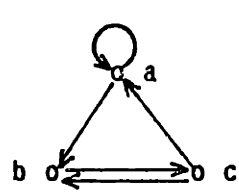
$$PQ = \{\langle p,q\rangle \mid \exists r(\langle p,r\rangle \in P \,\&\, \langle r,q\rangle \in Q)\}$$

If the relations happen to be functions, then the composition of two functions f: U -> V and g: V -> W is a third function h: U -> W such that h(w) = g(f(u)) for each u in U and some corresponding w in W.

2.2.2 Special Characteristics of Binary Relations

In this subsection we consider some special characteristics that may hold for a binary relation R which maps a set S into itself; that is, R $\subseteq$ SxS. The relation R may be depicted graphically as a *directed graph* in which nodes represent members of S and arcs stand for members of R. In particular, the graph has an arrow going from node s to node t iff the arc or ordered pair $\langle s,t\rangle$ is in R.

For example, let S = {a,b,c} and R = $\{\langle a,a\rangle,\langle a,b\rangle,\langle b,c\rangle,\langle c,a\rangle,\langle c,b\rangle\}$. The referent graph is shown in the accompanying diagram.



We next define the attributes of reflexivity, symmetry and transitivity.

Definition. Let R $\subseteq$ SxS be a relation on set S, and r,s,t $\in$ S. Then

1. R is said to be *reflexive* iff $\langle s,s\rangle \in R$ for each s in S. (In graphic terms, each node

has an arrow from itself to itself.)

2. R is said to be *symmetric* iff $\langle s,t \rangle \in R$ implies $\langle t,s \rangle \in R$. (Arrows between nodes come in pairs.)

3. R is said to be *bisymmetric* or *antisymmetric* iff (a) $\langle s,t \rangle \in R$ and $\langle t,s \rangle \in R$ implies that $s = t$, or (b) $\langle s,t \rangle \in R$ for distinct s and t implies that $\langle t,s \rangle \notin R$. Note that conditions (a) and (b) are equivalent constraints. (No pair of nodes has two opposing arrows.)

4. R is said to be *transitive* iff $\langle s,r \rangle \in R$ and $\langle r,t \rangle \in R$ together imply $\langle s,t \rangle \in R$. (If two nodes are connected by consecutive arrows, then the nodes are also connected directly.)

We illustrate these ideas by some examples. The relation "member of the family" is reflexive, symmetric and transitive. The relation "parent of" is antisymmetric and neither reflexive nor transitive. The relation "descendant of" is antisymmetric and transitive but not reflexive.

A relation that is reflexive, symmetric and transitive is called an *equivalence relation*. An equivalence relation R on set S divides S into a family of equivalence classes, each of which contains a collection of nodes which are totally connected. The equivalence class which contains a particular member s is denoted as [s]; that is, $[s] ::= \{t \mid \langle s,t \rangle \in R\}$.

The relationship between equivalence relations and partitions is given as follows.

Theorem. Let R be an equivalence relation on a set S. Then the equivalence classes of R comprise a partition of S.

Proof. Given in [Gellert et al. 1975: 324] and [Lewis and Papadimitriou 1981: 16], among others.

Hence there is an obvious bijection or isomorphism between the set of partitions of a set S and its set of equivalence classes.

We next define the notion or partial and total orderings.

Definition. Let R be a relation on a set S, and $s,t \in S$. Then

1. R is a *partial order* iff it is reflexive, antisymmetric, and transitive. Then we say that S is a *poset* or *partially ordered set*. We also denote R by the symbol $\leq$. Thus $\langle s,t \rangle \in R$ is written in infix notation as $s \leq t$.

2. R is a *total* or *complete order* iff (a) R is a partial order, and (b) for all s and t, either $\langle s,t \rangle \in R$ or $\langle t,s \rangle \in R$. Then we say that S is a *coset* or *completely ordered set*.

Hence a total order is a partial order in which any two objects may be compared.

To illustrate, the set of natural numbers *N* is partially ordered by the relation "a divides b". For example, 2 divides 4 but does not divide 5 without a remainder; hence we can say $2 \leq 4$ in this relation, but not $2 \leq 5$. On the other hand, the ordinary inequality relation "less than or equal to" is a total order on *N*.

Another example of a partial order is the family of subsets of a set S under the ordinary subset relation $\subset$. Further, the relation "descendant of" is a partial order on the set of all people; but it is not a total order, since siblings cannot be compared directly according to this relation.

We define the notions of precedence and succession among elements of an ordered set.

Definition. Let R be an order relation on set S, and $r,s,t \in S$. We use infix notation $s \leq t$ to mean $\langle s,t \rangle \in R$. Then

1. s is a *predecessor* of t, written $s < t$, iff (a) $s \leq t$ and (b) $s \neq t$. Then we also say that t is a *successor* of s and write $t > s$.

2. s is an *immediate predecessor* of t, written $s <_i t$, iff (a) $s < t$ and (b) there is no other r such that $s < r$ and $r < t$. Then we also say that t is an *immediate successor* of s, and write $t >_i s$.

We next define the notion of a chain or path.

Definition. Let R be a binary relation.

1. A *chain* or *path* in R is a sequence $Se = \langle s_1,\ldots,s_n \rangle$ for some $n \geq 2$ such that $\langle s_i,s_{i+1} \rangle \in R$ for $i = 1,\ldots,n-1$.

2. The path $Se = \langle s_1,\ldots,s_n \rangle$ is a *cycle* iff (a) $s_1, s_2,\ldots, s_{n-1}$ are distinct, and (b) $s_n = s_1$.

A cycle Sc is *trivial* iff $n = 2$; otherwise it is *nontrivial*.

A minimal element of a relation is given by a node which is not at the head of any arrow except possibly for self-loops. Similarly, a maximal element is one that is not at the tail of any arrow except perhaps its own.

Definition. Let R be a partial order relation on S, with $r,s,t \in S$.

1. A *minimal* element of R is a member s such that $\langle r,s \rangle \in R$ implies $r = s$.

2. A *maximal* element of R is a member s such that $\langle s,t \rangle \in R$ implies $s = t$.

3. An *extremal* element of R is a maximal or a minimal element of R.

A partial order relation may have more than one extremal element.

A hierarchy is a relation with unique immediate successors, a single maximal element, and no self-loops.

Definition. Let R be a partial order relation on S, with $r,s,t \in S$. Then R is called a *hierarchical partial order*, or simply *hierarchy*, iff

1. R is loop-free. That is, there is no s such that $\langle s,s \rangle \in R$.

2. Each member of S has at most one immediate successor. That is, $s <_i r$ and $s <_i t$ implies $r = t$.

3. R has exactly one maximal element, which is called the *root*.

We now define the notion of closure. Loosely speaking, a set is closed under a relation if the image of any permutation of objects in the set, is also in the set.

Definition. Let S be a set, and $R \subseteq S^{n+1}$ be an $(n+1)$-ary relation on S, where $n \geq 0$. Then a subset C of S is said to be *closed under R* iff $c_1,\ldots,c_n \in C$ and $\langle c_1,\ldots,c_{n+1} \rangle \in R$ imply that $c_{n+1} \in C$.

For example, the set of natural numbers *N* is closed under addition since the sum of two such numbers is again a natural number. In contrast *N* is not closed under subtraction, since the difference of two numbers may be a negative number.

2.3 Algebraic Structures

This section builds on the discussion of set theory and extends the concepts to algebraic structures. An algebraic structure is first defined as an ordered pair consisting of a set of objects and a set of relations on that set. This general definition is then used as an integrative framework to unify various constructs such as lattices, groups, fields, and rings [Birkhoff 1963; Birkhoff and Bartee 1970; Birkhoff and MacLane 1965; Gellert et al. 1975].

An *algebraic structure* is a set on which certain relations or operations are defined. An example of a relation is "less than", denoted by the symbol <. Examples of operations are set union denoted by the operator ∪, and ordinary multiplication denoted by the operator *.

> **Definition.** An *algebraic structure* is the ordered pair A = <S,R> where S is a set of objects and R is a set of relations or operations defined on S. That is, each relation or operation R ∈ R represents a mapping from S^n -> S for some n.

Note that every member of R is itself a relation.

For notational convenience, we may drop the brackets around the second coordinate of a structure. For example, the structure which specifies addition and multiplication on the natural numbers may be written simply as <N,+,*> rather than as <N,{+,*}>.

2.3.1 Graphs

An algebraic structure with a single binary relation is called a graph. In other words, n = 1 in the previous definition, and R = {R} where R is a subset of SxS.

> **Definition.** Let u, v be members of a set V, and E a binary relation on V.
>
> 1. The structure G = <V,{E}> is called a *directed graph, digraph* or simply *graph.* V is called the set of *vertices* or *nodes*, and E the set of *edges* or *arcs.* The relation E may be denoted by the symbol ->. Thus <u,v> ∈ E may be written in infix notation as u -> v, and G may be written as G = <V,{->}>.
>
> 2. The structure G = <V,{E}> is called an *undirected graph* iff (a) G is a directed graph, and (b) E is a symmetric relation, i.e. for all u and v, <u,v> ∈ E implies <v,u> ∈ E. Then we also denote the relation E by the symbol ↔.

As seen in Section 2.2, graphs are simple to interpret pictorially. Each edge <u,v> may be depicted by an arrow whose tail is at u and head is at v.

2.3.2 Lattices

The idea of a lattice originated with the attempt to generalize a number of relationships among sets and subsets, as well as substructures of structures such as groups and fields [Birkhoff and Bartee 1970; Birkhoff and MacLane 1965].

Definition. Let a, b, c be members of a set L, while $\cap$ and $\cup$ represent two operations called respectively *intersection* and *union*. The structure $L = \langle L,\{\cap,\cup\}\rangle$ is called a *lattice* iff the following axioms hold:

1. Commutativity: $[a \cap b = b \cap a]$ &
   $[a \cup b = b \cup a]$

2. Associativity: $[(a \cap b) \cap c = a \cap (b \cap c)]$ &
   $[(a \cup b) \cup c = a \cup (b \cup c)]$

3. Absorption: $[a \cup (a \cap b) = a]$ &
   $[a \cap (a \cup b) = a]$

Every lattice L is equivalent to a partially ordered set S(L) whose elements are the same as those of L. For example, the subsets of a set comprise a lattice under ordinary intersection and union. To illustrate, let 2^S be the power set of S = {a,b,c}. Let the members of S be A = Ø, B = {a}, C = {b}, D = {c}, E = {a,b}, F = {a,c}, G = {b,c}, H = {a,b,c}. In the figure below,

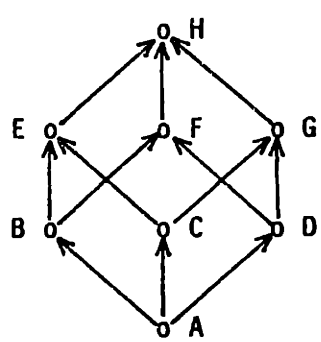


Figure 2-3: Example of a lattice.

the arc <X,Y> indicates that set X is a subset of set Y.

Another example of a lattice comes from propositional logic. Here L is the set of propositions while the operations $\cap$ and $\cup$ correspond respectively to the logical operations & and $\vee$.

A bijective mapping μ: L -> L' from a lattice L onto a lattice L' is called an *isomorphism* iff for a,b $\in$ L:

$$\mu(a \cap b) = \mu(a) \cap \mu(b) \ \& \ \mu(a \cup b) = \mu(a) \cup \mu(b)$$

Because of their generality, lattices occur frequently in the algebraic interpretation of systems theory. As later chapters will show, lattices may represent both spatial and temporal relationships among collections of systems. For example, the network organization of nested subsystems corresponds to a lattice, while a hierarchical configuration is a special type of lattice with a unique superset or successor.

2.3.3 Groups and Related Structures

A set with a mapping from ordered pairs of the set to itself is called an *operation* or *groupoid.* These and related structures are defined below.

Definition. Let G be a set and * an operation on G; that is, *: GxG -> G. Then

1. The algebraic structure G_p = <G, {*}> is called a *groupoid* or *operation.*

2. The structure G_s = <G, {*}> is called a *semigroup* iff * is associative; that is, (a*b)*c = a*(b*c) for any a,b,c in G.

3. The structure M = <G, {*}> is called a *monoid* iff (a) M is a semigroup and (b) G contains an identity element e such that e*a = a*e = a for any a in G.

4. The structure G = <G, {*}> is called a *group* iff (a) G is a monoid and (b) all inverses exist, i.e. for each a $\in$ G, there is some $a^{-1} \in$ G such that $a*a^{-1} = a^{-1}*a$ = e.

5. The structure G_a = <G, {*}> is called an *Abelian group* iff (a) G_a is a group and (b) * is commutative, i.e. a*b = b*a for all a,b $\in$ G.

To illustrate, let N = {1,2,3,...} be the set of natural numbers, Z = {...,-2,-1,0,1,2,...} the set of

integers, R the set of reals, and M the set of nxn matrices for some n. Then an example of a groupoid is $\langle N,\{+\}\rangle$, the set of natural numbers under ordinary addition. Since the + operation is associative, this structure also happens to be a semigroup; but it is not a monoid since no identity element (i.e. 0) is included in N.

Further, M is a semigroup under matrix addition but not under multiplication, since $AB \neq BA$ for two matrices A and B. Also, Z is a monoid under addition but not under multiplication since, for example, the element 5 has no inverse in Z. Finally R is a group--in fact an Abelian group-under both addition and multiplication.

The relationships among group-like structures are depicted in the figure below.

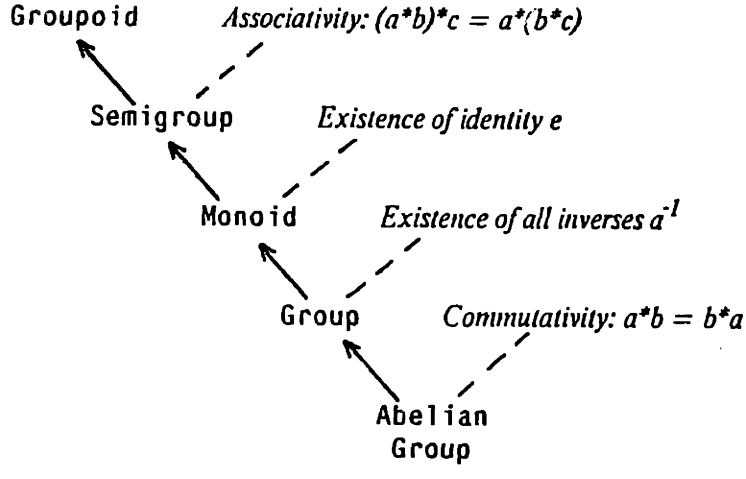


Figure 2-4:Precedence ordering among group-like structures in terms of set inclusion.

The descriptions in italics specify referent constraints. For example a semigroup is a groupoid with an associative operation; due to this constraint, the family of semigroups is a proper subset of the family of groupoids.

2.3.3.1 Homomorphism

One concept of homomorphism for a group pertains to the sets of elements of the group, while another relates to the operations defined on the group.

Definition. A mapping f of a group $G = \langle G,\{*\}\rangle$ into a group $G' = \langle G',\{*'\}\rangle$ is called a *homomorphism* iff the following axiom, called *relation H*,

$$f(a*b) = f(a)*'f(b)$$

holds for any $a,b \in G$.

The left-hand side product * is taken in G, while the right-hand side product *' is taken in G'.

The image of G under f represents a subgroup of G'. G' is called a *homomorphic image* of G iff there is a surjective homomorphism of G into G'; that is, every member of G' occurs as an image under f. A homomorphism need not be injective: it is possible for distinct elements in G to be mapped to the same image in G'.

To illustrate, consider the group defined by the set G of all 2x2 matrices under multiplication. The determinant is a homomorphic mapping from GxG into G' = R, since the H relation holds. That is, for all A, B in G:

$$\det(A*B) = \det(A) *' \det(B)$$

where * denotes matrix multiplication and *' denotes multiplication between real numbers. The homomorphism is surjective, since every real number g' in G' can be obtained as

$$\det \begin{vmatrix} 1 & 0 \\ 0 & g' \end{vmatrix}$$

Since different matrices can have the same determinant, however, the homomorphism is not injective.

A homomorphism preserves the structure of the domain G to some degree. Since the mapping need not be injective, however, it tends to lose some information: the inverse function does not exist when two objects in the domain have the same image. In this sense, a homomorphism is a one-way street from the domain to the range.

The set of elements that map into the unit element e' in the image G' is a measure of the "compression", "shrinkage", or information loss of G. These elements form a special subgroup of G, called the *kernel* of the homomorphism. In the example of the 2x2 matrices, the kernel is the set of all 2x2 matrices whose determinant is 1 (= e').

Objects in the real world, whether in the physical, social or life sciences, are usually complex phenomena characterized by myriads of properties and linkages. The only way to cope with such complexity is to ignore most of it. A real system must be analyzed by building simplified models, each of which represents a homomorphism of the original. Hopefully, the homomorphic models capture enough of the salient characteristics of the real system to produce useful results.

2.3.3.2 Isomorphism

A two-way homomorphism gives rise to an isomorphism.

Definition. An *isomorphism* is a bijective homomorphism.

If f is an isomorphism of G into G', then G' is also the image of F. The kernel of f is the unit subgroup of G. If there is an isomorphism from G to G', we say that the two are *isomorphic* and write $G \simeq G'$.

For example, let R^+ be the additive group of real numbers and R^* the multiplicative group of reals. Then the mapping f which takes a number x into the natural logarithm ln(x) is an isomorphism. The relation ln(a*b) = ln(a) + ln(b), corresponds to the functional notation f(a*b) = f(a) *' f(b), where * is the operation in R^* and *', taken here as +, is the operation in R^+. Hence $R^* \simeq R^+$.

Isomorphism is an equivalence relation between groups, so that the family of all groups may be partitioned into isomorphic classes. Since isomorphic groups have the same structure, calculations in them follow the same rules even through the objects and/or operations may be of different kinds.

2.3.4 Other Structures

A field is a structure which allows for ordinary arithmetic. For example, the rational numbers, the real numbers, and the complex numbers fall into this class.

Definition. Let F be a set, and + and * be operators defined on F. Then the structure F = <F,{+,*}> is a *field* iff

1. F defines an Abelian group on the operation +, with zero element 0.

2. The set F - {0} defines an Abelian group on the operation *.

3. The operators + and * are related by the distributive axiom; that is,

$$a*(b + c) = (a*b) + (a*c)$$

for any a,b,c in F.

A field E = <E,{+,*}> is called a *subfield* of F = <F,{+,*}> if E is a subset of F. In this case F is also called an *extension field* of E. For example, the complex numbers are an extension field of the set of reals, which in turn is an extension field of the set of rationals.

A ring is a structure possessing an addition operation which is commutative, but a multiplication operation which is not.

Definition. Let S be a set, while + and * are two operators called respectively *addition* and *multiplication*. The structure $Ri = \langle S,\{+,*\}\rangle$ is called a *ring* iff

1. The + operation with S defines an Abelian group; that is, $\langle S,\{+\}\rangle$ is a commutative group.

2. The + and * operators are related by the distributive axiom:

$$a*(b + c) = (a*b) + (a*c) \quad \& \quad (a + b)*c = (a*c) + (b*c)$$

for any a,b,c in S.

If the multiplication operation is associative, the structure *Ri* is called an *associative ring*; if multiplication is commutative, *Ri* is called a *commutative ring*.

For example, the set of complex numbers under ordinary + and * defines a commutative ring. An example of an associative ring which is not commutative is the set of nxn matrices under matrix addition and multiplication.

Rings play a central role in the study of abstract algebra. The notion of an algebra, in particular, may be defined in terms of rings which are vector spaces.

Definition. Let $A = \langle A,\{+,*\}\rangle$ be an associative ring. Then A is called an *algebra* iff

1. The additive group of A, denoted $A^+ = \langle A,\{+\}\rangle$, is a vector space over a field $F = \langle F,\{+,*\}\rangle$.

2. Multiplication by the scalar elements of the field commutes with ring multiplication; that is

$$(\alpha*a)*b = a*(\alpha*b)$$

for α in F and a,b in A.

The theory of representations is closely tied to the theory algebras. It addresses the issue of mapping a group, ring, or algebra homomorphically; that is, transforming them into a simpler representation while still retaining important characteristics of their original structure. The transformed structure is a group or ring of matrices or linear transformations of a vector space. Such a vector space is called a representation space.

Chapter Three

Infrastructure

Chapter 2 has shown how the notion of an algebraic structure can serve as a unifying framework for objects such as lattices, groups and fields. This chapter highlights the power and elegance of these simple structures by showing how they in turn may be used in the analysis of manufacturing systems to:

1. Serve as an infrastructure for making precise, various qualitative notions such as system connections and hierarchies.

2. Provide a uniform framework for more specialized theories such as automata theory, information theory, and control theory.

3. Serve as a foundation on which to build quantitative theories.

The first section presents a set of definitions of a system and associated concepts. Then the discussion delves into general structures for representing system characteristics--in particular, the graph, matrix and state space representations. Finally, the last section presents a typology of system models in terms of black, gray and beige boxes. According to this classification the models of automata theory are grey boxes, while those of control theory and information transmission are beige boxes.

3.1 Systems Framework

Systems theory, also known as systemics or general systems theory, is based on the premise that there exist (1) concepts and structural characteristics that seem to apply to most systems, and (2) modeling approaches such as the state space representation which appear to be generally applicable. Systems theory as a discipline provides a foundation of basic concepts and a framework on which to hang other specialized theories [Boulding 1956; Bunge 1979; von Bertalanffy 1950, 1958; etc.] Hence it offers an integrative perspective for such disciplines as automata theory, linear systems theory, and control theory.

3.1.1 Typology of Systems

The world around us seems to consist of 5 genera of systems: physical, chemical, biological, social and technical [Bunge 1979: 245-250]. The last category, technical systems, refers to artifacts of human construction as exemplified by tools and skyscrapers. Let these 5 genera of systems be designated respectively as S_1, S_2, S_3, S_4, and S_5.

Each of these system genera may be further subdivided by scale, such as the 3-way nominal classification consisting of micro, meso, and mega levels. Examples of systems in each classification are given in the figure below.[2]

	SCALE		
GENERA	Micro	Meso	Mega
S_1=Physical	Atom	Asteroid	Galaxy
S_2=Chemical	Monomer reactors	Polymer reactors	Compost pile
S_3=Biological	Single cell	Animal	Ecosystem
S_4=Social	Family	Village	City or nation
S_5=Technical	Book	Industrial plant	Global telecommunications system

Figure 3-1:Examples of systems by genera and scale.

According to this classification, manufacturing plants, for example, are mesotechnical systems.

We make the assumption that every concrete or material system is assembled from objects in the same or lower genera. First we define the notion of precedence.

> **Definition.** Let $S = \{S_1, S_2, S_3, S_4, S_5\}$ be the family of system genera. Then S_i *precedes* S_j, written $S_i < S_j$, iff objects of genera S_i serve as components or agents in the assembly of every system of S_j.

[2]Adapted from [Bunge 1979: 246-247].

We assert that every system except those in S_1 are preceded by lower-level objects.

Postulate.

1. For any $x \in S_j$, there is at least one $y \in S_i$, for $S_i < S_j$, such that y has served in the assembly of x.

2. The precedence relations among the system genera are $S_1 < S_2 < S_3 < S_4, S_5$

The system genera represent a partial ordering, since S_4 and S_5 are found at the same level while others are not. In other words, the tuple $\mathcal{S} = \langle S, < \rangle$ is a partially ordered structure. In a narrower and more specific context, a similar precedence ordering can be defined on all human artifacts. Examples lie in the assembly of manufactured products or in the design of industrial plants.

3.1.2 Aggregates versus Systems

This subsection gives a rigorous presentation based on set structures of notions such as system, component, bondage, and environment.[3] A set of objects may be interdependent or isolated from each other: a set is called a system if its elements are interrelated, and is otherwise called an aggregate.

Definition.

1. A set of objects is *independent* iff the objects have no bonds among them and do not interact. The set is then called an *aggregate.*

2. A set of objects is *dependent* iff they are not independent; the set is then called a *system.*

---

[3]Perhaps one of the most important works on systems thinking is [Bunge 1979], which does a commendable job of bringing order to a complex field where even basic concepts have no universally accepted definitions. The book, unfortunately, contains a surprising number of errors. Chapter 1 alone is problematic for the following reasons: (1) *Incompleteness.* For example, the discussion of atomic or A-composition on pp. 5-6 is inadequate since the set A is described merely as a "class of things" without further specification. (2) *Incorrectness.* For example, item (i) of Definition 1.8 in Bunge defines the *precedes* relation as we define *immediately precedes* here; but this is not sufficiently general to permit the recursive notion of "preceding levels" in item (ii) of that definition. (3) *Inconsistency.* For example, Bunge speaks of "systems of kind K" on p. 20; but since K is used as a set there, it only makes sense to speak of "kind k" for $k \in K$. (4) *Idiosyncrasy.* For example, Bunge does not allow for the notion of hierarchy because organizational levels are not considered to imply a "dominance relation" (p. 14); but most authors would argue otherwise. Despite these difficulties, an effort has been made to assure consistency with Bunge's terminology when practical, rather than to add to the proliferation of terminology.

Objects may also be categorized along another dimension in terms of tangibility: they may be abstract (conceptual) or concrete (material). An example of an abstract aggregate is the set {0,1}. The notion of a concrete aggregate is illustrated by the superposition of two waves, or by an assortment of samples from different production lines.

An abstract system is exemplified by a set of axioms and theorems, while a concrete system is illustrated by the set of machine tools in a manufacturing cell.

Any system, whether abstract or concrete, may be defined in terms of its (1) composition or component objects, (2) structure or relationships among its components, and (3) environment or collection of external objects with which it interacts. For example, the composition of a manufacturing cell is the set of machine tools which comprise the cell; its structure the set of relationships among the machines; and its environment the collection of other objects in the factory to which it interfaces. We define these concepts more formally below.

We first introduce the notion of the composition of an object as the set of its parts. Let x and y be members of Ω, the set of all objects or things. We let the symbol $\sqsubset$ denote the *part-of* relation, so that $y \sqsubset x$ means that y is a part of x.

Definition. The *(total) composition* of x is

$$Co(x) = \{y \mid y \sqsubset x\}$$

It would be inaccurate and infeasible, however, to try to characterize a system in terms of its parts at arbitrary levels. Consider a set of robots, for example. The molecules in all the robots are interdependent in at least the sense that they cannot occupy the same location in space and time; but these molecules do not otherwise form meaningful relations.

At a higher level, even the microprocessors in the robots cannot usefully be considered to comprise a system, since they are unable to interact by themselves. However, the collection of robots themselves may be viewed as the basis for a system.

The idea of restricting our attention to reasonable relationships among objects is given in the next definition.

Definition. A set of objects is *linkable* iff the objects can form plausible relationships.

The word *plausible* is used in the definition above to preclude insipid relationships such as those

among the molecules of the robot in the preceding example.

Let L be a class of linkable objects in Ω. Then the *L-composition*, or *composition at level L*, of an object x is the collection of the parts of x that belong to L:

$$Co_L = Co(x) \cap L = \{y \in L \mid y \sqsubset x\}$$

We turn now to the idea of links among the components of an object. It is important to distinguish between logical relations and structural relations. An example of the former is weight, as in "u is heavier than v"; an exemplar of the latter occurs when one billiard ball strikes another. Logical relations are passive, while structural relations tend to be active. The *logical bondage* Lb_L of a set L of objects is the set of logical relations among the members of L.

An object is said to *act* on another if the former affects the latter's history or behavior line. This behavior-modification may occur either actually or potentially.

If an object a acts on b, then we write $a \rhd b$. If b also acts on a, then a and b are said to *interact*. If b does not act on a, then a is called the *agent* and b the *patient*.

Two objects are *connected*--or *bonded* or *linked* or *coupled*--if at least one object acts on the other. The *(structural) bondage* B_J of a set J of objects is the set of structural relations (i.e. bonds, connections, links, or couplings) among them.

These ideas are used in defining the *L-environment* of an object x whose L-composition is $Co_L(x)$:

$$En_L(x) = \{y \in \Omega \mid \sim(y \in Co_L(x)) \;\&\; (\exists z)(z \sqsubset x \;\&\; (y \rhd z \vee z \rhd y))\}$$

In other words, the L-environment of x is the set of objects outside $Co_L(x)$ which act on or are acted upon by parts in x.

The notion of *environment* used here is that of the immediate environment of an object. Other writers often use the term to mean the total environment, or the set of all things which are not part of a given object.

The *structure* of an object is the set of all relations among the components, plus the relations among its components and objects in the environment. We now define the notion of a concrete system.

Definition. An object is a *concrete system* iff it is composed of two or more concrete,

connected objects.

Let Σ denote the set of all concrete systems; then the composition, environment and structure of any system in Σ are given as follows.[4]

Definition. Let $\sigma \in \Sigma$ be a concrete system and $L \subset \Omega$ a class of objects. Then

1. The *L-composition* of σ at time t is the set of its L-parts at t:

$$Co_L(\sigma,t) = \{x \in L \mid x \sqsubset \sigma\}$$

2. The *L-environment* of σ at time t is the set of all objects of type L, not components of σ, that act on or are acted upon by components of σ at t:

$$En_L(\sigma,t) = \{x \in L \mid x \notin Co_L(\sigma,t) \;\&\; (\exists y)(y \in Co_L(\sigma,t) \;\&\; (x \rhd y \vee y \rhd x))\}$$

3. The *L-structure*, or *organization*, of σ at time t is the set of relations or bonds among the components of σ, and among the components and objects in the environment of σ:

$$St_L(\sigma,t) = \{R_i \in B_L(\sigma,t) \cup Lb_L(\sigma,t) \mid B_L(\sigma,t) \neq \emptyset\}$$

where $B_L(\sigma,t)$ denotes the set of structural bonds and $Lb_L(\sigma,t)$ the logical bonds on the objects in $Co_L(\sigma,t) \cup En_L(\sigma,t)$.

We will often drop the subscript L and simply write Co, En and St when the level is understood or unimportant.

The simplest example of a system occurs when two objects, a and b, are connected and nestled in an environment which is treated as a single object c. In this case $L = \{a,b,c\}$, $Co_L(\sigma) = \{a,b\}$, and $En_L(\sigma) = \{c\}$. The internal structure may take one of three forms, as shown below.

$a \rhd b$:	$o \longrightarrow o$
$b \rhd a$:	$o \longleftarrow o$
$a \rhd b \;\&\; b \rhd a$:	$o \longleftrightarrow o$

[4] We may also define a system in terms of an algebraic structure. The structure $E = \langle E,R \rangle$ represents an *extended system* σ_E iff (a) E is the union of two disjoint sets of objects, called *composition* Co and *environment* En, and (b) R is the union of two disjoint sets R_{Co} and R_{En}, where $R_{Co} \subset Co \times Co$ represents the set of *internal bonds* and $R_{En} \subset (Co \times En) \cup (En \times Co)$ denotes the set of *external bonds*. Then the ordered pair $S = \langle Co, R \rangle \subset E$ represents system σ for Co and R defined as above.

The external structure, in addition, may consist of any collection of these forms: {a [> c}, {b [> c}, {c [> a}, and {c [> b}.

Our knowledge of a system is complete when all of the following information is available: (1) composition, environment and structure of the system; (2) history of the system, and (3) laws of the system. Such exhaustive knowledge is usually unavailable for systems of reasonable complexity. The minimal qualitative model, however, would require knowledge of at least the system's composition, environment, and structure:

$$S_L(\sigma,t) = \langle Co_L(\sigma,t), En_L(\sigma,t), St_L(\sigma,t)\rangle$$

Supplementing this minimal model with another construct such as the state space representation may be sufficient for a useful quantitative model.

3.1.3 System Organization

The set of relations which constitute the structure of a system may be partitioned into internal connections and external connections. However, structure must be distinguished from spatial *configuration*. All systems, both abstract and concrete, by definition possess structure or bondage; but only concrete systems have spatial structures. In contrast, concrete aggregates have spatial structures but no bondage, while abstract systems have neither.

3.1.4 Subsystems

If a component of a system is itself a system, then it is called a subsystem.

Definition. Let σ be a system with composition $Co(\sigma,t)$, environment $En(\sigma,t)$ and structure $St(\sigma,t)$ at time t. Then an object x is a *subsystem* of σ at t, written $x \prec \sigma$, iff both these conditions hold:

1. x is a system at time t.

2. $Co(x,t) \subseteq Co(\sigma,t)$ & $En(x,t) \supseteq En(\sigma,t)$ & $St(x,t) \subseteq St(\sigma,t)$.

By construction the subsystem relation is an order relation; the relation $\prec$ is reflexive, asymmetric, and transitive. For example, a piston is a component of the engine which is a subsystem of a car.

The idea of having systems-within-systems may be generalized to the recursive notion called nesting.

In the definition below we take the compound symbol $::=$ to mean that the quantity on the left-hand side is defined as that on the right-hand side.

Definition. Let σ be a system, Σ the collection of all systems, and

$$N_\sigma = \{\sigma_i \in \Sigma \mid \sigma ::= \sigma_0 \,\&\, \sigma_i \prec \sigma_{i+1} \text{ for } i = 0,1,2,\ldots,n\}$$

a set of supersystems of σ partially ordered by the subsystem relation $\prec$. Then

1. N_σ is a system of *nested* systems with core σ.

2. The *primary structure* of σ is the structure of σ itself; the *secondary structure* of σ is the structure of σ_1, the smallest supersystem of σ in N_σ. In general, the *n-ary structure* of σ is the structure of σ_{n-1}.

The nesting characteristic makes it apparent that the set Σ of all systems has the structure of a lattice.

Systems can usually be viewed as consisting of objects at different levels of organization. For example, a manufacturing plant may be composed to cells which in turn are composed of machine tools.

Definition. Let $L = \{L_i \mid 1 \leq i \leq n\}$ be a family of n nonempty sets of objects. Then

1. Level L_i *immediately precedes* level L_j iff all the objects in the latter are composed of (some or all) objects in the former.

2. Level L_i *precedes* level L_j iff L_i immediately precedes L_j or L_i immediately precedes another level L_k which precedes L_j. We also say that L_j *succeeds* L_i.

3. An object *belongs to level* L_i iff it is composed of objects in (some or all) of the preceding levels.

4. The tuple $L = \langle L, \langle \rangle$ is a *level structure.*

For example, the chemical level precedes the biological level since all objects in the latter are composed of those in the former.

The ideas of nesting and precedence allow for the case where subsystems have overlapping components. When an object is a component of more than one supersystem, it is said to have more than one organizational parent. Such a structure corresponds to a lattice.

When each object has only one organizational parent, a different structure is indicated. In particular, a hierarchy is a family of nested objects, each of which has exactly one organizational parent except for a unique object called the root, which has none.

Definition. Let σ and σ' be systems, while x is some object. Then

1. σ is called the *organizational parent* of x iff σ is the smallest supersystem of x, and it is unique. That is, x has only one organizational parent. Conversely, we say that x is the *organizational offspring* of σ.

2. σ is called an *organizational ancestor* of x iff (a) σ is the organizational parent of x, or (b) σ is the organizational parent of some system σ' which is an organizational ancestor of x. Conversely, x is called an *organizational descendant* of σ.

3. The family of objects consisting of σ and its organizational descendants is called an *organizational hierarchy* or simpley *hierarchy*. Moreover σ is called the *root* of the hierarchy.

The adjective *organizational* will be omitted when the context makes it is clear that we are referring to an organizational relationship rather than to a biological phenomenon.

3.1.5 Association

Association refers to the combination or juxtaposition of objects. Any two objects x and y may be considered to associate, or to form a compount object $z = x +_a y$. Thus the set of all objects is closed under the association operator $+_a$.

In contrast, two systems will not necessarily merge to form a third system z. In other words, x and y would be components of z except that no bonds exist between x and y. Hence systemicity is not preserved. For example, an assembly robot may or may not form a system with a spray-painting robot.

In general the environment, structure and composition of the resulting system is not given by the union of the individual systems. For the set of all systems Σ, there is no function which maps $\Sigma x \Sigma$ into Σ. Hence Σ does not have the structure of a groupoid nor a semigroup. As discussed previously, however, Σ does have the structure of a lattice. Moreover, since systems are objects, they obey the

algebra of things, such as closure under association.

3.1.6 Input and Output

Connections among objects may be characterized along temporal attributes in terms of duration and variability. The simplest partitioning of *duration* is in terms of *temporary* and *permanent* categories, while that of *variability* is in terms of *static* and *dynamic* categories.

When a connection is dynamic, it is often called a *flow*, as in the flow of electromagnetic energy. A physical flow may or may not be organized so as to carry information from one object to another. A physical flow which conducts information forms part of an *information system*. A physical flow might not carry information, but information depends on some physical flow such as energy.

A closed or isolated system is one that does not interact with the outside world.

Definition. Let σ be a system with environment $En(\sigma,t)$. Then σ is said to be *closed* at time t iff $En(\sigma,t) = \emptyset$; otherwise σ is *open*.

A system may be closed in some respects and open in other respects. An unopened soda can, for example, is open to heat flow but closed to fluids.

Definition. Let P be a property of a system σ in an environment $En(\sigma,t)$. We say that σ is *open with respect to P at t* iff P is related to at least one property of objects in $En(\sigma,t)$ at time t; otherwise σ is *closed with respect to P*.

We next define the notion of input and output between a system and its environment.

Definition. Let σ be a system with (immediate) environment En, and $Ac(u,v)$ be the (total) action or effect of an object u on object v [cf. Subsection 3.2.3]. Then

1. The totality of *inputs* of σ is the set of all environmental actions on σ:

$$U(\sigma) = \bigcup_{x \in En(\sigma)} Ac(x,\sigma)$$

2. The totality of *outputs* of σ is the set of all actions of the system on its environment:

$$V(\sigma) = \bigcup_{y \in En(\sigma)} Ac(\sigma,y)$$

3. The *activity of the environment* of σ is

$$E(\sigma) = \bigcup_{x,y \in En(\sigma)} Ac(x,y) \cup U(\sigma) \cup V(\sigma)$$

We hypothesize that every system receives some input from the environment but filters them so that it accepts only part of the input. Moreover, every system undergoes some spontaneous changes which is not traceable to any particular input. The machines of automata theory might seem at first sight to be counter-examples to these assertions; but such machines are only abstract creations.

3.2 System Representations

Two common and equivalent ways to represent connections between pairs of objects are as graphs or matrices [Bunge 1979; Simon and Ando 1963; Warfield 1974]. These linkage representations may then be given quantitative expression through the state space approach [Dorf 1974; Hsu and Meyer 1968; Padulo and Arbib 1974].

3.2.1 Graph Representation

A *graph* is a set of vertices and edges which connect pairs of vertices. An edge linking two vertices V_1 and V_2, may be represented algebraically as $\langle V_1, V_2 \rangle$ or pictorially as a line segment.

If an object a acts on another object b, the action may be positive or negative. In the first case a is said to *activate* or *stimulate* b, and we denote it as a --> b. In the second case, a is said to *deactivate* or *inhibit* b, and we use the notation a -/-> b. In the figure below,

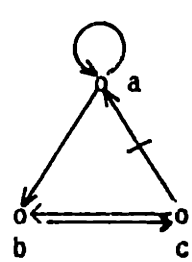


Figure 3-2:Diagram of a graph with activation bonds.

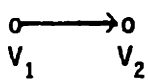
a activates itself and b; b and c activate each other, and c inhibits a.

Let σ be a system with m components and n different kinds of connections such as mechanical,

electrical and informational. Then σ is representable by a set of directed graphs $\{G_1, G_2, \ldots\}$ over the composition of σ, where G_i represents the ith kind of connection.

The graph $G_i = \langle V_i, E_i \rangle$ consists of the set of nodes V_i, whose elements represent the components of σ, and the set of edges E_i which represent the connections among them. The *connection capacity* of a system is the total possible number of pairwise connections, excluding self-connections $\langle x,x \rangle$. Since each of the n types of connections allows for $m(m - 1)$ such links, the total connection capacity is $nm(m - 1)$.

When some kind of precedence ordering may be established between pairs of vertices, the corresponding edges are called *directed.* The corresponding graph is called a directed graph, or *digraph.* A directed edge between vertices V_1 and V_2 is represented graphically as an arrow:



We say that V_1 *precedes* or *subcedes* V_2, while V_2 *succeeds* or *supercedes* V_1.

A *path* is a sequence of vertices which are serially connected by edges.

Example 1.

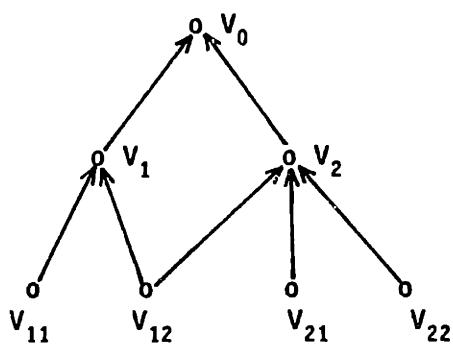


Figure 3-3:Example of a graph. The sequence $\langle V_{11}, V_1, V_0 \rangle$ is a path.

We define a *deoriented graph* as a digraph in which the sense of direction is ignored; in other words, any directed edges are replaced by undirected edges. In algebraic notation, we may say that if any edge $\langle a,b \rangle$ is in E_i, so is $\langle b,a \rangle$.

If a path exists between any two vertices V_i and V_j in a deoriented graph, we say that the graph is

connected. A *cycle* is a nonempty path containing each vertex only once, except for the starting and ending vertices, which are identical.

A nonempty connected graph is called a *tree* if it contains no cycles. A *top* is a vertex which precedes no other vertex. A *hierarchy* is a digraph with the following properties:

1. The corresponding deoriented graph is a tree.

2. The digraph has only one top.

The digraph in the preceding example is not a hierarchy; but removal of the directed edge $\langle V_{12}, V_2 \rangle$, for example, would transform it into one.

Example 2. Let F_1 and F_2 denote the main functional requirements of a given design. The first requirement, F_1, has two subordinate functions F_{11} and F_{12}. We may define a "consolidated" functional requirement, F_0, to represent the composite requirement for the whole design. These relationships may be represented by the following tree:

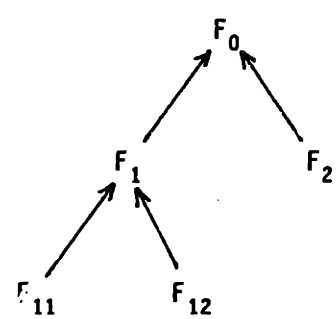


Figure 3-4:A tree of functional requirements.

Example 3. Suppose that a minicomputer M is both part of the inspection and packaging functions on a production line. Then the organization shown in Figure 3-5 is a network, which corresponds to a lattice structure. If two different computers M_1 and M_2 are used in the configuration, then the organization is a hierarchy as shown in Figure 3-6.

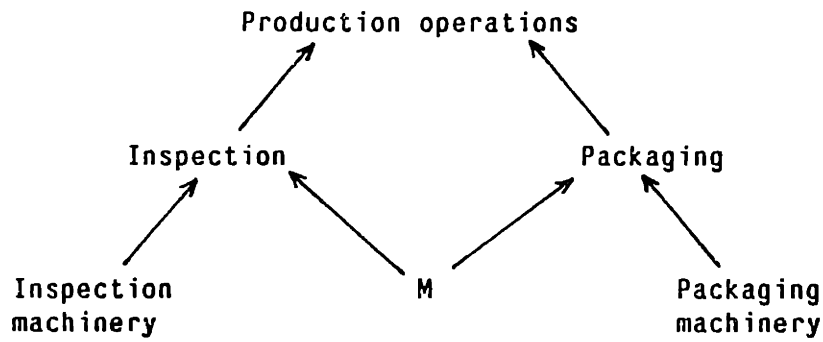


Figure 3-5:Example of a network organization.

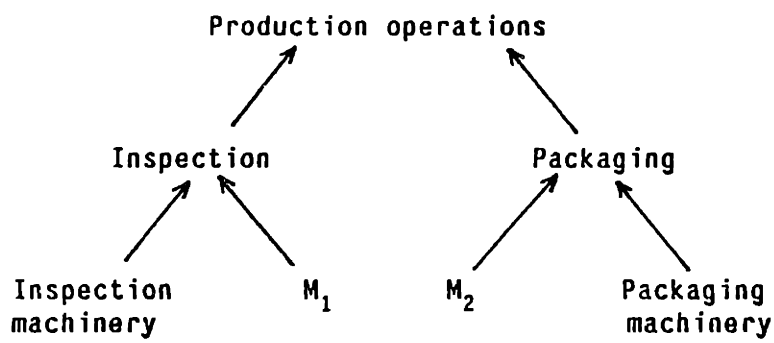


Figure 3-6:Example of a hierarchical organization.

3.2.2 Matrix Representation

Another way to represent system bondage is in terms of matrices. The two applications of matrices discussed in this subsection are connection and reachability matrices.

3.2.2.1 Connection Matrices

A system σ with m components and n kinds of connection can be represented as a set of n matrices $\{{}^1M,\ldots,{}^nM\}$. The $\langle i,j\rangle^{th}$ element of the kth matrix, ${}^kM_{ij}$, represents the strength of the kth type of connection imposed by component i on component j.

An obvious algebraic representation of a matrix is as a set of ordered triples $\langle i,j,M_{ij}\rangle$, where M_{ij} is the $\langle i,j\rangle^{th}$ element of the matrix M. If more than one matrix is to be represented, then the $\langle i,j\rangle^{th}$ element of the kth matrix may be encoded as the quadruple $\langle k,i,j,{}^kM_{ij}\rangle$.

If the strengths of the connections fall into a nominal classification given by the set of three values {"Inhibits", "Has no effect", "Stimulates"}, then the element ${}^{k}M_{ij}$ may be defined respectively as -1, 0, or 1. Hence the 3-component example shown in Figure 3-2 may be represented by the matrix

$$\begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{vmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \end{vmatrix} \end{array}$$

3.2.2.2 Reachability Matrices

Hierarchical relationships may also be represented by matrices. Given a particular hierarchy, we say that a vertex V_2 is *reachable* from vertex V_1 if there is a directed path from V_1 to V_2, or if the vertices are identical. The *reachability indicator* r(i,j) is a binary variable which is defined to equal 1 if vertex j is reachable from vertex i; otherwise it equals 0. A *reachability matrix* is a matrix whose elements are reachability indicators.

Example 3.

Consider the hierarchy in Example 2. Let the rows and columns of the reachability matrix R correspond to vertices F_0, F_1, F_2, F_{11} and F_{12}, respectively. Then R is given by:

$$\begin{array}{c} \\ F_0 \\ F_1 \\ F_2 \\ F_{11} \\ F_{12} \end{array} \begin{array}{c} \begin{array}{ccccc} F_0 & F_1 & F_2 & F_{11} & F_{12} \end{array} \\ \begin{vmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & 1 & & 1 & \\ 1 & 1 & & & 1 \end{vmatrix} \end{array}$$

where blank spaces denote 0's.

3.2.3 State Space Representation

Let K denote the set of all kinds of bonds that may hold among system components. The set K includes bond types such as electrical, mechanical, and informational.

We define $P = \{P_1, P_2, ...\}$ to be the collection of all properties that systems may exhibit. Examples of such properties are age, weight, color and location. Moreover, we take V_i to be the range of values assumable by property P_i. For example, if P_i = weight, then V_i is the half-closed interval $[0,\infty)$ in R.

Consider a system $\sigma \in \Sigma$. Let $\sigma|_k$ be the object obtained from σ when only connections of kind $k \in K$ are considered. Although σ is a system by construction, $\sigma|_k$ may or may not be a system. The latter object is only an aggregate when no bonds of kind k exist in σ; for example, two machining centers may have informational bonds but no mechanical bonds.

We take $\Sigma|_k$ to be the union of the $\sigma|_k$ taken over $k \in K$. Suppose that systems in $\Sigma|_k$ can be characterized by n properties $P_1,...,P_n$. Let X_i be a function that operates on $\Sigma|_k$ and yields V_i, the set of values taken by property P_i. Then the n-tuple ${}^kX = \langle X_1,...,X_n \rangle$ is the *(total) state function* for bonds of kind k, whose image is given by ${}^kV = V_1 x ... x V_n$. We will often write ${}^kX$ as simply X when no confusion results.

The state function ${}^kX$ depends not only on the kind of linkage k, but on other factors as well. For example, the position of an object depends on the reference frame f as well as time instant t.

Let F be the set of reference frames and T the collection of time instants. Then in general ${}^kX:D \rightarrow {}^kV$, where the domain is a multifold Cartesian product $D = K x \Sigma x F x T x ...$, whose coordinates depend on the particular model used.

If the range or codomain V_i for each property P_i is a subset of the real numbers R, then the *conceivable state space* of system structures of the kth kind, is denoted $S_c(k)$. However, since the properties of systems are often interdependent, not all combinations of R^n are usually possible. In such a case the set of permissible points in R^n, called the *permissible state space* $S_p(k)$ for systems of kind k, is a proper subset of the conceivable state space $S_c(k)$. In other words, the set of all feasible states for $\Sigma|_k$ is given by the range of X, and is denoted $S_p(k)$.

In kinematics, for example, the center of mass of any system may be specified by the 6-tuple $X = \langle x,y,z,x',y',z' \rangle$ where the first 3 variables denote positions along different axes, and the next 3 their

respective time derivatives. The conceivable state space is given by R^6; but the constraint, for example, that $[(x')^2 + (y')^2 + (z')^2]^{1/2}$ must be less than the speed of light implies that the allowable state space is a finite subset of R^6.

The state s of a system structure of kind k at d $\in$ D may be represented by the value of X at d. For the 2-dimensional case, $s = X(d) = \langle X_1(d), X_2(d)\rangle$.

An *event* is a disturbance in the values of the properties of a system. For a system description of kind k, an event may be represented as a triple $\langle s,s',e\rangle$ in the state space, where s and s' are members of $S_p(k)$, and e is a permissible map of $S_p(k)$ into itself. The figure below depicts an example of an event for the 2-dimensional case.

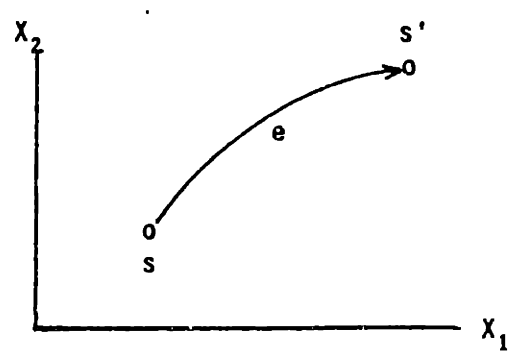


Figure 3-7:Example of an event in 2-dimensional space.

The *event space* of k, denoted $E_p(k)$, is the collection of all permissible events occurring in structures of type k.

The state function X often takes time T as an argument; that is, for t $\in$ T, we have X: {..,t,..} -> R^n. In this case we may speak of the *totality of processes* consisting of all the conditions assumed by properties of $\Sigma|_k$ during time interval $\tau \subseteq T$. For a system structure q of kind k, such processes may be represented by the values of the corresponding state function:

$$\pi(q,\tau) = \{X(t) \mid t \in \tau\}$$

The *history* of a time-dependent state function during interval $\tau \subseteq T$ is given by the collection of ordered pairs

$$h(x) = \{\langle t,X(t)\rangle \mid t \in \tau\}$$

Using these concepts, we define the *total action* or *effect* of an object a on object b in the following

way. Let h(a) denote the free history of a, when unaffected by object b. For the case of a 2-dimensional state space, h(a) might be given by the region A ∪ B in the figure below.

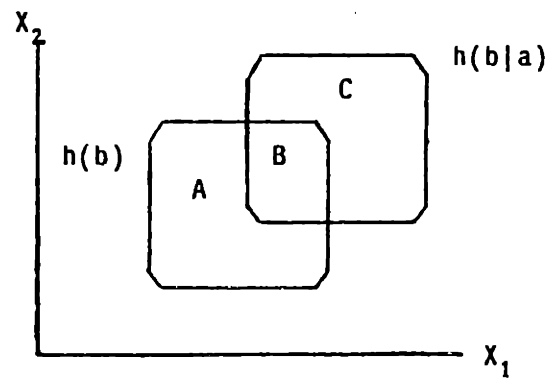


Figure 3-8: Area C represents the effect of object a on object b.

Suppose that the history of b under a's influence is given by h(b|a) = B ∪ C. Then the differential impact of a's action is seen to be C = h(b|a) - h(b).

We summarize the preceding discussion in the following definition.

Definition. A *state function* X is a mapping X: D -> $V_1 x V_2 x ... x V_n$ from a domain D into a range with n coordinates, each of which represents the value of a property. There is a state function X = ^kX for each kind k ∈ K of system structure.

1. The *totality of general properties* of system structures of kind k is representable by the set of all coordinates of X; that is, P(k) = $\{X_1,...,X_n\}$.

2. A *particular property* of a system of kind k may be represented by a value of a component of X; that is, by some $X_i(d)$ for d ∈ D.

3. The *state* of system structures of kind k at d ∈ D is given by the value of X at d; that is, s = X(d) = $\langle X_1(d),...,X_n(d)\rangle$.

4. The set of all such states of kind k, i.e. the range of X, is called the *permissible state space* of kind k, and is denoted $S_p(k)$.

5. Every *event* occurring in a system structure of kind k may be represented by a triple $\langle s,s',e\rangle$ where s, s' $\in S_p(k)$ and e is a permissible map of $S_p(k)$ into itself.

6. The *event space* of k, denoted $E_p(k)$, is the set of all permissible events occurring in system structures of kind k.

7. For a system with a particular environment and a given reference frame, the state function often takes the form of a time-dependent function $X : \{..,t,..\} \to R^n$ for $t \in T \subseteq R$.

8. If $X : \{..,t,..\} \to R^n$, then $\pi(q,\tau)$ denotes the totality of *processes* occurring in a system q of kind k during time interval $\tau \subseteq T$. This quantity may be represented by the set of states assumed by q during τ:

$$\pi(q,\tau) = \{X(t) \mid t \in \tau\}$$

9. The *history* of a system structure of kind k which is representable by a state function $X : \{..,t,..\} \to R^n$ during the time interval $\tau \subseteq T$, is given by the trajectory or locus of points

$$h(x) = \{ \langle t, X(t)\rangle \mid t \in \tau\}$$

10. The *total action* or *effect* of an object a on object b is given by the difference between the forced trajectory and the free trajectory of the patient b:

$$Ac(a,b) = h(b|a) - h(b)$$

3.2.4 Representation of System Characteristics

Subsections 3.2.1 and 3.2.2 presented the isomorphism or equivalence between graphs and matrices for representing system linkages. In view of this equivalence, this subsection discusses a number of system characteristics solely in terms of matrix notation.

3.2.4.1 Total Structure

The previous discussion has focused on the internal structure of a system. If a system is open--that is, interacts with its extended environment--then the representations may be easily augmented by

lumping the environment into a single object.

In the graph representation, we would add a new node called *Environment*. In matrix representation, we would add a new 0th row and column for each of the $k \in K$ kinds of bonds. If the environment acts on component i, then ${}^kM_{0i} \neq 0$; otherwise ${}^kM_{0i} = 0$. Similarly, M_{i0} is 0 or not depending on whether component i acts on the environment.

Let σ be a system with m components, n different kinds of connections, and an environment labeled collectively as 0. Then σ may be represented by n matrices of dimension $(m + 1)x(m + 1)$, such that:

1. The *internal connectivity* of σ in the kth aspect may be represented by deleting the 0th row and column of matrix ${}^kM$.

2. The *input* to σ in the kth aspect is representable by the row of input items of ${}^kM$:
$$ {}^k\mathrm{In}(\sigma) = \langle {}^kM_{01}, \ldots, {}^kM_{0m} \rangle $$

3. The *output* of σ in the kth aspect is representable by the column of output entries of ${}^kM$:
$$ {}^k\mathrm{Out}(\sigma) = \langle {}^kM_{10}, \ldots, {}^kM_{n0} \rangle^T $$
where the superscript T denotes the transpose of the corresponding tuple or vector.

4. The *behavior* or *observable performance* of σ *in the kth aspect* is the ordered pair
$$ {}^k\beta(\sigma) = \langle {}^k\mathrm{In}(\sigma), {}^k\mathrm{Out}(\sigma) \rangle $$

5. The *(total) behavior* of σ is the collection of its partial behaviors:
$$ \beta(\sigma) = \{ {}^k\beta(\sigma) \mid 1 \leq k \leq n \} $$

3.2.4.2 Decomposability

The matrix in the Example 3 of Subsection 3.2.2.2 is lower triangular. Such a matrix is called decoupled [Suh and Rinderle 1982] or decomposable [Courtois 1977; Ando et al. 1963: 4].

In general, let a matrix M be partitioned into submatrices S_{ij}:

$$\begin{vmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{vmatrix}$$

Suppose that the diagonal components S_{ii} are nonzero while all the off-diagonal components are submatrices containing only 0's. Then we say that M is *block diagonal* and the corresponding system is *completely decomposable.*

Suppose that the northeast components S_{ij} for $i < j$ are all 0's. Then we say that M is *block triangular* and the corresponding system is *decomposable.*

By appropriate ordering of the columns (and thereby rows) corresponding to the set of vertices in the hierarchy, it is possible to transform the reachability matrix into decomposable matrices. In fact, Warfield [1974] describes a procedure for obtaining such an ordering.

When the non-diagonal components S_{ij} are small but nonzero, the corresponding matrices are called *nearly decomposable* or *nearly completely decomposable* [Courtois 1977]. If a nearly completely decomposable matrix describes a system of simultaneous equations, the system may be represented as a superposition of (1) a set of independent subsystems, one for each diagonal submatrix S_{ii}, and (2) an aggregate system having one variable for each system [Simon and Ando 1963].

3.3 System Models

This Section presents a typology of system models and shows how they may be specified in algebraic terms. The typology is a 3-way classification consisting of black, grey and beige boxes.

A black box model considers only the relationships between inputs and outputs. A grey box allows for internal states as well, but considers them as hypothetical constructs. A beige box not only considers internal states but lends a factual interpretation to the intermediate variables.

All three types of models focus on the kinds of bonds among system components, and not necessarily on the nature of the components themselves. Hence they are applicable to a great variety of systems.

3.3.1 Black Box Models

Black box models, also known as input-output models, ignore the internal structure of their referent systems. The simplest black box is one that does not interact with the environment; an example is found in the free particle in motion.

A slightly richer model is that of a *monopole*. This is a black box with one input or one output port, depending on whether the system is regarded as active or passive.

The full-fledged black box is one that has both input and output terminals. The simplest such model is a *dipole*, which has exactly one input and one output terminal and therefore has limited capacity for stimulation and response. A black box with more terminals is called a *multipole*.

Let U and V denote respectively the sets of input and output values of a dipole. Then the formal representation of the dipole is UxV; that is, the pairs of elements $\langle u,v \rangle$ for $u \in U$ and $v \in V$. This corresponds to a binary relation f, which may or may not represent a function. The associated graph or extension is $E(f) \subset UxV$.

> **Example.** A simple example of a dipole is one with binary inputs and outputs: $U = V = \{0,1\}$. The transfer function is a mapping $f:\{0,1\} \rightarrow \{0,1\}$. A switching function defined by $f(0) = 1$ and $f(1) = 0$ may be represented in set notation as $\{\langle 0,1 \rangle, \langle 1,0 \rangle\}$.

The inputs and/or outputs may also be continuous rather than discrete.

> **Example.** A differentiator is a system whose state at time $t \in T$ is defined by $\langle u(t),u'(t) \rangle$, where $u:T \rightarrow \mathbf{R}$ and u' is the time derivative of u. The state space is then given by $S = UxU'$ where U is the domain of u and U' the domain of u'.

Despite their prevalence and usefulness, black box models may be too lean to model complex systems adequately. In this case they may be superceded by richer models; for example, statistical mechanics provides a more accurate description of the behavior of gas molecules than does thermodynamics.

A system, however, can often be modeled as a set of interconnected black boxes, each of which represents a subsystem or component. But the resulting model is no longer a black box. Since the inputs and outputs to the component boxes are modeled explicitly, the resulting model is a beige box.

3.3.2 Grey Box Models

A grey box is a black box whose internal states may be conjectured but not observed. The set of internal states is a hypothetical intermediary between inputs and outputs; that is, the intermediate link is a formal construct rather than a structural variable. Grey box models are generalizations of black box models. For example, they can model the system which produces different outputs in response to the same input, depending on the internal state; a black box model would fail in this case.

Two classes of grey box models with wide applicability are those of automata theory and information channels.

3.3.2.1 Automata Theory

Many systems, whether of the hardware or software variety, alternate among a finite number of conditions or states. The transition from one state to another (which might be the same state), may occur deterministically or stochastically. Moreover, the interval between any two transitions may also be fixed (deterministic) or variable (random).

Automata theory investigates fundamental principles shared by artifacts such as computers and control systems, as well as natural systems such as self-replicating cells and the human nervous system [Burks 1970; von Neumann 1966; etc.]. These principles relate to the structure, organization, control and programming of systems.

In this goal it is similar to cybernetics [Wiener 1961] and its derivative fields such as linear and nonlinear control theory. The two approaches differ primarily in emphasis and methodology. Cybernetics highlights the importance of feedback loops for control and for implementing purposeful behavior. Moreover, it relies primarily on the tools of continuous mathematics such as differential equations. Cybernetics and its derivative theories are exemplars of analytic theories. Any implications for design are simply by-products of the results of analysis.

In automata theory, feedback loops appear only implicitly in the form of cooperating processes at the hardware level and conditional braching at the software level. The theory relies primarily on discrete mathematics such as symbolic logic and set algebra. Since automata theory considers programming as a central concern, at least one aspect of the theory relates directly to system design.

Automata theory addresses the nature of information-processing devices in relation to their

interactions with the environment. The theory ignores peripheral issues such as the materials of fabrication. Hence any system which processes information--be it mechanical, biological, or social--may be regarded as an automaton.

An *automaton* is a system which accepts inputs and produces outputs in the form of symbols [Ginzberg 1962; Hopcroft and Ullman 1979; Lewis and Papadimitriou 1981]. The system consists of a single component which moves from one state to another based on input stimulii. Any input or output, as well as the set of states, is assumed to be denumerable. Hence the canonical model of an automaton is discrete and sequential.

A *deterministic* automaton is one whose transition from one state to another is fixed for each state and input symbol. A *probabilistic* automaton, on the other hand, is one whose state transitions are specified in probabilistic terms: a machine in one state may jump to any of one or more states in response to a given input. In addition, a *nondeterministic* automaton is a probabilistic automaton whose transition probabilities are not specified: each transition is simply regarded as "possible" or "impossible".

Deterministic automata are of primary interest in technical systems, such as those in industrial production systems, where a high level of reliability is demanded. Probabilistic automata are generally appropriate for biological and social systems which seem to be inherently stochastic, whether by design or lack of understanding of the underlying processes.

The ensuing discussion first treats deterministic automata, then generalizes the model to include probabilistic automata.

Deterministic Automata

We begin by introducing some definitions relating to operations on symbols.

Definition. Let $\star$ denote a relation on a set S, whose members we call *symbols*.

1. $\star$ is called a *concatenation* iff it forms compound symbols based on S. That is (a) $\star$ maps $S^m x S^n$ into S^{m+n} for $m,n \geq 1$, and (b) for all $u \in S^m$ and $v \in S^n$, $uv \in S^{m+n}$.

2. S^* is called the *Kleene star* operator of S iff it contains all finite concatenations based on S. That is, S^* maps S into S^n for $1 \leq n < \infty$.

Symbol	Mathematical nature	Interpretation
S	Set	State space
Σ	Set	Set of unit inputs
Ω	Set	Set of unit outputs
$\star$	Operation	Concatenation
M	Function	Transition function
N	Function	Output function
s_0	Element	Initial state
F	Set	Set of final states
σ_0	Element	Null input
ω_0	Element	Null output

Figure 3-9:Primitive Objects of Automata Theory.

The primitive concepts of automata theory are given in the following table. These concepts are discussed further and tied together in the next definition.

Definition. The structure $A = \langle S,\Sigma,\Omega,\star,M,N,s_0,F,\sigma_0,\omega_0\rangle$ defines a *finite deterministic automaton* A iff:

1. S, called the *state space* of A, is a finite nonempty set with $n > 1$ elements. Every element of S represents an internal state of A.

2. $\Sigma = \{\sigma_0,\sigma_1,\ldots,\sigma_k\}$, called the *input alphabet*, is a nonempty set of elements called *letters*. Each letter represents a unit input into A from its environment.

3. $\Omega = \{\omega_0,\omega_1,\ldots,\omega_k\}$, called the *output alphabet*, is a set of elements called *letters*. Each letter represents a unit output of A to its environment.

4. $\star$ is the *concatenation operator* which forms compound inputs or outputs from unit letters in Σ or in Ω. For example, $\sigma\star\sigma'$ is a compound input for any unit inputs σ and σ' in Σ. Where no confusion results, $\sigma\star\sigma'$ will be written more simply as $\sigma\sigma'$. A finite concatenation of elements in Σ or in Ω is called a *tape* or

word.

5. The element $\sigma_0 \in \Sigma$ is such that for every tape $x \in \Sigma^*$, the following condition holds: $\sigma_0 x = x\sigma_0 = x$. Similarly, $\omega_0 \in \Omega$ is such that, for every $y \in \Omega^*$, we have $\omega_0 y = y\omega_0 = y$.

6. The function $M: S \times \Sigma^* \rightarrow S$ is called the *transition* (or *next state*) function. If $s \in S$ and $x \in \Sigma^*$, then $M(s,x) \in S$ denotes the state that the automaton A goes to when A, while in state x, receives input x.

7. The function $N: S \times \Sigma^* \rightarrow \Omega^*$ is called the *output function* and is characterized in the following way. $N(f,x) = \omega_j \in \Omega^*$ for every $f \in F \subset S$ and every $x \in \Sigma^*$, where $N(s,x)$ denotes the output of A when it receives input $x \in \Sigma^*$ while in state s. (The interpretation is that A outputs some value ω_j when it enters one of the final states of the set F.)

8. S_0 is an element of S denoting the initial internal state of A.

9. F is a nonempty subset of the state space S; every member f in F represents a final state of A.

10. A may not undergo spontaneous transitions; that is, for each s in S, $M(s,\sigma_0) = s$. Thus the null input has no effect.

11. The internal states form sequences. That is, the effect of a compound input xy, where $x,y \in \Sigma^*$, equals the effect of the second input acting on the automaton to which the first input carried it; that is, $M(s,xy) = M(M(s,x),y)$.

This definition assures that a deterministic automaton is activated solely be external stimulii and proceeds in a sequential, deterministic way. For example, the automaton is allowed to make no spontaneous transitions based on the null input. Relaxation of this restriction would give rise to a nondeterministic automaton, which will be discussed later.

We note that the output set Ω may be empty; more on this in the subsequent discussion. In the case

where $\Omega = \{0,1\}$, $\omega_0 ::= 0$ may be interpreted as the null output (i.e. absence of an output), and 1 as a unary output. Since a string of 1's of length n may represent the natural number n, the output repertoire is still countably infinite.

The relationships among the various objects may be depicted as shown in Figure 3-10.

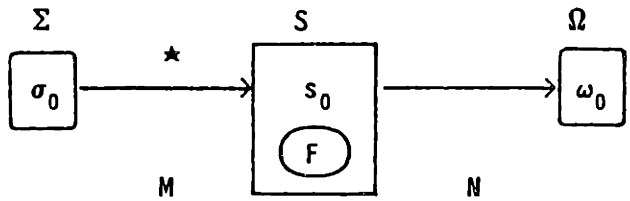


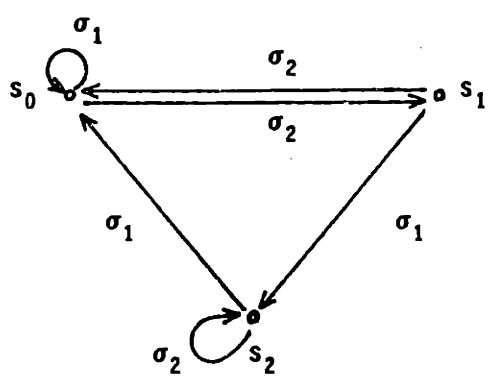
Figure 3-10:Coordinates of a deterministic automaton.

A number of these ideas are illustrated in the following example.

. Example. Suppose that the transition function M is represented by the matrix below,

$$\begin{array}{c|cc|} & \sigma_1 & \sigma_2 \\ s_0 & s_0 & s_1 \\ s_1 & s_2 & s_0 \\ s_2 & s_0 & s_2 \end{array}$$

where the $\langle i,j \rangle^{th}$ element stands for $M(s_i,\sigma_j)$. The transitions may also be represented as a graph:



To facilitate the discussion of the properties of deterministic automata, we introduce another definition.

Definition. Let $A = \langle S,\Sigma,\Omega,\star,M,N,s_0,F,\sigma_0,\omega_0 \rangle$ denote a finite automaton A.

1. The restriction of the transition function M to s_0 is called the *response function*; that is, $resp(x) = M(s_0,x)$, for $x \in \Sigma^*$.

2. A state s in S is *accessible* iff there exists some input x in Σ^* such that $s = resp(x)$; otherwise s is *inaccessible*.

3. A tape $x \in \Sigma^*$ is *accepted* (or *recognized*) by A iff $resp(x) \in F$; that is, s produces an output by taking A from the initial state to a final state.

4. The *behavior* of A is the set of all tapes accepted by A:
$$\beta(A) = \{x \in \Sigma^* \mid resp(x) \in F\}$$

5. Two automata A and A' are *behaviorally equivalent* iff $\beta(A) = \beta(A')$.

Some Special Deterministic Automata

The general model of a deterministic automaton may be specialized to yield restricted types of automata. In particular, a recognizer or Rabin-Scott automaton is one in which the output set Ω and output function N are undefined [Ginzberg 1968: 55].

Definition. A *recognizer* Re is the automaton represented by the structure $\boldsymbol{Re} = \langle S,\Sigma,\star,M,s_0,F,\sigma_0\rangle$, in which the coordinates are as defined previously.

A recognizer goes into a final state under appropriate input but does not tell the world of its findings.

A semiautomaton is a Rabin-Scott automaton in which the final state set F is unspecified.

Definition. A *semiautomaton* Se is the automaton specified by the structure $\boldsymbol{Se} = \langle S,\Sigma,\star,M,s_0,\sigma_0\rangle$, whose coordinates are as defined previously.

This is the automaton-analogue of a "dumb" machine which changes states upon receiving input but does not recognize any input string as being special.

3.3.2.2 Probabilistic Automata

A probabilistic automaton may be defined as follows.

Definition. The structure $\boldsymbol{P} = \langle S,\Sigma,\Omega,\star,M,N,s_0,F,\sigma_0,\omega_0\rangle$ represents a *finite probabilistic automaton* P iff all the coordinates of $\boldsymbol{P}$ except M satisfy the conditions in the

definition given for finite deterministic automata; M is now defined as a function $M: S \times \Sigma \to U^{n+1}$ where U is the real interval [0,1]. The range in U^{n+1} is called the table of *transition probabilities* such that for each state $s \in S$ and letter $\sigma \in \Sigma$,

$$M(s,\sigma) = \langle p_0(s,\sigma), p_1(s,\sigma), \ldots, p_n(s,\sigma) \rangle$$

where

1. $p_i(s,\sigma)$ for $i = 0,1,\ldots,n$ gives the likelihood that the automaton, when it receives a unit stimulus σ while in state s, will jump to state $s_i \in S$.

2. The $p_i(s,\sigma)$ for $i = 0,1,\ldots,n$ are probabilities subject to the constraint
$$\Sigma_{i=0,\ldots,n}\ p_i(s,\sigma) = 1$$
for each s in S and σ in Σ.

This definition only deals with unit inputs or letters. To handle tapes or words, we introduce the next definition.

Definition. The *stochastic matrix* associated with unit input σ in Σ is given by
$$P(\sigma) = [p_j(s_i,\sigma)] \text{ for } i,j = 0,1,\ldots,n$$

The key result pertaining to the operation of a probabilistic automaton lies in the statistical independence of the effect of tapes. In other words, the stochastic matrix of an input $x = \sigma_p\sigma_q\ldots\sigma_r$ equals the product of the stochastic matrices of the unit components of the tape:
$$P(\sigma_p\sigma_q\ldots\sigma_r) = [p_j(s_i,\sigma_p\sigma_q\ldots\sigma_r)] = P(\sigma_p)*P(\sigma_q)*\ldots*P(\sigma_r)$$

3.3.3 Beige Box Models

A beige box, also called a dynamical model, is a grey box in which the intermediate links or variables are explicitly modeled rather than merely hypothesized. The intermediate link is regarded as representing a structural mechanism which is actually responsible for transforming inputs to outputs.

One class of dynamical models are field models such as those in electromagnetism; here the electric field generates the magnetic field, which in turn propagates the electric field. Another example is found in population models: an increase in the number of prey induces a growth in the predator population which decimates the prey, and so on recursively.

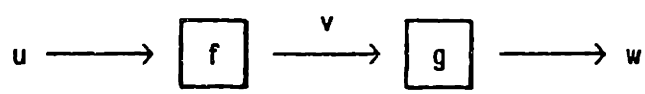
Beige boxes occur any time two or more black or grey boxes are interconnected to form a richer

model.

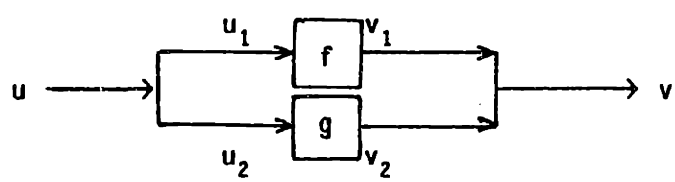
3.3.3.1 Linkages among Black Boxes

The standard types of interconnection among black or grey boxes are as follows.

1. *Series.* The output of one box serves as input into the next. If the respective transfer functions are given by f: U -> V and g: V -> W, then the overall transfer function is given by the composition h ::= f★g: U -> V. The output values are given by w ::= g(f(u)) for any input u in U. This relationship holds whether the inputs and outputs are signals or material objects. An example is a production plant g which assembles goods manufactured by plant f.



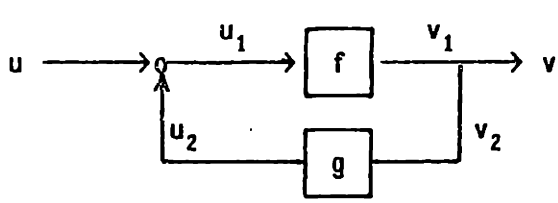
2. *Parallel.* In this configuration, two subsystems share the inputs and contribute jointly to the output.



   a. If the input u is a physical object such as a batch of chemicals, then $u = u_1 + u_2$ and $v = f(u_1) + g(u_2)$.

   b. If u is a signal such as a message, then $u = u_1 = u_2$ and $v = f(u_1) + g(u_2)$.

3. *Feedback.* In this setup, part of the output is fed back as input to the system.



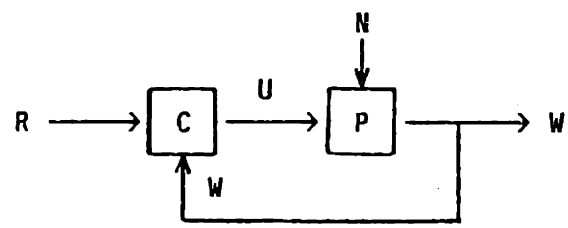
   a. If the input u is a physical object, then $u = u_1 + u_2$ and $v = v_1 - v_2$, where $v_1 = f(u_1)$ and $u_2 = g(v_2)$.

b. If u is a signal, then $u_1 = u + u_2$. But now $v = v_1 = v_2$, , implying $u_2 = g(v)$.

The use of series, parallel and feedback couplings allows for the development of richer models. The overall model is of the input-output type, but is no longer a black or beige box since intermediate variables among subsystems have been identified and modeled explicitly. Moreover, the resulting system may possess emergent properties such as instability which were unknown in the behaviors of the components.

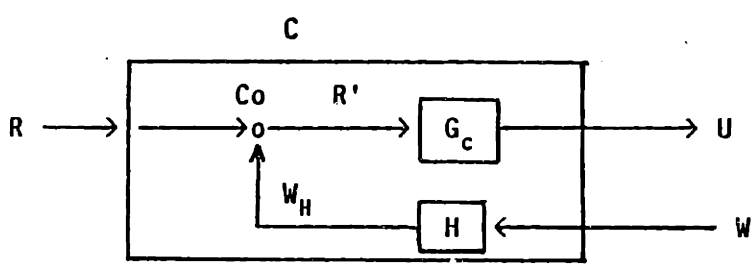
3.3.3.2 Control Systems

A control system S consists of two subsystems: a controlled system or plant P and a controller C. These components are connected as shown in the figure below,



where P takes input U and yields output W in the face of environmental noise N. The role of the controller C is to issue a controlling signal U to P in order to keep W close to some reference signal R.

The controller C in turn consists of the following components: a moderator H which transforms



the system output W into the filtered signal W_H, a comparator Co to determine the deviation R' of W_H from the reference R, and a control signal generator G_c which operates on R' to produce the signal U.

The total state function of the control system S is given by Z = XxU, where $X = \langle x_1,...,x_n \rangle$ is the vector of state variables and $U = \langle u_1,...,u_n \rangle$ the control variables. This function represents a mapping from the set of time instants T to a vector of real numbers; that is, Z: T -> R^{n+m}. Z is

subject to the following constraints:

1. State constraints on the state variables X. The values of the x_i must lie within certain bounds specified by the application.

2. Control conditions which stipulate how the control vector U affects the state vector X as well as itself; that is,

$$dx_i/dt = g_i(Z,t)$$

for i = 1,...,n.

3. Optimization criteria for certain state variables, depending on the application.

Examples of optimization criteria are found in the minimization of fuel consumption or of mean squared error from a reference signal.

3.3.3.3 Information Transmission System

Another example of a grey box model is the information transmission system [Gallager 1968; Shannon and Weaver 1949]. This subsection develops an algebraic model for an information channel by drawing on the deterministic and probabilistic automata models of Subsection 3.3.2.1. Then it interprets the classical information measure in terms of this model.

Algebraic Model

An information transmission system consists of three subsystems: a source, a channel and a destination. Let the *source* So be a probabilistic automaton having internal states $S = \{s_0, s_1, ..., s_m\}$. The state s_0 is the initial state, and all other states comprise the set of final states, $F_{So} = \{s_1, ..., s_m\}$.

The output alphabet $\Omega ::= U = \{u_1, ..., u_m\}$ is also called the set of *messages* or *symbols* of the transmission channel. The input alphabet Σ is empty, and the "table" of transition probabilities takes the form of a vector, $M(s_0) = \langle p_1(s_1), ..., p_m(s_m)\rangle$. In other words, So jumps spontaneously from the initial state s_0 to one of the final states s_i with probability p_i. When the system enters state s_i, it produces the output u_i.

The *channel* Ch is a probabilistic automaton which moves from the initial state c_0 to one of the final states $F_{Ch} = \{c_1, ..., c_m\}$. The probability of jumping from c_0 to a final state c_i depends on the inputs $U = \{u_1, ..., u_m\}$ and another alphabet $No = \{No_1, ..., No_L\}$. The set No is interpreted as *noise.*

The output alphabet of the channel is $V = \{v_1,\ldots,v_n\}$. The channel outputs message v_j when it enters state c_j.

Since the input to Ch is given by UxNo, the table of transition probabilities is given by[5]

$$M_{Ch}(c_0,u_i,No_j) = \langle p_1(c_0,u_i,No_j),\ldots,p_m(c_0,u_i,No_j)\rangle$$

The *destination* De is a deterministic automaton with input V and internal states $\{d_0,d_1,\ldots,d_m\}$. The automation jumps from its initial state d_0 to a final state d_i, for $i = 1,\ldots,m$, upon receipt of input v_i.

A schematic diagram is given below for the information transmission channel having $m = 2$ messages.

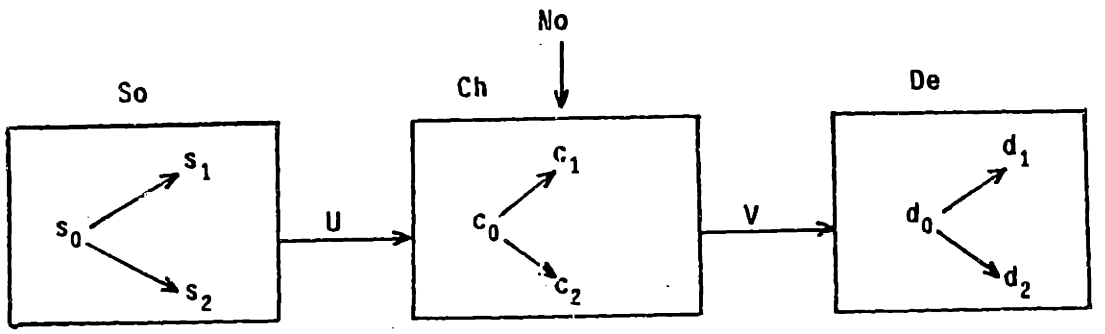


Figure 3-11: Information transmission channel for the case of $m = 2$ messages.

We summarize these ideas as follows.

Definition. The structure I_c = ⟨So,Ch,De,U,V,No⟩ represents an *information transmission system* I_c iff

1. So, called the *source*, is a probabilistic automaton with initial state s_0, final states $\{s_1,\ldots,s_m\}$, and output set U.

2. De, called the *destination*, is a deterministic automaton with initial state d_0, final states $\{d_1,\ldots,d_m\}$, and input set V.

[5] We require Ch to be a probabilistic automaton in order to account for its unreliable nature: this incorporates the phenomenon of *equivocation* or information loss in the channel. In other words, suppose that noise were the only complication for the channel subsystem. Then we could assume that the noise alphabet $\{No_i\}$ has a probability distribution $\{p_i\}$, but Ch itself is a deterministic machine which changes states reliably on input U x No. But such a deterministic machine would not allow for equivocation. Hence we require Ch to be a probabilistic machine.

3. No, called *noise*, is a set of symbols.

4. $U = \{u_1,...,u_m\}$ and $V = \{v_1,...,v_m\}$ are sets of symbols called *messages*.

5. Ch, called the *channel*, is a probabilistic automaton that takes UxNo as input and yields V as output.

Classical Information Measure

We now turn to the task of defining an information metric on this system. In the absence of any specific knowledge of the state of s_0, an observer at the destination De can only assume that s_0 is in a final state s_i (for $i = 1,...,m$) with probability $p_i ::= p(s_i)$; this is called the *prior* probability of being in state i. When De receives message v_i and enters state d_i, the observer can feel more confident that the source So is actually in state s_i. The *posterior* probability that So is in state s_i is then given by

$$p_i' ::= p'(s_i|v_i) = \frac{p(s_i)*p(v_i|s_i)}{\Sigma_{j=1,...,m}\, p(s_j)*p(v_i|s_j)}$$

where $p(s_i)$ is the prior probability of So being in state s_i, and $p(v_i|s_i)$ is the probability that Ch outputs v_i on input u_i, which is obtained by summing over all values of No_k in No. Note that $p(v_i|s_i) = p(v_i|u_i)$, since the source So outputs u_i with probability 1 when in state s_i.

We define the information content of a message as follows.

Definition. Let $p_i ::= p(s_i)$ be the prior probability that source So is in state s_i (for $i = 1,...,m$), as estimated by an observer at destination De in an information transmission system. Similarly let p_i' denote the posterior probability that So is in s_i, after receipt of a message v_i at De. Then

1. The *information content* or *information measure* of message v_i with respect to state s_i is given by

$$I(v_i,s_i) ::= lb(p_i'/p_i)$$

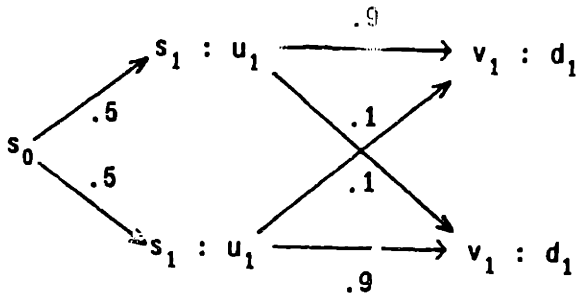
where lb denotes the binary log (logarithm to base 2).

2. The *average information content of the channel* or the *message stream* is given by

$$I_{ave} ::= \Sigma_{i=1,...,m}\, I(v_i,s_i)$$

The most common units of I are *bits*, or *b*inary dig*its*. However, the natural logarithm is also used from time to time; in this case the units of I are *nits*, or *n*atural dig*its*. The information metric yields positive values if we make the mild assumption that the transition table for the channel Ch is at least partially reliable; that is, Ch is more likely to output message v_i upon receipt of input u_i, than any other message v_j.

We illustrate these ideas by an example. Let the number of messages be m = 2, and the noise be a singleton, i.e. No = $\{\nu\}$. Moreover, the table of transition probilities for Ch may be depicted as



Then $p(s_1) = p(s_2) = .5$ and

$$p'(s_i|v_i) = .5*.9 / (.5*.9 + .5*.1) = .9$$

for i = 1,2. Hence $I(v_i,s_i) = lb(.9/.5) = .848$ bits for each i.

* * *

In summary, a typology of system models has been presented according to three categories in terms of black, grey and beige boxes. Black box models are the simplest of the three, but may be inadequate for modeling many systems. In this case grey box models such as those of automata theory, or even beige box models such as those of control theory, may be required.

Chapter Four

Implications for Analysis Phase

This chapter demonstrates how the infrastructure of Chapter 3 may be applied to the analysis phase of manufacturing systems. It begins with a framework for manufacturing systems based on the systems viewpoint, then turns to a discussion of the nature of complexity and its usual manifestation in hierarchical form. The informal notion of complexity must ultimately be interpreted quantitatively in terms of measures of information. A general framework for information is presented, followed by a discussion of information characteristics in terms of effectiveness, efficiency, and timeliness.

Since the most important component of a manufacturing system may be information, it is imperative to model this factor adequately. This may be done by treating an industrial plant as an automaton: in this way, the investigation may draw on the techniques and results of automata theory.

Moreover, the structures described in this chapter are fundamental to the study of manufacturing systems in relation to the Design Axioms. Hence this chapter and the next serve as the bases for the integration of analysis and synthesis phases presented in Chapter 6.

4.1 Framework for Manufacturing Systems

This section begins with a discussion of the types of work in terms of labor, culture and management activities [Bunge 1979], then continues with the specification of a manufacturing system using the systems framework.

4.1.1 Types of Work

A system is considered to perform an activity if it is a control system characterized by a definite goal. The goal may be self-appointed or imposed by the environment, such as a superior.

Definition. A system performs an *activity* iff it executes a sequence of goal-directed actions.

Work is considered to be a useful type of activity. An example of a control system relating to work is a laborer or robot with its tools.

Definition. Let x be a system, y an object, and z a person which may or may not equal x. Then x *works* iff

1. x exerts an activity on y--other than consuming y--and thereby produces an appreciable change in the state of y.

2. The change in the state of y, produced by x, is of value to z.

There are three types of work: labor L, cultural work K, and management M. The goal of labor is to transform nonhuman things into more valuable objects; that of cultural work is to alter feelings or mental states; that of management is to direct labor or cultural work. The notion of labor or primary work may be specified as follows.

Definition. Let σ be a society with composition Co, environment En, and structure St $= S \cup T$, where S is the collection of social and kinship relations and T is the collection of transformation relations in σ. Moreover let L be a subset of transformation relations in σ. Then

1. The member $L_i \in L$ is called the ith kind of *labor*, or *primary work*, performed by members of σ, iff (a) L_i is an activity performed on objects, and (b) the outcome of the activity is a new set of objects or a change in the states of the objects, which is of value to members of σ.

2. The domain of L_i is $[Co]^p x[En]^q$ for $p, q \geq 1$. The codomain $A_i \subset En$ of L_i is the *product* of the ith labor L_i, and each element of A_i is called an *artifact*.

3. The totality of all the products (goods and services) resulting from the union of all types of labor in σ, denoted A, is called the *material production* of σ.

4. The subset ${}^1P_i \subset Co$ occurring in the domain of the labor relation L_i is called the set of *primary producers* of A_i, or the ith *labor force* of σ.

5. The union ${}^1P = \cup_i {}^1P_i$ of primary producers of σ is called the *labor force* or *primary workforce* of σ.

Cultural or secondary work pertains to esthetic or intellectual work. It is the work performed by societal members such as engineers, scientists, writers, and actors.

Definition. Let σ be a society with composition Co and social relations in S, while $K \subset St$ is a proper subset of the social structure of σ. Then

1. The member K_j of K is called the jth kind of *cultural* or *secondary* work performed by members of σ iff K_j is an activity (a) capable of influencing moods or thoughts, or (b) supplying ideas which intervene in the primary production of σ.

2. The subset ${}^2P_j \subset Co$ occurring in the domain of the cultural work relation K_j, is called the jth set of *cultural* or *secondary* producers of σ.

3. The union ${}^2P = \cup_j {}^2P_j$ is called the *cultural* or *secondary* workforce of σ.

The next definition specifies the nature of managerial or tertiary work.

Definition. Let $M \subset S$ be a subset of the social relations in a human society σ. Then

1. The member M_k of M is called the kth kind of *managerial* (or *organizing* or *tertiary*) *work* performed by members of σ iff M_k is an activity contributing to or controlling some primary or secondary work undertaken in σ.

2. The subset ${}^3P_k \subset Co$ found in the domain of the managerial relation M_k, is called the kth set of *tertiary* (or *managerial* or *organizing*) *workers* of σ.

3. The union ${}^3P = \cup_k {}^3P_k$ is called the *managerial* or *tertiary workforce* of σ.

4. The subset ${}^3P_L$ of ${}^3P$ dedicated to the control of primary production is called the *production management* of σ, and its complement ${}^3P_C = {}^3P - {}^3P_L$ the *culture management* of σ.

According to this definition, executives, judges, planners and politicians--as well as other makers and

conveyors of decisions--comprise the managerial workforce. Much of the work relating to industrial production, whether performed by humans or robots, involves primary workers and production management.

Production relations between two workers may be based on coproduction, management, or comanagement.

Definition. Two individuals, whether human or machine, hold a *production link* or *bond* iff any one of the following holds:

1. Both individuals take part in producing a set of products (goods or services) either as primary (L) or as secondary (K) workers.

2. One of the individuals produces material or cultural goods while the other manages (or controls or oversees) him.

3. The two individuals comanage at least one producing worker.

4.1.2 Systems Framework Applied to Manufacturing Plants

This subsection presents a systems interpretation of a manufacturing system. But we first introduce the notion of a technosystem.

Definition. A system τ is a *technosystem* iff

1. The composition of τ includes humans and artifacts.

2. The environment of τ includes components of a society.

3. The structure of τ includes production, maintenance, or use of artifacts.

For example, the production and distribution of aircraft constitutes a technosystem.

A manufacturing system may be defined as a special subsystem of a technosystem.

Definition. The structure $M = \langle Co, En, St \rangle$ represents a manufacturing plant μ iff μ is a subsystem of a technosystem whose structure St includes primary work relations pertaining to production and to production management.

The coordinates of M may be described as follows. The composition is given by Co = {laborers, robots, engineers, managers, janitors, secretaries,...}. The (immediate) environment is En = {trucks, warehouses, telecommunication lines, families of workers, safety inspectors, corporate headquarters,...}. The structure is given by St = {relations of supervising, fabricating workpieces, ordering supplies, stockpiling inventories, shipping orders, maintaining cleanliness, handling grievances, receiving visitors, informing headquarteres,...}.

The composition of a manufacturing system contains elements which may or may not constitute systems. For example, the laborers working independently do not constitute a system; when organized for a strike, however, the same people constitute a social system.

Manufacturing plants have features which are characteristic unto themselves.

Definition. Let P be a property. Then

1. P is a *plant property* iff there exists, at any point in time, some manufacturing plant possessing P.

2. If P is a plant property of some plant μ, then P is a *resultant* property of μ iff it is also a property of some components of μ. Otherwise P is called an *emergent* property.

For example, the total labor expense per year is a resultant property of a plant, since it can be obtained by summing over all the individuals' wages. In contrast, the operating efficiency of the plant or its pollution output are examples of emergent properties, since they cannot be obtained by such simple summation.

4.2 Nature of Complexity

A *complex* system might be described as "one made up of a large number of parts that interact in a nonsimple way" [Simon 1969: 86]. In a complex system, the whole is more than the sum of its parts in that it is a nontrivial matter to deduce the properties of the entire system from those of its component parts.

A system may be defined by state or process descriptions. *State* descriptions portray the world as sensed; among such vehicles are pictures, blueprints and diagrams. In contrast *process* descriptions portray the world as acted upon; examples of such vehicles are recipes or differential equations which

show how the system may be reconstructed [Simon 1969: 111]. In axiomatics, design variables are analogous to descriptions of state, and process variables to descriptions of process.

Complex structures often take the form of hierarchies. In their structure as well as their dynamic behavior, complex systems often can be partitioned into modules which depend only weakly on one another. The correlation matrix showing the interconnections or interactions among the subsystems, usually exhibits regions whose elements are zeros or very small numbers. Such near-decomposability into separate submatrices highlights the simplicity of the system structure and behavior. Such matrices may in fact be used to obtain quantitative measures of subsystem dependencies.

When all the subsystems interact strongly with one another, no partitioning is feasible. Hence hierarchical decomposition only makes sense in a near-decomposable system.

4.2.1 Complexity in Numbers

Complexity often arises when the number of components increases, due to an increase in the number of system bonds. Such numerically-induced complexity may be classified into 3 nominal categories [Weaver 1948]:

1. *Few variables: problems of simplicity.* Before 1900, physical scientists made great strides in the treatment of phenomena involving only a few variables, usually two. An example lies in the description of a particle's trajectory subject to external forces.

2. *Moderate number of variables: problems of interdependent or organized complexity.* This situation occurs in complex phenomena characterized by some degree of organization. Questions in this category include, "What affects the price of gold?" and "How do you motivate workers?".

3. *Many independent variables: problems of independent or disorganized complexity.* In this situation, the law of large numbers may be brought into play. Examples lie in calculations of the average pressure exerted by a vial of gas, of premiums for life insurance, and of sampling statistics in industrial production.

The second type of problem still eludes us, as our tools and procedures are inadequate when dealing with problems of dependent complexity.

Typology of Hierarchies

The architecture of a manufacturing system may be classified into logical and physical components. The physical hierarchy, for example, may consist of levels such as these:

1. Facility: entire manufacturing plant.

2. Shop: e.g., molding, machining, or assembly.

3. Physical cell: cluster of machinery.

4. Workstation: e.g., milling, polishing, or inspection station.

5. Device: e.g., robot, conveyor belt, or inspection unit.

The logical hierarchy might consist of these levels:

1. Facility

2. Shop

3. Logical cell: production capabilities which may be represented by a set of machines dispersed in space.

4. Operation: e.g., machining, orientation, transportation, or measurement.

There are obviously close parallels between the physical and logical hierarchies. In general, however, there is no simple mapping between these two structures. In database management systems, for example, it is often important to separate the logical and physical aspects of the database architecture.

The logical hierarchy is itself a collection of different structures. We may identify a functional hierarchy pertaining to functions or production capabilities, as well as an informational hierarchy depicting the flows of information within and among different nodes in the structure.

It is an intuitively appealing concept that the informational and functional hierarchies should be related. We will return to this idea after a discussion of hierarchical representation by graphs and matrices.

4.3 Purpose (Function) and Information

We know that self-organization can occur under certain conditions at the microscopic level. For example, amino acids may form in a broth consisting initially of elementary chemicals.

In contrast, self-organization is never seen to occur in inanimate systems at the macroscopic level. In fact this descrepancy between the microscopic and macroscopic levels accounts for the paradox between the First Law of thermodynamics and the results of statistical mechanics. Statistical mechanics would assert, for example, that there is a nonzero probability that all the gas molecules in this room will rush to one corner at any given time. But the First Law would forbid such an occurrence, since the higher gas pressure in that corner of the room would imply the spontaneous availability of useful energy. This paradox is resolved by noting that the probability of occurrence of such an event, although nonzero, is so small as to be negligible for practical purposes.

The situation is similar in terms of the tunneling effect. The electronic switch known as the Josephson junction works on the principle that an electron can pass through barriers that, according to classical laws, should contain the electron. But at the macroscopic level we would expect a bridge, for example, to remain in one place rather than to travel around the countryside. In a similar way, one can never expect to observe a car assembled spontaneously from an assortment of components.

It is consistent with the laws of physics to assert that any useful object--at least at the macroscopic level--must be constructed with some expenditure of energy. Moreover, the application of the energy cannot be random, since the result may be a destruction of the original components. Rather, the energy must be structured; that is, it must be directed through information.

We take as axiomatic the following proposition:

> Postulate. A system consisting of arbitrary components arranged randomly and behaving in random ways cannot serve a useful purpose.

An immediate consequence is the following corollary:[1]

> Corollary. A purposive system must consist of well-defined components exhibiting nonrandom structure or behavior.

---

[1] By the properties of logical functors, (Random -> not Useful) is equivalent to (Useful -> not Random).

Purposive or purposeful behavior is one which serves to attain a goal. A distinction, however, may be made between them.

Definition.

1. *Purposive* behavior pertains to a physical, natural or artificial system whose objectives are assigned or inferred.

2. *Purposeful* behavior applies to a system which can choose its own goals.

A purposeful system can select its own goals and the means with which to pursue them, independently of environmental conditions. Such a system exhibits will, as exemplified by human beings [Ackoff and Emery 1972: 31]. The remainder of this chapter, however, deals only with purposive systems.

Definition. An action a of a system σ has the *purpose* or *goal* g iff the following conditions hold:

1. σ values g.

2. σ may choose to perform action a, or not.

3. σ has learned that executing a results in, or increases the likelihood of, attaining g.

We take the view that any artifact which fulfills a set of functional requirements is a purposive object. Hence any manufacturing system as well as its components, are purposive objects.

Knowledge refers to an awareness of the state of the world or of procedures for attaining goals. Information is a subset of knowledge that pertains only to the state of the world. Data is a quantity that can be transformed into information by the use of knowledge.

Definition. Let σ be a system and G a set of goals. Then

1. σ has *knowledge* K iff K is a fact relating to the state of the world (whether inside or outside σ), or if K can be used to to attain some goal g in G.

2. σ has *information* I iff I is a type of knowledge pertaining to the state of the world.

3. σ has *data* D iff it can use its store of knowledge to transform D into information.

The nonrandom nature of purposive systems may be encapsulated by a quantity called structural information.

Definition. *Structural information* is a measure of the deviation from randomness in the structure or behavior of a purposive system.

Typology of Purposive Systems

Purposive systems may be identified along one dimension in terms of corporeity: *hardware, software* or *hybrid*, where the last category refers to an amalgam of the first two. Moreover purposive systems may be partitioned along a second dimension based on the degree of initiative that they exhibit:

1. *Passive.* The system does not operate on the environment in response to input.

2. *Active.* The system operates on the environment in response to input.

These categories are useful for purposes of discussion but may not always have clear-cut boundaries. For example, consider a software package for logging transactions by making changes in a data set or data file. If the package is considered to consist of the algoritmic procedure as well as the data set, then all changes are internal to the system; hence the package is passive. On the other hand, if the data set is part of an autonomous database, then the data set is external to the logging package; consequently the transaction package is an active system.

4.4 A Generalized Information Model

Information accounts for a large fraction of the total manufacturing cost [Suh 1984], and therefore plays a key role in the axiomatic methodology. If we are to manage information judiciously both in the design and production phases of manufacturing, then it must be measured properly. Despite the central position that information occupies in the manufacturing arena, little has been written about the nature and characteristics of information as they relate to the fulfillment of functional requirements and manufacturing performance. This section is intended to address this shortcoming.

We first turn to a discussion of the limitations of classical information theory (CIT), whose basis is statistical uncertainty (background material on CIT is available in Appendix D). This is followed by a new model of information based on the probability of success, called the Generalized Information

Model (GIM). CIT is then shown to be a special case of GIM. Finally, information characteristics are discussed in terms of effectiveness, efficiency and timeliness.

4.4.1 Classical Information Model

The definition of information used by communications engineers provides us with a quantitative measure of information. This formulation, however, depends strictly on the probabilistic nature of a predefined set of symbols. By focusing only on the statistical properties and ignoring the semantic content of the symbols, we are left with a metric which is of limited use in modeling decision making situations.

For example, suppose that a production manager receives a report stating that the following year's sales forecast F for Crazie Gloo is 27.4 tons per week. What is the "information content" of this communication?

There are major difficulties in attempting to apply classical information theory (CIT) in this context.

1. *Lack of a predefined set of symbols or messages.* The classical definition of information is based on a predefined, finite set of symbols $\{s_i\}$ originating from an information source. In our example, the set of messages $\{F_i\}$ is uncountably infinite.

2. *Unknown probability distribution.* CIT assumes that p_i, the probability of occurrence of symbol s_i, is well defined. In our example, one would be hard-pressed to offer a probability density a priori.

3. *Blindness to semantic content.* In addition to the technical difficulties above, the classical definition of information was never intended to capture semantic content or value. A string of bits representing a sales forecast is calculated to have the same "information" whether received by a production manager, a robot, a donkey, or no one at all. This is somewhat at odds with our everyday notion of "information".

For these reasons, the notion of information must be modified and/or generalized for use in decision making environments.

4.4.2 A Cybernetic Model of Information and Decision Making

In everyday speech, we associate information with the value of a communication. A medical journal is likely to convey more information to a physician than to a carpenter, for example.

When we speak of "value", there is a notion, implicit if not explicit, of value with respect to some purpose or objective. The value of a communication depends on the receiver and his goals. A communication that takes a decision maker DM closer to his goal conveys information; one that takes him further away conveys misinformation. Since we live in a nondeterministic world, we may generalize the idea of "moving closer" to a goal by saying "more likely to attain" the goal.

The cybernetic model of information is as follows. Let a *decision maker* DM be a purposive entity with a goal set G, which consists of one or more goals. Let E be the event that goal G is attained, and p the probability that E occurs. Let p_0 be the prior probability before the receipt of a communication C, and p_1 the posterior probability. The information content of C is defined as

$$I = \mathrm{lb}\, p_1/p_0$$

In other words, information is a measure of the probability of success as proposed in [Nakazawa and Suh 1984]. If a communication reduces the likelihood that E will occur, then $p_1 < p_0$ and therefore $I < 0$. Such a communication conveys misinformation.

By writing the information equation as $I = \mathrm{lb}(1/p_0) - \mathrm{lb}(1/p_1)$, we see that information is the logarithmic difference in the inverses of probabilities. This interpretation is consistent with the proposition that the information needed by a subsystem to perform a given task is equal to the difference in the information required for the task, versus that which is already available [Galbraith 1977].

A number of corollaries follow.

Corollary. If a communication does not affect the probability of success, then the information content of the message is 0.

Proof. If the probability of success is unaffected, then $p_1 = p_0$. Hence $I = \mathrm{lb}\, 1 = 0$.

Corollary. If the probability of attaining the goal state is 1, then no communication can convey positive information.

Proof. Since $p_0 = 1$, $I = \text{lb } p_1 \leq 0$. Hence a communication can convey no information at best, and misinformation otherwise.

4.4.3 CIT as a Special Case of GIM

CIT is the offspring of studies in communications engineering. The purpose of a communication channel is to reproduce messages faithfully. Let $\{X_1,....,X_m\}$ be the set of messages or symbols originating at a source S, with probability set $\{p_1,....,p_m\}$. These symbols are transmitted by a channel to a destination D. Upon receipt of a message, the decision making unit DM at the destination must determine which of the m messages it is.

Suppose that the DM is asked to select among the m messages before having received any information. In this case the DM might as well select randomly among the messages. If the actual message to be sent is X_i, then the prior probability that the DM will successfully select X_i is p_i. After receipt of the communication, the probability of success is 1. Since the prior and posterior probabilities are p_i and 1, respectively, the information content of the communication is

$$I_i = \text{lb}(1/p_i)$$

The average information or variability among all m messages is then

$$I = \Sigma_{i=1,...,m}\, p_i * \text{lb}(1/p_i)$$

which is the classical formulation of information. Hence CIT may be viewed as a special case of GIM.

4.5 Characteristics of Information

Descriptions of physical systems, being idealized models, will often be at odds with reality. Moreover a physical plant will be subject to disturbances, not all of which can be accounted for in advance [Tsypkin 1971]. Hence our confidence in any component designed to control a physical plant will increase if real-time data on both the model and plant are monitored and reconciled.

In a static situation, process information is unnecessary. The system can be configured at the beginning and trusted to behave as originally implemented. In reality, the system may change (e.g. due to wear and tear) as may the environment (e.g. as reflected in temperature fluctuations). For these reasons, the system requires information to cope with disturbances.

Even when information about a situation is available, however, it may be misleading by being irrelevant (not effective), incomplete (not sufficient), or delayed (not timely). Usually, however, the controller does not require complete information about a situation in order to make proper decisions. Visual information impinging on the human retina, for example, is aggregated and condensed through several levels until only a small fraction of the original information filters into the visual cortex.

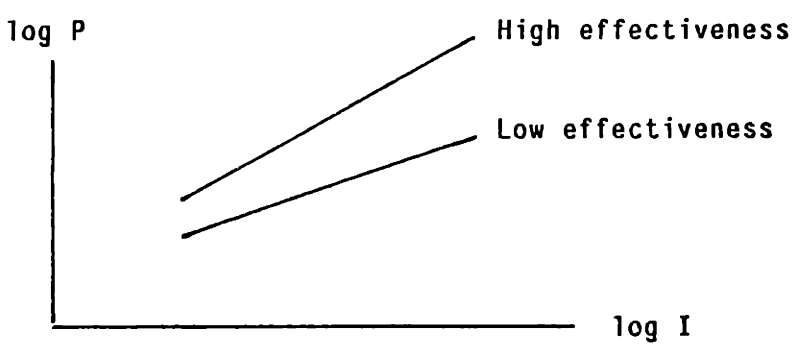
4.5.1 Effectiveness

The *effectiveness* or *criticality* of information may be defined by the elasticity or sensitivity of performance with respect to the information variable. Let P denote a performance index relating to a functional requirement, and I a measure of information. Then the effectiveness of I is given by

$$E = \frac{\partial P/P}{\partial I/I}$$

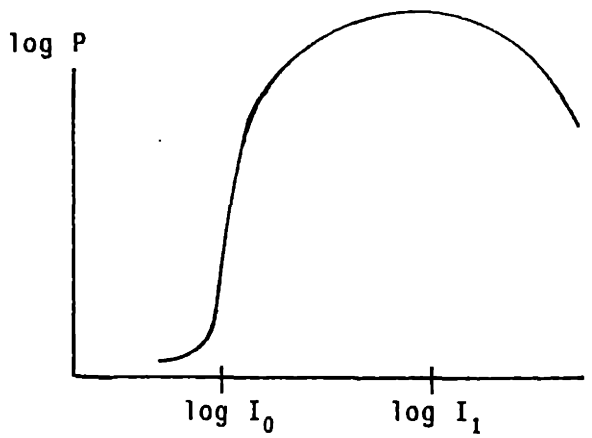
where ∂ denotes the partial derivative.

By the property of logarithms, E is equal to $\partial(\log P)/\partial(\log I)$. Hence a plot of log P against log I will highlight the relative effectiveness of different types of information.



Let I_0 correspond to the minimal information needed to yield the performance level P_0. For large values of I, we would expect diminishing returns in P. Hence the log P versus log I curve will level off.

If the decision making entity must process the stream of incoming information to sift out the critical components, excessive information input may well lead to decreased performance. In the figure, log P begins to decrease for $\log I > \log I_1$.



Such a declining curve reflects the phenomenon of *information overload* often observed in biological subsystems. At a higher systemic level, the curve mirrors the proposition that decision makers often receive too much irrelevant information rather than too little in a highly computerized society.

Axiom 2 would require that the actual information I' be as close as possible to I_0 without falling below this threshold. This idea is better addressed by the criterion of efficiency.

4.5.2 Efficiency

Efficiency is a measure of the parsimony with which information is conveyed. Let I_0 be the *sufficient*, or minimal, information required to fulfill a particular functional requirement. Then the actual information I' may be less than, equal to, or greater than I_0.

1. $I < I_0$. The actual information I' is insufficient for the task intended. The *relative deficiency* $\delta = (I_0 - I')/I_0$ is a measure of the extent to which the actual information falls short of the amount needed.

2. $I' = I_0$. The actual information is also the sufficient information. In this case Axiom 2 is satisfied.

3. $I' > I_0$. The actual information is in excess of the minimum required. The *efficiency* $\eta = I_0/I'$ is a measure of information parsimony. Its complement, $R = 1 - \eta$, may be called *redundancy.*

By construction all three metrics, δ, E and R, take on values in the half-open interval (0,1]. Efficiency may fall below 1 when redundancy is required in order to guard against the effects of noise or the

possibility of failure in various components.

4.5.3 Timeliness

. Information loses value over time due to changes in internal or external factors. An example of an internal disturbance is drift in a system parameter, such as positioning error due to mechanical wear. Examples of external disturbances are changes in temperature or vibration levels. Sometimes the external and internal factors, while innocuous in isolation, may have dramatic repercussions in concert; this is exemplified by the phenomenon of resonance.

In a changing situation, a purposive system must react in time to adapt to the change. An intelligent lathe, for example, must respond quickly to tool breakage if it is to avoid ruining the workpiece.

Hence the information must be available quickly to allow enough time for the system to respond. The response lag may be due to both software and hardware effects:

1. The decision making unit needs time in which to process the incoming data and make a decision.

2. Due to inertial effects, the system needs time to accomodate itself to the new decision.

One way to categorize time delays is to classify them into different functional phases in terms of sensing, processing, comparing, deciding, and acting stages [Lawson 1981]. The figure below

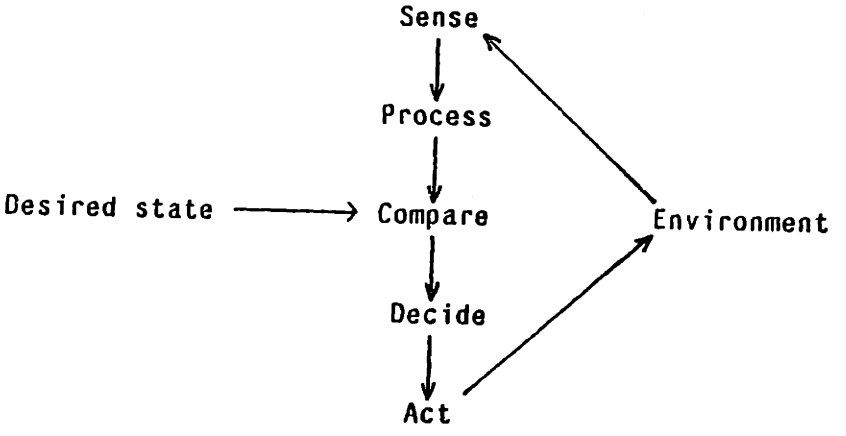


Figure 4-1:Time delays due to sensing, processing, comparing, deciding, and acting stages.

indicates that sensing delays occur in extracting information from the environment. The sensed data

must then be interpreted, and compared against some standard configuration stored in memory. The comparison leads to a decision, which in turn stimulates some action that affects the environment.

An example is found in an automatic inspection system. The robotic system must first sense the samples arriving on the materials transport system. This information is processed and compared against a canonical state stored in memory. If the match is good, the sample is routed to the "Accept" batch; otherwise it is sent to the "Reject" pile.

The interarrival time between samples constrains the maximum time available for any stage of the inspection function. Moreover, the time available as a sample moves from the sensing area to the "Accept"/"Reject" decision point, defines a window of opportunity for the inspection system. If the robotic system is too slow, it fails to properly discharge its inspection role.

4.5.4 Discussion

The attributes of effectiveness, efficiency and timeliness address the questions of *What*, *How much*, and *When*. Sufficiency and effectiveness are interrelated. To show this, we define a performance index P in terms of the associated functional requirements as follows.

Let $P_i = 1$ if the ith functional requirement is satisfied, and $P_i = 0$ otherwise. If the information for the ith functional requirement is insufficient, for example, then $P_i = 0$. Since the functional requirements are independent specifications of the problem, they must all be satisfied in order for the problem to be solved. If there are n functional requirements, then the overall performance index may be defined as

$$P = \Pi_{i=1,\ldots,n} P_i$$

This formula indicates that $P = 1$ if and only if each $P_i = 1$, and $P = 0$ otherwise.

Sufficiency is defined in terms of fulfilling a set of functional requirements. Hence when information is insufficient, it is ineffective. Conversely, when information is ineffective it makes little sense to speak of sufficiency.

The efficiency and timeliness of information are vital concerns in computerized manufacturing systems [Kuo 1980: 694]. For example, the use of microprocessors results in information loss due to temporal lags and the discretization of continuous information. The increasing trend toward computerized control and intelligent machine tools implies that such concerns will become even

more paramount in the years to come.

4.5.5 Information and Process Performance

Loss of information can result in the degradation of process performance. An extreme example lies in the genesis of instability: a control system of first or second order, which is inherently stable as a continuous-time system, may become unstable as a discrete-time system. We may attribute this metamorphosis to the loss in information resulting from sampling information at discrete intervals rather than continuously.

Another class of instability occurs in various nonlinear elements such as backlash components. Here, the loss in information may be traced to lack of knowledge about the relative displacement of interfacing elements within the dead zone. Obviously, not all nonlinear elements result in instability. An example is found in saturation effects, which yield stable limit cycles.

4.6 Manufacturing System as an Information Processing Unit

Information is required to transform any object in one form into another form. Since the function of a manufacturing system is to transform raw materials and intermediate goods into other products, it must process information. This is true whether the transformation operations are performed manually, automatically, or in combination. In fact, information has been asserted to be the single most important aspect of a manufacturing system [Suh 1984]. The discussion of automata theory in Chapter 3 makes it clear that the theory is applicable to any information-processing system, including manufacturing systems.

4.6.1 Manufacturing System as an Automaton

Chapter 3 has suggested the suitability of automata theory as an exemplar of a precise model for describing the nature and interrelationships among components of a system. The suitability follows from these reasons:

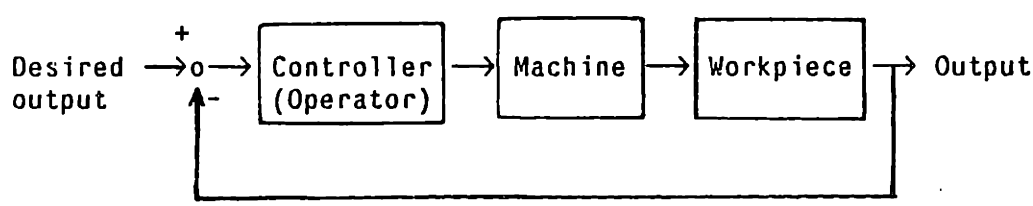
1. The theory is applicable to any system that processes information, regardless of its composition. Since information is seen to be a major component of manufacturing systems, the theory is applicable here.

2. The theory recognizes the discrete nature of objects and processes, hence is better suited to serve as a formal infrastrcture than the continuous models of cybernetics and its derivatives.

3. By recognizing programming as a central topic, at least part of the theory deals explicitly with design issues. This is compatible with the issues of manufacturing science, which was born of the need for a rigorous theory to help design superior production systems.

The role of human labor in the manufacturing process may also be subdivided into two categories: hardware functions such as materials transport and handling, and software functions such as inspection and quality control. Certain tasks, such as parts assembly, incorporate both types of functions; but these may be analyzed separately, just as they are in the case of robot workers.

We may regard the entire manufacturing system and its components, whether arranged hierarchically or not, as automatons. This includes the human operator,[2] whose job is to help keep the production line running smoothly. This situation is exemplified by a lathe operator, who serves as both the controller and feedback loop element in a simple control system.



Manufacturing systems may, in general, be modeled as automatons. One type of photocopy machine, for example, may be regarded as an automaton which accepts inputs called *paper, electricity, ink* and *information*, and yields similar outputs in somewhat altered form.

[2]While the choice of word *automaton* may be unfortunate in this context, little is to be gained by coining yet another neologism in search of a neutral term.

4.6.2 Theorems on Manufacturing Systems

An important advantage of modeling an industrial plant as an automaton is that the methods and results of automata theory may be brought to bear. To illustrate, this section discusses a number of propositions concerning the nature and limits of manufacturing systems. We begin with some definitions.

Definition. An *effective* or *admissible* process is one that is well-defined and finite.

We assume that all practical manufacturing processes are effective.

Postulate. All manufacturing processes are admissible.

This means, in particular, that any production operation is finite. However, it allows for the possibility that the manufacturing system as a whole may operate indefinitely by looping through its repertoire of operations.

Theorem. The control procedures and information-processing procedures for a admissible manufacturing system are equivalent to Turing machines.

Proof. Automata theory tells us that Turing machines are equivalent to effective processes [Ginzberg 1968]. By the previous postulate, so are control procedures and information processing requirements.

The next theorem bears on the limits of automatability.

Theorem. There are well-defined manufacturing processes which are not admissible.

Proof. For automation to occur, a well-defined process must be encodable into a series of instructions or computations. An important result of automata theory is that there are well-defined processes which are not finite, hence not effective. The theorem then follows.

If a manufacturing process does not terminate, the envisioned output will never materialize. Hence we can infer the following result.

Corollary. It is impossible to construct a generalized manufacturing system which can

manufacture arbitrary products and processes.

* * *

This chapter has shown how the infrastructure presented in Chapter 3 may be usefully applied to the analysis of manufacturing systems. The next chapter turns to the use of symbolic logic as a formalizing construct for the synthesis phase.

Chapter Five

Implications for Synthesis Phase

The Design Axioms are generalized decision rules applicable to the design of products and processes, both of the software and hardware variety [see Subsection 1.2.2]. These principles are embodiments of generally observed truths in design and manufacturing. In this sense they are equivalent to propositions which are called *laws* in the physical sciences, such as the laws of gravitation or thermodynamics.

This chapter shows how the Axioms and their corollaries may be given rigorous expression through symbolic logic. The result is that various propositions which were previously believed to be corollaries of the Axioms may be classified into two categories consisting of direct and indirect consequences. The latter category may be further subdivided into those results that require mild assumptions versus others which require more severe ones.

Moreover, stating the Axioms in symbolic logic readily suggests how they may be encoded in a logical programming language such as Prolog. A schema for an expert system is presented in the form of the Computerized Axiomatic System (CAS). CAS consists of 3 layers relating to the Primary, Secondary, and Data Levels. The Primary Level consists of the two Axioms, the Data Level to raw design specifications, and the Secondary Level to the interface between the first two.

Generalized structures are presented for encoding procedures and data in the Secondary and Data Levels. Further, the three operating modes of CAS are presented in the context of a small example. This is followed by the overall architecture for a full-fledged expert system.

5.1 Manufacturing Axioms and Predicate Logic

5.1.1 Manufacturing Axioms in Predicate Logic

A design which satisfies the functional requirements and constraints is said to be *feasible*. Let F denote the set of feasible designs. We define the predicate feas(*) in the following way: for a design

x, feas(x) is true if x is in F, and is false otherwise.

Let idm(*) be a measure of functional independence, as exemplified by reangularity and semangularity [Rinderle 1982; Rinderle and Suh 1982; Suh and Rinderle 1982]. Then idm(x) > idm(y) would imply that the functional independence of design x exceeds that of design y. In a similar way, we let ifm(*) denote a measure of information content defined on the set of feasible designs.

For convenience we sometimes use infix notation to stand for the associated predicate. For example, we will write

```
ifm(x) < ifm(y)
```

to denote the predicate

```
lessThan(ifm(x),ifm(y))
```

which is true iff (if and only if) ifm(x) is less than ifm(y).

Let sup(x,y) be the predicate asserting that design x is superior to design y. [3]We also define coup(x) to be true iff x is coupled, while unc(x) is defined as ~coup(x). In addition acc(x) means that design x is acceptable.

We turn to the task of stating the Axioms in symbolic logic. When stated informally as in Subsection 1.2.2, the Axioms are subject to misinterpretation since they harbor a number of assumptions which are only implicit. For example, a tacit assumption in Axiom 1 is its restriction to feasible designs: a design which maintains functional independence but does not meet the individual functional requirements within their allowable tolerance bands is still unacceptable [Kim and Suh 1985]. The formal statement of the Axioms must, of course, bring out all such latent hypotheses.

At the current state of art in axiomatics, we may allow for three versions of the Functional Independence Axiom--the restricted, mediate and general forms.

* Axiom 1R (Restricted Form). Only feasible designs which are (completely) uncoupled are acceptable:

[3]If desired, we may also relate sup(*,*) to a cardinal metric. Let j(*) be a measure of performance or superiority of a design, in the way that utility is used as an abstract index of usefulness in decision analysis or satisfaction in economics. Then the predicate sup(x,y) may be defined to be true if and only if the condition j(x) > j(y) holds.

```
∀x { feas(x) & unc(x)  ->  acc(x) }
```

* Axiom 1M (Mediate Form). Of two feasible designs for which one is coupled and the other not, the uncoupled design is superior:

```
∀x∀y { feas(x) & feas(y)  & unc(x) & coup(y)
        -> sup(x,y) }
```

* Axiom 1G (General Form). Of two designs, the one with higher functional independence is superior:

```
∀x∀y { feas(x) & feas(y) & idm(x)>idm(y)
        -> sup(x,y) }
```

The universal quantifiers, ∀x and ∀y, specify that the conditions within braces are valid for *all* x and y, not just some particular instances of these variables.[4]

To show that the Mediate Form of Axiom 1 is a special case of the General Form, we can relate the independence metric to the coupling predicate as follows:

```
idm(x) = 0 <--> coup(x)
idm(x) = 1 <--> ~coup(x)
```

The advantage of the General Form, of course, is its versatility.

For example, suppose that functional independence is to be assessed according to 3 nominal categories called coupled, decoupled, and uncoupled systems. Then Axiom 1G can be used by mapping these categories respectively into a sequence of increasing idm values such as <0, 0.5, 1> or

[4]Logicians often include existential checks in the antecedent to ensure that the latter refers to some valid object. For example, Axiom 1R may be written as ∀x {(∃z feas(z)) & (∃z unc(z)) & feas(x) & unc(x) -> acc(x)}. The first two expressions within the braces specify that there should be at least one feasible design and one uncoupled design. Strictly speaking, this kind of check is unnecessary and is therefore omitted here.

<-1, 0, 1>.[5]

Moreover, the general version also allows for a continuous range for idm. The General Form is compatible, for example, with the independence measures of reangularity and semangularity [Rinderle and Suh 1982] which are defined on the interval [0,1], while the Mediate Form is not.[6] The remainder of this chapter will consider Axiom 1 in the General Form.[7]

We turn to the Information Minimization Axiom. Axiom 2R below corresponds to version 1R of Axiom 1 and Axiom 2G to versions 1M and 1G.

* Axiom 2R (Restricted Form). Of two acceptable designs, the one with less information is superior.

```
∀x∀y { acc(x) & acc(y) & ifm(x)<ifm(y)
        -> sup(x,y) }
```

* Axiom 2G (General Form). Given two feasible designs of equal functional independence, the one with less information is superior.

```
∀x∀y { feas(x) & feas(y) & idm(x)=idm(y)
        & ifm(x)<ifm(y)  ->  sup(x,y) }
```

5.1.2 Discussion on Multiple Versions of Axiom 1

Axiom 1 serves as a filter for designs before they are passed on to Axiom 2. Axiom 1G is a large-grained filter which admits the largest set of designs; version 1M permits only a subset of those allowed by 1G; and 1R accepts a subset of those approved by 1M.

---

[5] Since these values are to be used for direct comparison among pairs of designs rather than as a basis for further calculation, any monotonic transformation of <0, 0.5, 1> will serve.

[6] Axiom 1M is a specialization of the general version given as Axiom 1G in the following sense. Let A be the set of cases—in our case, designs—to which Axiom 1M applies, and B the set to which Axiom 1G applies; then A is a proper subset of B. The same relationship applies to version 1R as a specialization of 1M. The use of the terms *generalization* and *specialization* is consistent, of course, with their usage in other areas of science.

[7] At present, there is some debate about the usefulness of allowing for *continuous* measures of functional independence such as reangularity and semangularity. Hence Axiom 1G may fall into desuetude in the future. But since 1R and 1M are special cases of 1G, the arguments in this chapter would still remain valid.

In general, the choice of version 1R, 1M or 1G will have no effect on the relationships among the Axioms and corollaries. For example, a corollary that was derivable from Axiom 1 will still be derivable no matter which version is used. Of course, each corollary which has to date been available only in an informal version must be stated formally in accordance with the corresponding version of the Axioms. For example, invoking version 1R necessitates use of the acc(*) predicate in any corollary that depends on the first axiom, while version 1G requires use of the sup(*,*) predicate.

In short, different versions of the Axioms affect the set of admissible designs, but not the validity of the relationships among the Axioms and their derivatives. For convenience only versions 1G and 2G are explicitly considered in the remainder of this chapter. The applicability of the reasoning to the other versions is transparent.

5.1.3 Implications of the Axioms

A number of propositions were included in the original list of decision rules set forth in [Suh et al. 1978a]. Subsequent research indicated that the Function and Information Axioms are the key concepts, and that the other decision rules follow as corollaries. The next two subsections offer a more rigorous argument than was previously possible. The third subsection, in fact, indicates that previous beliefs concerning the lineage of certain corollaries should be revised.

5.1.3.1 Direct Consequences

The following proposition is an immediate consequence of the Functional Independence Axiom. Since idm for a decoupled system is greater than that for a coupled system, decoupling is advisable.

Proposition (Decoupling). Decouple functions which are coupled.

The Decoupling Corollary may be stated more precisely in the following way. Let u and v be two feasible designs in which the functional independence of u exceeds that of v; that is, idm(u) > idm(v). Then we have the following facts:

1. `feas(u).`
2. `feas(v).`
3. `idm(u) > idm(v).`

If we instantiate x to u and y to v in Axiom 1, the result is

```
4. feas(u) & feas(v) & idm(u)>idm(v)
   -> sup(u,v).
```

These five items yield the conclusion

```
5. sup(u,v).
```

This Proposition is related solely to Axiom 1. The only way for a theorem to be provable from a single axiom is for it to have the same formal structure as the axiom, as is the case here. In other words, the Decoupling Proposition is a corollary or alternative statement of Axiom 1 in an informal sense, but is strictly a restatement of the Axiom in a formal sense. Hence it would be more appropriate to call this proposition an *alternative informal statement* of Axiom 1 rather than a *corollary*.

5.1.3.2 Indirect Consequences

A number of other corollaries are not, strictly speaking, obtainable solely from the axioms. With some assumptions, however, they may be justified as indirect consequences. Some of these assumptions are mild, while others are more substantial.

Weak Assumptions

The first corollary depends on the reasonable assumption that information decreases with the number or complexity of part surfaces. This postulate may be written, using obvious choices for predicates and functions, as

```
1. (numSurf(u) < numSurf(v))      &
   (complexSurf(u) < complexSurf(v))
   -> ifm(u) < ifm(v).
```

The informal statement of the corollary is

Corollary (Surface Finish). Minimize the number and complexity of part surfaces.

A formal statement of the corollary is

Corollary (Surface Finish). Given two designs of equal functional independence from a nonempty set of feasible solutions, the one with fewer and less complex part surfaces is superior.

```
∀x∀y { feas(x) & feas(y) & idm(x)=idm(y) &
       (numSurf(x) < numSurf(y)      &
       complexSurf(x) < complexSurf(y))
       -> sup(x,y) }
```

By instantiating x to u and y to v, the antecedent is composed of the following facts:

```
2. feas(u).
3. feas(v).
4. idm(u) = idm(v).
5. numSurf(u)<numSurf(v)      &
   complexSurf(u)<complexSurf(v).
```

From items 1 and 5, we obtain

```
6. ifm(u) < ifm(v).
```

We instantiate x to u and y to v in Axiom 2, giving

```
7. feas(u) & feas(v) & idm(u)=idm(v) & ifm(u)<ifm(v)
   -> sup(u,v).
```

Items 2, 3, 4 and 6 in conjunction with item 7 implies

```
8. sup(u,v).
```

which is the consequent of the Surface Finish Corollary. Since u and v are arbitrary designs (which satisfy the antecedent), the result is valid for all x and y. Hence the Corollary follows.

From this example, it is clear that a formal proof can get long-winded even for a relatively simple corollary. In the interest of expedience, the corollaries that follow are proved informally.

The next corollary requires the simple assumption that processing information is a subset of information in general, and therefore--other things being equal--information decreases when processing information does. This assumption plus Axiom 2 implies

Corollary (I_p). Minimize processing information.

The following corollary relies on the Purpose Postulate, the hypothesis that the fulfillment of functional requirements requires order or structure, and hence information.

Corollary (Conservation). Conserve materials and energy.

Proof. Suppose that designs x and y are feasible, that idm(x) = idm(y), and that the amount of material used in x is a proper subset of that of y. Let w = y - x, where the minus sign denotes a difference operation on sets; in other words, w represents the "excess" material in y when compared to x.

Adding w to x randomly cannot improve the performance of x. Since w must be placed judiciously to prevent impairing x's performance, its addition requires extra information. By Axiom 2, the original design x is superior to design y.

The argument against superfluous energy requirements is identical to that for materials. Hence the Conservation Corollary follows.

The next proposition is based on the assumption that an increase in the number of components increases information.

Corollary (Integration). Integrate components if functional independence is not impaired.

Proof. Integration implies that at least one interface or component is combined, thereby reducing information requirements. The corollary then follows from Axiom 2.

Example. Let x be a system consisting of two cubes, each with edges of length 1 ± 0.1. The the geometric information is given by ifm(x) = 2*(3*lb(1/.1)) = 19.93 bits, where *lb* refers to the binary logarithm (log to base 2) and *bit* to the corresponding unit of measure [cf. Appendix D].

Suppose that system y is a larger block which represents the integration of x's unit blocks. If we consider the long edge of the block to have tolerance 0.2, then ifm(y) = 2*lb(1/.1) + lb(2/.2) = 9.97 bits. On the other hand, if we consider the long edge to have a tolerance of 0.1, then ifm(y) = 2*lb(1/.1) + lb(2/.1) = 10.96 bits. Either way,

information requirements are lower in the integrated system y.

An immediate consequence is the following.

Corollary (Part Count). Part count is not a measure of productivity.

Proof. The Integration Corollary implies that productivity may increase when the number of parts decreases. On the other hand, a proliferation of parts will--in general--result in an increase in information; but this outcome is anathema to Axiom 2. In short, productivity may increase with a rise *or* fall in the number of parts.

The following two results rely on the assumption that the information requirement for denoting "ditto" or "same as before" requires less information than the original specification.

Corollary (Standard Parts). Use standard or interchangeable parts whenever possible.

Proof. Let x_1 and x_2 be two subunits of system x, while y_1 and y_2 are two subunits of y; the structure and behavior of all other subsystems in x and y are identical.

Suppose that x_1 and y_1 are identical components, as are y_1 and y_2; in other words, only x_2 differs from the others in this set of 4 subsystems. We may reasonably assume that each of the 4 subunits is of sufficient complexity that its digital encoding requires more information than a code for replication, such as "Same as y_1" or "ditto".

Then the information requirements for encoding systems x and y are given by

$$\text{ifm}(x) = \text{ifm}(x_1) + \text{ifm}(x_2) + c$$

$$\text{ifm}(y) = \text{ifm}(y_1) + \text{ifm}(\text{"ditto"}) + c$$

where c is some constant. Since we have assumed that x_1 and y_1 are identical, so are their information requirements. Moreover, $\text{ifm}(x_2) > \text{ifm}(\text{"ditto"})$. Hence $\text{ifm}(x) > \text{ifm}(y)$.

By Axiom 2, standardization is preferable. This argument is easily generalizable to replications of 3 or more components.

Corollary (Standard Processes). Use standardized processes whenever possible.

Proof. According to the Church-Turing thesis [Ginzberg 1968], any effective process can be expressed algorithmically and encoded for mechanical interpretation. Hence the basic argument used in the preceding proof applies here as well.

The Standard Processes Corollary encapsulates the idea that a repetitive procedure reduces information, for example by eliminating the need to change cutting tools in machining a workpiece.

Strong Assumptions

Despite previous beliefs to the contrary, the following propositions are independent decision rules which cannot be justified solely on the basis of Axioms 1 and 2. They may be considered to be informal statements of entirely new axioms, or may be derived from the two existing Axioms plus some additional hypotheses.

The first of the indirect propositions which would require strong assumptions is the following:[8]

Proposition (Weakness). If weaknesses cannot be avoided, separate parts.

A more precise statement of this Proposition is as follows. Suppose u and v are two feasible designs of equal functional independence and degree of weakness; u has components which are separated at a physical weakness while v does not; moreover, the designs have equal information content when the geometric information for the separation in u is adjusted for; then u is superior to v. In predicate logic we may write:

```
feas(u) & feas(v) & idm(u)=idm(v) &
ifm(u)=ifm(v) & equallyWeak(u,v) &
separateParts(u) & ~separateParts(v)
-> sup(u,v)
```

However, neither Axiom 1 nor 2 has predicates called equallyWeak or separateParts. Hence the

[8]The motivation behind this Proposition is that a weak joint will be prone to failure unless fortified. On the other hand, eliminating the weak joint by designing two separate parts will reduce the likelihood of catastropic failure.

Weakness Proposition cannot be derived from the key Axioms.

This is an example of a more general phenomenon. The following metatheorem states that no new predicate can be derived from a set of axioms which does not contain such a predicate.

Metatheorem (Nondeducibility). Let A be a set of axioms, none of which includes the predicate P. Then no theorem which includes P can be derived from A.

Proof. The equivalence of the semantic and syntactic approaches to symbolic logic [cf. Subsection 2.1.2.4] implies that we need to consider only one of the two viewpoints. Since the metatheorem pertains to provability, the natural approach is to take the syntactic view.

We make the following assumptions:

1. Without loss of generality, let $A = \{B_1, B_2, ..., B_m, C_1 \rightarrow D_1, C_2 \rightarrow D_2, ..., C_n \rightarrow D_n\}$ be the set of axioms. We assume that the B_i's contain no implication symbols at the highest level; for example, ((U -> V) & W) is a permissible expression for some B_i, but not (U -> V).

2. Each expression B_i, C_j, and D_k consists of one or more predicates, none of which is the predicate P. (The predicate P may contain zero or more variables.)

3. $T = \langle T_1, T_2, ..., T_q, ... \rangle$ is a sequence of theorems resulting from A. In other words, each T_q is a theorem of $A \cup \{T_1, T_2, ..., T_{q-1}\}$.

The proof will proceed by induction.[9]

If $m = n = 0$ in A, then A is empty and there is nothing to prove. Henceforth we assume that A is nonempty (i.e. that $m + n \geq 1$).

From Assumption 3 above, T_1 is the first theorem resulting from A. The only way to

[9] An inductive proof involves the following steps: (1) Show that a given condition C holds for the base case, $n = 1$. (2) Assume that C holds for each of cases $n = 1,2,...,q$. (3) Show that the previous assumption implies that C holds for case $n = q + 1$.

obtain T_1 from A is through generalization or modus ponens [cf. Subsection 2.1.2.3].

1. If generalization is used, then some axiom A_i of A yields the theorem $(\forall x\ A_i)$. That is,

   $$A_i \vdash \forall x\, A_i$$

   Since A_i does not contain the predicate P, however, neither does $\forall x\ A_i$.

2. If modus ponens is used, then some axiom B_i must equal some premise C_j in A, thereby deriving theorem D_j. That is,

   $$(C_j \,\&\, (C_j \rightarrow D_j)) \vdash D_j$$

   By Assumption 2, however, D_j does not contain the predicate P.

We see that the first theorem, T_1, cannot include predicate P no matter how it is derived.

Now assume that the set $\{T_1,\ldots, T_q\}$ comprises the first q theorems of A, none of which contains predicate P. Then T_{q+1} is the subsequent theorem of the augmented set $A \cup \{T_1,\ldots, T_q\}$.

By a line of reasoning similar to that in the previous paragraph, T_{q+1} is free of predicate P. Hence no theorem can be derived which contains P, and the metatheorem holds.

The next two propositions are also unable to trace their lineage to the Axioms, since the Axioms contain no predicates relating to continuity, energy conduction, cost, or surface area.

Proposition (Continuum). A part should be a continuum if energy conduction is important.

Proposition (Surface Area). Cost is not proportional to surface area.

The following proposition also lies outside the framework of the Function and Information Axioms. Suppose, however, that Axiom 2 is expanded from "Minimize information subject to functional requirements" to simply "Minimize information". Then Axiom 2 and the Purpose Postulate imply the following statement.

Corollary (Minimization of FR's). Minimize the number of FR's.

Proof. If the functions of a system A are a proper subset of those of system B, then system B must do all that A can do, and more. The Purpose Postulate states that the fulfillment of any purpose or function requires information. Consequently, invoking Axiom 2 in the expanded form implies that the number of functional requirements must be minimized.

In summary, the propositions in this subsection cannot be derived strictly from the two Design Axioms. Hence they should be considered to be independent axioms, or may be obtained from the existing Axioms when augmented by other hypotheses. Such derivations, when appropriate, may proceed as illustrated in connection with the Surface Finish Corollary at the beginning of this subsection.

5.2 Computerized Axiomatic System (CAS)

Prolog (*Pro*gramming in *log*ic) is a very high-level programming language based on mathematical logic [Clocksin and Mellish, 1981; Futo et al., 1978; Gallaire and Minker, 1978; Gallaire, Minker and Nicolas, 1978; Kowalski, 1974; Pereira, 1984]. It was created at Marseille [Battani and Meloni, 1973] and adopted by Japan as the language of choice for the Fifth Generation Computer Project. Further background on Prolog is available in Appendix F. The ability to state the Axioms as clauses in a logical programming language indicates the feasibility of developing an expert system for axiomatics [Horstmann, 1983; Mizoguchi, 1983; Oliveira, 1984; Walker, 1983].

This section describes the Computerized Axiomatic System (CAS), a schema to generate an axiomatics expert system in Prolog. In addition, the modes of operation of CAS are discussed in terms of a simple example.

5.2.1 Primary Level

The Primary Level of CAS consists of the Synthesis Axioms encoded in Prolog. Let idf(X,Idm) be a functor which maps a design X into its independence measure Idm. Then Axiom 1 may be written as

```
sup1(X,Y) :-
        feas(X),
        feas(Y),
        idf(X,Idmx),
        idf(Y,Idmy),
        Idmx > Idmy.
```

This says that design X is superior to design Y if, for feasible designs X and Y, the independence measure of X exceeds that of Y. The notational convention in Prolog is that an identifier which begins with a capital letter (e.g. X) denotes a variable, while one that starts with a lower-case letter (e.g. p) is a constant.

We now turn to the Information Axiom. Let iff(X,Ifm) be a predicate which maps a design X into its information metric Ifm. Then Axiom 2 may be given as

```
sup2(X,Y) :-
        feas(X),
        feas(Y),
        idf(X,Idmx),
        idf(Y,Idmy),
        Idmx = Idmy,
        iff(X,Ifmx),
        iff(Y,Ifmy),
        Ifmx < Ifmy.
```

The verbal translation is that design X is superior to design Y if, given feasible designs of equal functional independence, X has less information content than Y.

Axiom 2 stated as a Prolog clause contains 5 predicates in its antecedent or body. The first two refer directly to Axiom 1, while the fifth is a simple logical test. The only substantial component pertains to the two manifestations of the iff predicate. This construct is part of the Secondary Level, to be described in Subsection 5.2.3.

5.2.2 Data Level

The Data Level pertains to structures for representing raw design specifications. A generalized representation of data takes the form Dj(X,Lj), where Dj denotes the jth type of data for design X and Lj the corresponding list of attributes.

For example, suppose that the surface tolerance information for the machined surfaces on design X is U ± DU in some consistent units such as micrometers. Then the appropriate data structure is given by the pair of substitutions

```
Dj ::= surfd ,
Lj ::= [U,DU],
```

where the compound symbol ::= means *is defined as*. In other words, the resulting data structure is

```
surfd(X, [U,DU]).
```

To take another case, consider the geometric information for a rectangular block whose 3 dimensions are specified as U1 ± DU1, U2 ± DU2, and U3 ± DU3 units. The corresponding data structure is given by Dk ::= geomd and Lk ::= [U1,DU1,U2,DU2,U3,DU3].

When the parameter list Lj is short, we prefer to simplify the notation by omitting the square brackets which serve as delimiters for lists. For example, when Lj is empty, we use the format Dj(X) rather than Dj(X,[ ]); when Lj is a singleton, we write Dj(X,Item) in lieu of Dj(X,[Item]).

An appropriate data structure for functional requirements, for example, takes the form

```
fr(Name, [I,Fi,DFi]).
```

This specifies that the Ith functional requirement for the design problem called Name has the value given by Fi, with tolerance DFi.[10]

The actual values of the functional requirements for each design may be encoded by the data structure

```
dFR(X, [I,Fi]).
```

where Fi is the actual or intended value of the Ith functional requirement for design X. In addition, we may use the data structure

```
cCoef(X, [I,J]).
```

to mean that the $\langle I,J \rangle^{th}$ coupling coefficient for design X is nonzero.

5.2.3 Secondary Level

The Secondary or Intermediate Level serves as the interface between the Primary and Data Levels. The first aspect of satisfying Axiom 1 relates to feasibility. A design X is feasible if it meets each of the functional requirements for the problem:

```
feas(X) :- meetFR(X,1), meetFR(X,2),..., meetFR(X,n).
```

[10] If the lower and upper tolerances--LTi and UTi respectively--are unequal, then the data structure may be modified to fr(Name,[I,Fi,LTi,UTi]).

where meetFR(X,I) is satisfied if design X meets the Ith functional requirement. The role of the meetFR(X,I) functor is to check that the operating value of the Ith functional requirement for design X lies within the tolerance band prescribed for the problem.

The second aspect of satisfying Axiom 1 pertains to the idf(X,Idm) functor, which maps a design X into its independence measure Idm. A categorical measure of functional independence classifies designs into three groups based on functional independence: coupled, decoupled and uncoupled; these correspond respectively to fully cluttered, triangular, and diagonal coupling matrices.[11] An appropriate assignment of values to Idm would be "low" for a coupled system, "medium" for a decoupled system, and "high" for an uncoupled one. Alternatively, the 3-way classification <coup, dec, unc> may be mapped onto the codomain <0, 0.5, 1> or onto <-1, 0, 1>.[12]

To illustrate, consider a coupling matrix of the form

$$\begin{vmatrix} ax^2 & 0 \\ be^y & cxy \end{vmatrix}$$

which has the general structure

$$\begin{vmatrix} X & 0 \\ X & X \end{vmatrix}$$

The triangular structure of the matrix makes it obvious that the system is decoupled, implying that Idm = 0 when the classification {-1, 0, 1} is used.

Consider now the second Axiom. Let iff(X,Ifm) be the predicate which takes a design X and yields its information measure Ifm. The iff clause must take into account all the different types of information which are of consequence for the application at hand. Its general structure is

```
iff(X,Ifm) :-
        F1(X,I1),
        F2(X,I2),
```

[11] Further discussion of coupling matrices is available in Appendix C as well as in [Rinderle 1982; Rinderle and Suh 1982; Suh and Rinderle 1982].

[12] At the current state of art of design axiomatics, measures of functional independence are based on values which represent ordinal rather than ratio or even interval scales. Hence monotonic transformations of the codomain of Idm, as in this example, are equivalent.

```
        :
        :
    Fn(X,In),
    Ifm is (I1 + I2 + ... + In).
```

Here Fj is the jth functor which takes design X and returns the information value Ij of the jth attribute. The last item in the body of the iff clause sums up the component information values, I1 through In, to yield the overall measure Ifm.

To illustrate, consider the two types of information discussed previously, namely surface smoothness and geometric tolerance. Let F1 ::= surff, I1 ::= Surfm, F2 ::= geomf and I2 ::= Geomm. Then we obtain

```
iff(X,Ifm) :-
        surff(X,Surfm),
        geomf(X,Geomm),
        Ifm is (Surfm + Geomm).
```

In other words, the iff predicate takes design X, determines its information values due to surface and geometry specifications, and returns their sum.

We note that the iff clause interfaces directly with the clauses in the Primary Level. We also need a sublevel consisting of clauses that interface directly with the Data Level. This is achieved through the Fj functors introduced above.

The general structure of the Fj functor is

```
Fj(X,Ij) :-
        Dj(X,Lj),
        Oj(Lj,Ij).
```

The first item in the body finds the jth data structure Dj for design X and returns its attribute list Lj. The second item Oj is a set of predicates which takes Lj and yields the corresponding information value Ij.

Consider the data structure relating to surface tolerance. A plausible Fj clause is

```
surff(X,Surfm) :-
        surfd(X, List),
        infoLM(List, Surfm).
```

This clause corresponds to the general structure by the following set of substitutions:

```
Fj ::= surff,
Ij ::= Surfm,
Dj ::= surfd,
```

```
Lj ::= List,
Oj ::= infoLM.
```

In other words, surff takes a design X, obtains its list of specifications and returns the corresponding information value Surfm through the functor infoLM.

The case for geometric information is similar. Both the surff and geomf functions depend on the infoLM(L,M) predicate, which takes a list L of tolerance parameters Ui ± DUi and returns the resulting information measure M. In other words, infoLM accepts a list L = [U1,DU1,U2,DU2,...,Un,DUn] and calculates

$$M = lb(U1/DU1) + lb(U2/DU2) + ... + lb(Un/DUn).$$

where lb denotes the binary logarithm (log to base 2).[13]

The clauses described so far comprise the Computerized Axiomatic System (CAS), which is presented as Figures 5-1 through 5-3.

```
/* Primary Level */

        sup1(X,Y) :-
                feas(X),
                feas(Y),
                idf(X,Idmx),
                idf(Y,Idmy),
                Idmx > Idmy.

        sup2(X,Y) :-
                feas(X),
                feas(Y),
                idf(X,Idmx),
                idf(Y,Idmy),
                Idmx = Idmy,
                iff(X,Ifmx),
                iff(Y,Ifmy),
                Ifmx < Ifmy.
```

Figure 5-1:Primary Level of Computerized Axiomatic System (CAS).

[13]The infoLM functor works by passing List (the list of attribute pairs) and an inital value of 0 to the infoLM1 predicate. In turn infoLM1 recursively evaluates MesoM = lb(Ui/DUi) for each pair [Ui,DUi] in List and adds it to the cumulative total Oldm. When the list of data pairs is exhausted, Oldm is returned as the desired value of Meas in infoLM.

In these figures, any wording enclosed by the /* and */ delimiters are programming comments which are ignored by the Prolog interpreter. In Figure 5-2, the idf predicate has been defined in such a way that the categories {coup, dec, unc} correspond respectively to Idm values of {-1, 0, 1}.

5.3 Illustrative Application

Suppose that a polymer slab is to be fabricated whose critical parameters are its density ρ and thickness t. Then the functional requirements might be specified as $F_1 = 1600 \pm 200$ kg/m^3 and $F_2 = 30 \pm 2$ mm.

These specifications are encoded as the first component of the casebase shown in Figure 5-4. For example,

```
fr(slab, [1,1600,200]).
```

indicates that the 1st functional requirement for the polymer slab is 1600 ± 200 (in units of kg/m^3).

Consider the following three candidate designs for fabricating the polymer slab, as portrayed in Figure 5-5.

1. *Design dl: Extrusion.* The polymer slab is extruded through a rectangular aperture, moving left to right in the diagram shown in Figure 5-5(a). Let P be the upstream pressure and T the height of the orifice. The design parameters may then be defined as $D_1 ::= P$ and $D_2 ::= T$. This design configuration is assumed to produce a polymer slab of density $F_1 ::= \rho = 1700$ kg/m^3 and thickness $F_2 ::= t = 31$ mm.

   The density ρ of the polymer slab is to be controlled[14] by the pressure P and the thickness t by the aperture T. Since the slab will expand slightly after passing through the orifice, the actual thickness will be a shade greater than the gap size T. Moreover, a change in P or in T will affect both ρ and t. Since the coupling coefficients are given by [see Appendix C]:

   $$c_{ij} = \partial F_i / \partial D_j$$

   we conclude that $c_{12} \neq 0$ and $c_{21} \neq 0$.

[14] In reality, the pressure affects the extrusion velocity much more than the material density.

```
/* Secondary Level */

/** Feasibility Check  **/

        feas(X) :- meetFR(X,1), meetFR(X,2).

        meetFR(X,I) :-
                fr(slab, [I,F,DF]),
                Lowval is F - DF,
                Hival is F + DF,
                dFR(X, [I,Ival]),
                Lowval =< Ival,
                Hival  >= Ival.

/** Independence Measure: consists of 3 cases **/

        /** Cluttered coupling matrix
            --> coupled design          **/

        idf(X,Idm) :- cCoef(X, [I1,J1]), I1 < J1,
                      cCoef(X, [I2,J2]), I2 > J2,
                      Idm is -1, !.

        /** Lower or upper triangular matrix
            --> decoupled design        **/

        idf(X,Idm) :- cCoef(X,[I,J]), notEqual(I,J), Idm is 0, !.

        /** Otherwise we have diagonal matrix
            --> uncoupled design        **/

        idf(X,Idm) :- Idm is 1.
```

Figure 5-2:Feasibility and Independence components of the Secondary Level.

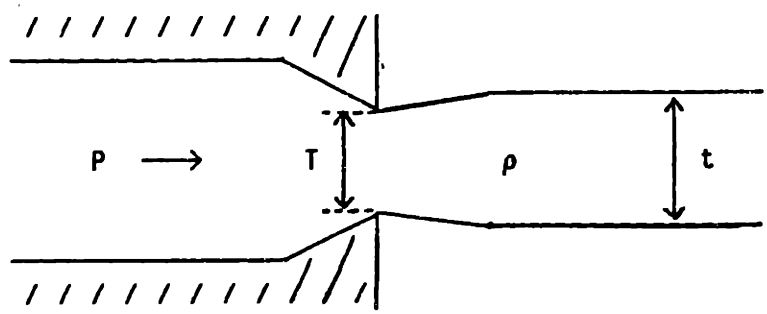
```
/* Secondary Level */

/** Information Measure **/

        /* Main information function */

        iff(X,Ifm) :-
                surff(X,Surfm),
                geomf(X,Geomm),
                Ifm is (Surfm + Geomm).

        /* Information measure due to surface tolerance */

        surff(X,Surfm) :-
                surfd(X,List),
                infoLM(List,Surfm).

        /* Information measure due to geometric tolerance */

        geomf(X,Geomm) :-
                geomd(X,List),
                infoLM(List,Geomm).

        /* Information attributes:
           To convert from LIST of n pairs of (U,DU)
           to corresponding MEASURE */

        infoLM(List,Meas) :- infoLM1(List,0,Meas).

        infoLM1([],Oldm,Newm) :- Newm is Oldm.
        infoLM1([U,DU|L2],Oldm,Newm) :-
                Ratio is U/DU,
                lb(Ratio,Mesom),
                Sum is Oldm + Mesom,
                infoLM1(L2,Sum,Newm).
```

Figure 5-3:Information component of the Secondary Level.

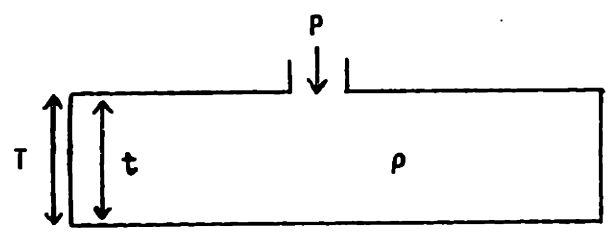
```
/* Casebase */

/** Problem Specification **/

        fr(slab, [1,1600,200]).
        fr(slab, [2,30,3]).

/** Candidate Designs **/

        /* Design d1 */

        dFR(d1, [1,1700]).
        dFR(d1, [2,31]).
        cCoef(d1, [1,2]).
        cCoef(d1, [2,1]).

        surfd(d1,[80,10]).
        geomd(d1,[160,5,640,10,320,5]).

        /* Design d2 */

        dFR(d2, [1,1500]).
        dFR(d2, [2,33]).
        cCoef(d2, [1,2]).

        surfd(d2,[40,10]).
        geomd(d2,[80,5,160,10,160,5]).

        /* Design d3 */

        dFR(d3, [1,1550]).
        dFR(d3, [2,28]).
        cCoef(d3, [1,2]).

        surfd(d3,[40,10]).
        geomd(d3,[320,10,160,10,160,5]).
```

Figure 5-4:Casebase for the Polymer Slab Problem.

(a) Design d1: Extrusion.



(b) Design d2: Injection Molding.



(c) Design d3: Powder Liquefaction.

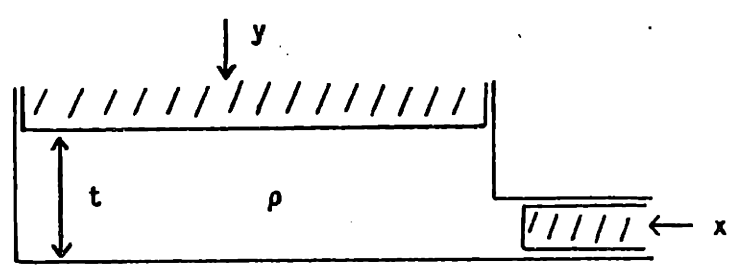


Figure 5-5: Alternative designs for fabricating a polymer slab of density $\rho \pm \Delta\rho$ and thickness $t \pm \Delta t$.

2. *Design d2: Injection Molding.* The polymer is injected into a mold through a small aperture at the top. Let D_1 ::= P, the pressure at the entrance to the aperture, and D_2 ::= T, the height of the mold.[15] The actual operating points for this design are F_1 = 1500 kg/m^3 and F_2 = 33 mm.

   The pressure P affects density ρ but not the thickness t. In contrast, a change in T affects both t and ρ. We conclude that $c_{21} = 0$ and $c_{12} \neq 0$.

3. *Design d3: Powder Liquefaction.* The polymer in powder form is poured into a special mold fitted with moving rods in both the vertical and horizontal axes. The apparatus yields a slab of density F_1 = 1550 kg/m^3 and thickness F_2 = 28 mm.

   Let D_2 ::= y be the height of the vertical ram designed to control the thickness t, and D_2 ::= x the position of the horizontal ram used to control density ρ. Then y affects both ρ and t, while x affects only t. Hence $c_{21} = 0$ but $c_{12} \neq 0$.

The data pertaining to all three designs are given in Figure 5-4. For example, cCoef(d1,[1,2]) implies that c_{12} for design d1 does not vanish.[16] Similarly, dFR(d1, [2,31]) stipulates that the second functional requirement yielded by design d1 is 31 (in units of mm.).

The data structures relating to surface and geometric tolerances are identical to those discussed in Subsection 5.2.2. For example, surfd(d1,[80,10]) implies that the surface tolerance information for design d1 is 80 ± 10 microns.

---

[15] The design as described may not be satisfactory: in expanding into the cavity, the polymer material would tend to distribute itself anisotropically (i.e. with uneven density). This problem may be rectified by using an expanding platform. For example, the bottom face of the mold may be positioned initially near the top surface, then expanded slowly in unison with the incoming fluid until the final mold height T is reached.

[16] Since the ith design parameter is intended to satisfy the corresponding functional requirement, we may reasonably assume that $c_{ii} \neq 0$ for all i. Hence we will not explicitly include cCoef(X,[I,I]) in the casebase for the diagonal elements c_{ii} of any design X. The design of CAS, however, allows for the inclusion of the cCoef(X,[I,I]) without affecting system behavior.

5.4 Operating Modes of CAS

CAS may be used to address different types of queries. The three modes of operation are as follows:

1. *Basic.* Example queries are "Is design X acceptable?", or "Is design X superior to Y?".

2. *Information retrieval.* An example is "What is the information content of design X?"

3. *Enumeration.* Examples are "Give me the list of all feasible designs", or "Which designs are superior to which others?"

These categories are clearly related, yet conceptually separable.

To render the discussion more concrete, these mode of operation are discussed in the context of the preceding example.

5.4.1 Basic Mode

This operating mode is illustrated in Figure 5-6, which is excerpted from an interactive session. In this figure, input from the user is marked by underlining.

```
| ?- sup1(d1,d2).

no
| ?- sup1(d2,d1).

yes
| ?- sup1(d1,d3).

no
| ?- sup2(d1,d2).

no
| ?- sup2(d2,d3).

yes
```

Figure 5-6: Basic mode of operation.

In response to the "?-" prompt from Prolog, the first item of user input is "sup1(d1,d2)": this corresponds to the question, "Is design d1 superior to design d2 according to Axiom 1?" Through a process invisible to the user, Prolog accepts the input and matches the d1 against the X and the d2

against the Y in Axiom 1. This goal then generates several subgoals. A check against the casebase confirms that d1 and d2 are both feasible since they meet the functional requirements within their respective tolerance bands. However, the independence measure of d1 is less than that of d2. Hence Prolog responds with a "no" to indicate that d1 is not superior to d2 by Axiom 1.

The second query in the figure asks the oppositive question, "Is design d2 superior to design d1?" It is met by a positive response.

The third query and its reply indicate that design d1 is not superior to design d3 by the first Axiom. The fourth query and its response reveal that, according to Axiom 2, design d1 is not superior to d2 (since the former has more functional coupling than the latter), while the last exchange indicates that design d2 is superior to d3.

5.4.2 Information Retrieval Mode

The second mode of operation relates to the acquisition of specialized information. Some examples of this type of usage are given in Figure 5-7.

```
| ?- idf(d1,Idm).

Idm = -1

yes
| ?- idf(d2,Idm).

Idm = 0

yes
| ?- iff(d2,Ifm).

Ifm = 15

yes
| ?- iff(d3,Ifm).

Ifm = 16

yes
```

Figure 5-7:Information retrieval mode.

The first query corresponds to the question "What is the independence measure of design d1?" Prolog replies that this value is -1 (corresponding to a coupled design).

The second query and its response indicate that design d2 has the Idm value of 0. The last two queries and their responses show that the information contents for designs d2 and d3 are 15 and 16 bits, respectively.

5.4.3 Enumeration Mode

The third mode of operation relates to an enumeration of objects with a particular set of characteristics. Such enumeration may result in a partial or complete list of the relevant set.

This mode is facilitated by the *backtracking* characteristic of Prolog. When Prolog fails to find a solution to a query on the first pass through the database, it retraces its steps and attempts to use alternative candidates. Even when a solution is found, Prolog saves its pointers to the database so that alternative solutions may be obtained if desired. Such a request is communicated to Prolog by typing a semicolon (;). Each time Prolog sees a semicolon in response to a generated solution, it goes back over the database and attempts to find a new solution.

Figure 5-8 illustrates this usage. The first query, "sup1(X,Y)", corresponds to the question "Which designs are superior to which others according to Axiom 1?" Prolog first replies that design d2 is superior to design d1.

The subsequent user input consisting solely of a semicolon is interpreted as a request for an alternative solution. Prolog's response is that d3 is also superior to d1. When the semicolon is presented a second time, Prolog is unable to find any more solutions and therefore responds with a "no".

The query "sup2(X,Y)" is equivalent to asking "What design X is superior to which design Y according to Axiom 2?" Prolog replies that design d2 is superior to d3. In reponse to the subsequent semicolon prompt, Prolog attempts to find another such pair. But there is no other such pair, and Prolog replies in the negative.[17]

---

[17] An alternative method for obtaining a complete enumeration is to use the built-in predicate called setof [Clocksin and Mellish 1981].

```
| ?- sup1(X,Y).

X = d2,
Y = d1 ;

X = d3,
Y = d1 ;

no
| ?- sup2(X,Y).

X = d2,
Y = d3 ;

no
```

Figure 5-8:Enumeration mode.

5.5 Flexibility of Design

The modular structure of CAS allows for system expansion by simply scaling up in obvious ways. This modularity springs from two sources:

1. *Layered system design.* The system is modular by construction. The inference engine, for example, is separate from the casebase.

2. *Hierarchical interpretation of Prolog.* Clauses in Prolog encourage modularity through hierarchical decomposition. Each literal in the body of a clause may be expanded into a set of literals which in turn are expanded recursively.

The modularity engendered by the second characteristic may be illustrated by refering to the Prolog statement of Axiom 2. Suppose that CAS is to be enhanced by including a new type of information called process information. This may be achieved by adding two new predicates called procd and procf.

The predicate procd(X,L) specifies the list L of process data pertaining to design X in the casebase. On the other hand, the function procf(X,Procm) operates on procd to yield the corresponding information measure Procm. The iff clause is modified by simply incorporating this new predicate (including the Procm variable) in the body. In other words, the new clause is given by

```
iff(X,Ifm) :-
        surff(X,Surfm),
        geomf(X,Geomm),
        procf(X,Procm),
        Ifm is (Surfm + Geomm + Procm).
```

The general structures presented in Subsections 5.2.2 and 5.2.3 are sufficiently flexible to accomodate extensions such as these.

5.6 Architecture for a full-fledged Expert System

The ability to state the Axioms as clauses in a logical programming language suggests that it would be feasible to develop an expert system for axiomatics [Horstmann 1983; Mizoguchi 1983; Oliveira 1984; Walker 1983]. This section discusses some issues relating to system capability and architecture.

5.6.1 System Capabilities

Experts systems may be distinguished, among other parameters, by the degree of autonomy and of capability. Autonomy relates to the amount of interaction required with a human user, while capability pertains to the range of system behaviors. Obviously these two characteristics are interrelated. On the one hand, an expert system when invoked by a user may run to completion by itself; on the other hand, the system might be driven entirely by a series of requests from a human user.

The generate-and-test methodology to be used by the first type of system represents the ultimate goal of the axiomatic approach to design. But for problems of reasonable complexity, such an expert system may be impractical until the development of subsystems capable of encoding knowledge in various realms of engineering. And this may have to await the development of programs fully capable of self-instruction.

5.6.2 System Architecture

A potential system architecture designed to support an expert system for manufacturing axiomatics is shown in Figure 5-9. The heart of the system is, of course, the axiomatics expert consisting of the Primary, Secondary and Data Levels. These components are augmented by subsystems for explaining system behavior and for handling errors.

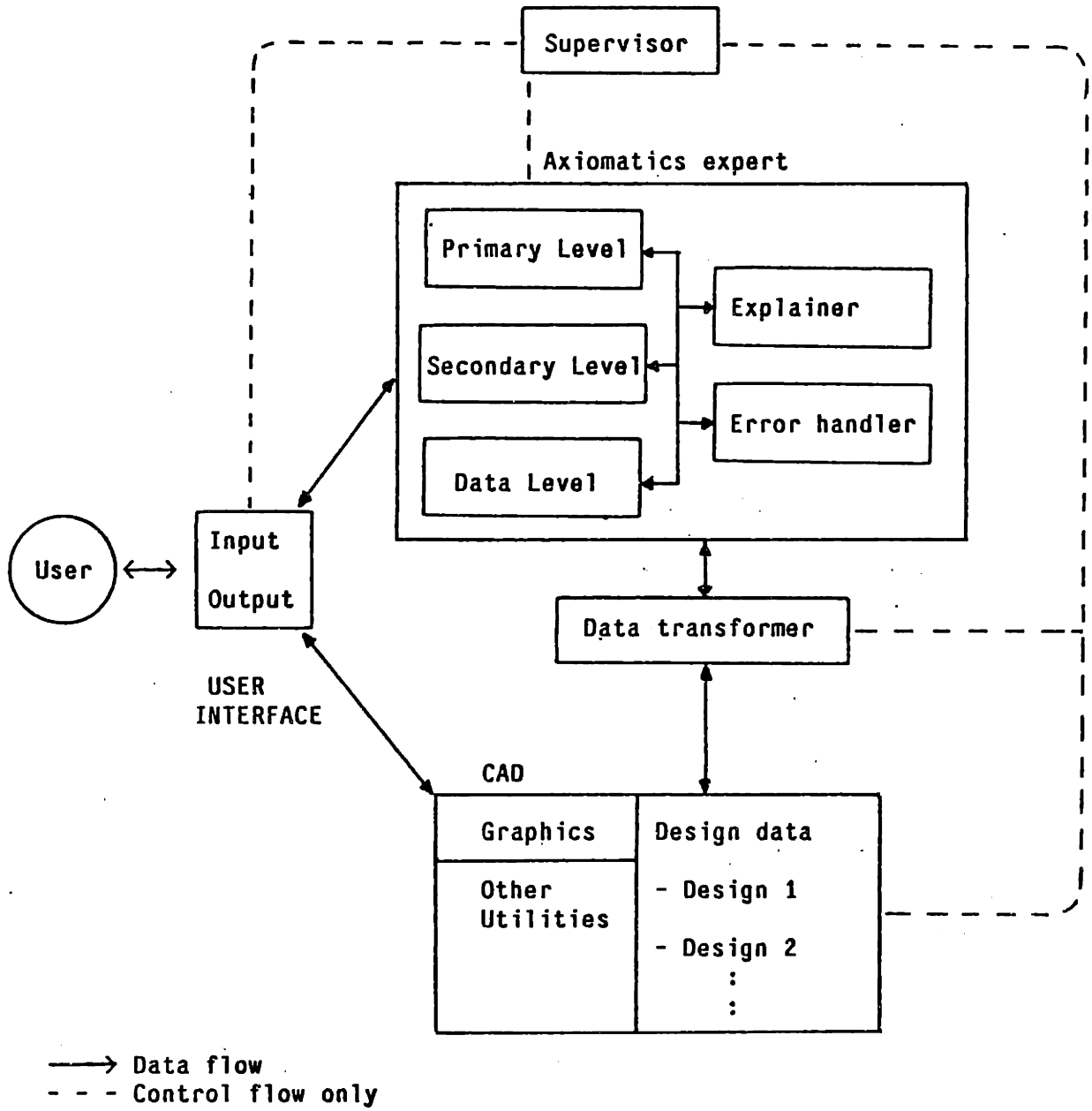


Figure 5-9:Overall architecture for an axiomatics expert system.

The user may interact with the system through a friendly interface as well as the graphics capabilities of a computer-aided design (CAD) package. The core axiomatics expert and the CAD systems communicate through the Data Transformer, a package that converts the data for alternative designs from CAD-compatible to CAS-compatible format and vice versa.

Chapter Six

Integration of Analysis and Synthesis Phases

This chapter discusses the way in which the analysis and synthesis phases may be combined. This is accomplished by using the Information Minimization Principle as a vehicle for discussion.

6.1 Managing Complexity in Design

System design seems to be an increasingly complicated task. The fifth-generation computer, for example, will be built from chips containing over a million transistors. At existing rates of design productivity, however, the layout for a million-transistor chip would take between 40 to 800 workyears [Trimberger 1983].

There are three main techniques for managing such complex design tasks:

1. *Hierarchical decomposition of design.* The design may be partitioned into successively smaller modules until the pieces are simple enough to understand and design for. Such a top-down decomposition of design requirements is tantamount to specifying a hierarchy of functional requirements [Yasuhara 1980].

   In addition to problem simplification, functional decomposition allows for the use of previous solutions or components. In automotive design, for example, the Pontiac Fiero uses components from the older Firebird and other models.

   While each module may be studied in relative isolation, however, it is still important to take into account its relationships to other modules. The individual modules must have mutually compatible interfaces when they are merged together.

2. *Regular structures.* Replication allows for the amortization of development cost, whether in time or money, across multiple units. A regular component has to be designed once, but may be used many times over. Since a standard component will have standard interfaces, the reduction in effort may be traced both to the design of the component and

its interfaces.

3. *Automation.* The complete displacement of design engineers by computer technology is not likely to occur in the forseeable future. Computers may, however, complement designers by taking over routine tasks and assisting in others. They may be used to perform calculations, manipulate images, prepare blueprints, and check for data consistency. Computers may also be used transform one model into another, as in mapping a tree of functional requirements into its corresponding matrix.

The techniques discussed above may be synergistic rather than mutually exclusive. For example, a computer system may help the designer construct the tree of functional requirements; the tree in turn might identify regular structures which may be satisfied with standard components.

6.2 Interpretation of Axiom 2 in the Dynamic Case

A characteristic of relativistic physics is the treatment of time as merely another coordinate for locating objects in a 4-dimensional coordinate system. In engineering and classical physics, however, it is usually convenient to categorize time and space variables separately. We will take the latter perspective in this section.

The temporal dimension can be further subdivided into two categories, *static* and *dynamic*. A static attribute or situation is one which is independent of time, while a dynamic context is not. For example, the geometric information associated with the physical dimensions of an engine is static. On the other hand, the attitude control system for a supersonic plane must process information dynamically in order to maintain a flight mode which is inherently unstable.

The original emphasis of the Information Minimization Principle was on the static case [Suh et al. 1978a]. For example, a tool of simple geometric configuration is considered to be a superior design to one with a complex geometry, as long as the functional requirements are served. In other words, enough information should be specified to fulfill the functional requirements (effectiveness), but not any more than necessary (efficiency).

The Information Axiom was applied to a quasi-dynamic situation in the case of process planning [Nakazawa and Suh 1984]. In this application, information for determining a work-process schedule

is determined by the tolerances specified by a designer and those due to the physical plant. This situation is quasi-dynamic in the sense that the information can be processed in batch mode: both tolerances are static quantities which need not be considered in real time, although the application area deals with process planning.

We now explicitly consider the dynamic case as well as the static case for information minimization. Axiom 2 requires that information be minimized subject to the satisfaction of functional requirements:

> **Axiom 2.** Minimize information requirements subject to the fulfillment of functional requirements.

If we assume that information can assume both static and dynamic forms:

> **Postulate.** Static information and dynamic information are subsets of information.

then we obtain the following result,

> **Corollary 2.** Minimize static and dynamic information requirements subject to the fulfillment of functional requirements.

This result is named *Corollary 2* since it is an immediate consequence of Axiom 2 plus the preceding postulate.

If we accept the following hypothesis:

> **Postulate.** Information content relating to design or process specification, is a subset of static information requirements.

then combining it with Corollary 2 yields

> **Corollary (Information Content).** Minimize information content subject to the fulfillment of functional requirements.

In a similar way, acceptance of the following hypothesis

> **Postulate.** Information processing requirements are a subset of dynamic information requirements.

in conjunction with Corollary 2 leads to

> **Corollary (Information Processing).** Minimize information processing requirements.

Note that the Information Content Corollary focuses on the static situation while the Information Processing Corollary applies to the dynamic case.

6.3 Information Reduction

Systems, by definition, have linkages. Unfortunately, the number of linkages tends to expand combinatorially with the number of components in an elaborate system, and thereby to impede efforts toward rational design. This perennial problem is addressed by assembling the components into *clusters* or *modules*. The two main approaches to modularity involve hierarchies and layers. These two types of structures are discussed below, following the derivation of an information measure for structural complexity.

6.3.1 Information Measure for Organizational Complexity

We may define an information measure for organizational complexity based on the probabilistic interpretation of information [cf. Appendix D]. Suppose that a system organization may be represented by a particular structure (examples of which are given in the following subsections) consisting of N links. Moreover, the system succeeds in fulfilling its function if and only if each link succeeds (works), where p_i is the probability that the ith link works.

In that case p_s, the probability of success of the entire system, is given by

$$p_s = \Pi_{i=1,..,N} \; p_i$$

where N is the number of links. The information associated with this event is

$$I' = -\,lb\, p_s = -\,lb\,(\Pi_{i=1,..,N} \; p_i)$$

where *lb* denotes the logarithm to base 2. If the probability of success of each node is fixed and has value p, then

$$I' = -\,N\, lb\, p$$

Suppose we normalize I' by the probability of success of each link; then the information measure takes the form

$$I ::= I' / (-\,lb\, p) = N$$

In other words, organizational information is proportional to the number of linkages.

6.3.2 Hierarchies

The notion of hierarchies is central to the integration of the analysis and syntheses phases, since it permeates both domains. Hierarchical structures are found, for example, in the organization of industrial plants when analyzing manufacturing systems, as well as in function trees generated when synthesizing designs.

A hierarchy is a special type of lattice with unique successor elements, a single maximal member and no self-loops [see Subsection 2.2.2]. The number of predecessors of a given element or *node* y is called the *span* of y. Consider a hierarchy H with levels 0,1,...,n, where level n denotes the maximal element known as the *root* or *top*.

In general different nodes may have differing spans. For the sake of simplicity, we assume hereafter that H has a fixed span s for each node. In that case H has $m_i = s^{n-i}$ members at level i. The number of links in H, and therefore the organizational information, is given by

$$N_H = \Sigma_{i=0,1,..,n-1}\ s^{n-i} = \Sigma_{i=1,..,n}\ s^i$$

On the other hand, consider a system with n components which are totally connected. The first element can form n-1 bonds with the other components, the second can form n-2 bonds, and so on. The total number of links is

$$N_T = n(n-1)/2$$

Information content expands quadratically with the number of components in a totally connected system, but more slowly for a hierarchical organization. Hence a hierarchical structure in system design is compatible with the Information Axiom.

If the hierarchical organization is taken to be sufficient for satisfying the functional requirements, then the ratio of information requirements is

$$\eta = I_H/I_T = N_H/N_T$$

To illustrate, take a small hierarchy with $m_0 = 32$ elements at the bottom level, organized into a binary tree (i.e. the span s is 2). This hierarchy has $N_H = 62$ links, while the totally connected arrangement would have $N_T = 32*31/2 = 496$, resulting in an information ratio of $\eta = 0.125$. In other words, the hierarchical organization requires only 1/8 the number of links needed for the totally connected system.

6.3.3 Layers

Layering involves the stratification of system components into different slices or levels. A component in one layer can interact only with the adjacent layers, with which it communicates through a standard interface. The purpose of such an interface is to serve as a gateway which mediates all interlayer communication and to hide the structure of each layer from its neighbors.

A prime example is found in the levels of computer languages. Consider the progression of languages defined by a machine language M, assembly language A, procedural languge P (e.g. Fortran or Pascal), and a nonprocedural language N (e.g. a program specification language). The sequence <M,A,P,N> represents a total order among the languages, each of which represents a different layer. For example, the statements in a Fortran program need to interact only with each other; the linkages among the corresponding assembly language statements are hidden by the Fortran-to-assembly interface, known as a compiler.

Another example of layering is found in the Open Systems Interconnection Reference Model adopted by the International Standards Organization as a telecommunications protocol. This system consists of 7 levels starting from the physical level pertaining to signal interpretation, all the way up to the applications level dealing with specific end-user packages.

Information Reduction

Consider a system composed of layers $L_1,...,L_n$. Let L_i and L_{i+1} be two consecutive levels, with m_i and m_{i+1} nodes respectively.

Since each component in one layer may form a bond of a particular kind with a component in another layer, the total number of bonds of that kind is $m_i m_{i+1}$. Since there are n levels, and bonds can form only between components in adjacent layers, the maximum number of links for the system is

$$N_* = \Sigma_{i=1,..,n-1} m_i m_{i+1}$$

For a small system consisting of 10 components in 3 levels, we have $N_* = 200$ and $I_* = 7.64$ bits.

In contrast, assume that all communications between consecutive levels occur through only one interface, as shown in Figure 6-1. The m_i components in level L_i can form m_i bonds of a particular kind with the interface. Similarly, the components in level L_{i+1} can form m_{i+1} bonds with the interface. Hence the total number of interlevel bonds between adjacent layers is $m_i + m_{i+1}$. Since

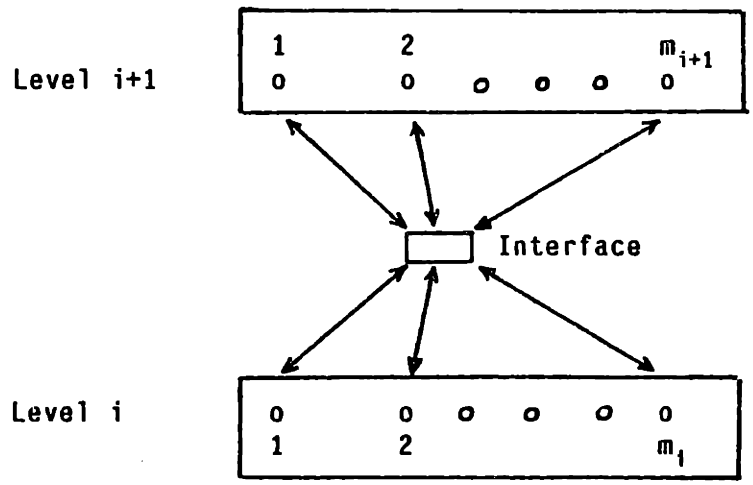


Figure 6-1:Adjacent levels L_i and L_{i+1} for a layered architecture.

there are n levels, the maximum number of bonds--as well as the organizational information--is

$$N_L = m_1 + m_n + 2\,\Sigma_{i=2,..,n-1}\, m_i$$

When the number of components at each level is the same, then $m_i ::= m$ for all i, and $N_L = 2m(n - 1)$.

The information ratio of the layered system with respect to the maximally connected one is

$$\eta_L = I_L/I_T = N_L/N_T$$

To illustrate, consider a system with $m_i = 10$ components at each of 3 levels. The information and linkage are calculated as $I_L = N_L = 40$. For the totally connected system without any kind of modularity, the corresponding figure is $I_T = N_T = 29*30/2 = 435$. The resulting information ratio is $\eta_L = 0.092$.

6.4 Information and Purpose

The need for information to fulfill purpose or function has been discussed in Section 4.3. If we assume that products and processes are designed to serve specific functions, then entire manufacturing systems as well as their components may be viewed as purposive systems. This viewpoint suggests the following hypothesis:

Postulate. Information minimization is an organizing principle for purposive systems.

A purposive system may take the form of hardware (e.g. polymer processing machine) or of software (e.g. process planning system). Increasing numbers of devices are of hybrid form (e.g. microwave ovens or robot sentries).

Information minimization may be viewed as a design guideline from two levels [see Section B.4 of Appendix B]:

1. *Object level, pertaining to the designed product or process.* An example lies in the proposition "Minimize the number and complexity of part surfaces."

2. *Meta level, relating to the process of design.* An illustration is found in "Minimize the number of functional requirements."

The Information Axiom at the object level implies that information requirements must be minimized in a designed system, while the Axiom at the second level implies that the design process should avoid processing excess information.

In either case, information minimization must of course be consistent with the functional requirements relating to the object system or the design process. Otherwise information minimization would call for a "purposive" system with zero information, and therefore zero capacity to fulfill any purpose -- a contradiction in terms.

Information minimization appears to be a plausible organizing principle for purposive systems in general. Certainly, information maximization could not be a useful principle. A purposive system must maximize information necessary to achieve its objectives; but any excess can only impede its progress. An analogy may be drawn to structural reinforcement in an aircraft: metallic alloys are needed to reinforce the structural integrity of the plane, but excess metal will only result in a slothful contraption which may not fly at all.

6.5 Information Minimization for an Automaton

We recall from Chapter 3 that an automaton is specified by the structure $\mathcal{A} = \langle S,\Sigma,\Omega,\star,M,N,s_0,F,\sigma_0,\omega_0\rangle$. Since $\star$ is an operator while s_0, σ_0, and ω_0 each represents an element, we are left with 6 coordinates which constitute sets.

These coordinates may be viewed as static and dynamic attributes in the following way. The state set S, input alphabet Σ, output alphabet Ω, and final state set F are fixed characteristics of the automaton. Hence reducing the numerosity of these coordinates results in a reduction of static information requirements I_s.

Let X be a dummy variable denoting any of the 4 original sets S, Σ, Ω, or F. Moreover X' is taken to be the new, reduced set corresponding to X; and |X| the cardinality or number of elements in X. The the static information reduction is given by

$$\Delta I_s = lb|X| - lb|X'| = lb(|X|/|X'|)$$

On the other hand, the input transition function M and the output transition function N represent both static and dynamic characteristics: in a practical automaton, these specifications would be encoded in software as lookup tables for consultation in real time. The static information reduction in *encoding* these tables is given by

$$\Delta I_s(\text{encode}) = lb|X| - lb|X'| = lb(|X|/|X'|)$$

where X represents M or N.

However, the time spent in table *lookup* is a dynamic characteristic. The information associated with this activity depends on the mode of search.

1. For *linear search* of a lookup table of length n, the average number of steps is n/2. Hence the expected processing speed improves by (n - n')/2, where n' is the new, reduced length of the table. For example, halving the table length from n to n/2 speeds up the processing by n/4 steps.

2. For *binary search* of an ordered table, the number of steps is approximately lb(n). Hence halving the table size, for example, would increase processing speed by lb(n) - lb(n/2) = lb 2 = 1 step. Although the number of lookups increases only logarithmically with n in binary search, there is overhead associated with the maintenance of an ordered (sorted) table. When the table itself is updated only infrequently, this overhead is of minor consequence.

3. Using a *hash method* of table lookup results in an expected number of steps which only slightly exceeds 1. When the table is sparse compared to the memory used, the desired item will usually be "hit" or found on the first try; if not, one or two more steps will be

required by proceeding to overflow memory banks. The hash method is wasteful of memory, a good fraction of which must remain idle to maintain a high "hit rate". But this cost must be borne if rapid access is imperative.

The static and dynamic information reduction aspects are not independent. An important characteristic of the input and output functions (M and N) is that they tend to shrink automatically with a reduction in the state set S as well as the input and output alphabets (Σ and Ω).

For example, the input transition table must encode the function f: $S \times \Sigma \rightarrow S$. Each transition specification is defined by a triple $\langle s,\sigma,s' \rangle$ for s, s' $\in$ S and $\sigma \in \Sigma$. For this reason M tends to decrease linearly with Σ and quadratically with S. That is, $|M| \propto |\Sigma|$ and $|M| \propto |S|^2$. In a similar way, N tends to shrink linearly with Ω and quadratically with S.

Shorter tables imply a reduction in the static information required for encoding the transitions. A consequence of having smaller tables is that the dynamic information required for table lookup will also decrease.

6.5.1 On The Relationship Between Axioms 1 and 2

Axiom 1 calls for a separation of functional requirements. In general, each functional requirement may be subdivided into lower-level functions, which in turn may be subdivided [Yasuhara 1980]. Axiom 2 asserts the advisability of minimizing information requirements.

When the functions can be partitioned, information requirements corresponding to interactions are eliminated. Moreover, when the functions can be decomposed into a hierarchy, the information requirements for the overall design are reduced. This point is aptly demonstrated by the tale of the watchmakers [Simon 1969]. The watchmaker who divides his work into subassemblies can complete his watches in a small fraction of the time required by the one who does not. The subject of the relationship between Axioms 1 and 2 is taken up further in Appendix E.

6.5.2 "Conservation" of Minimal Information

Axiom 2 calls for a minimization of information in the design of a product or process. Since the design must satisfy the functional requirements, we can safely assume that some positive amount of information is required. Let's call this minimal amount of information I_0.

If we accept the proposition that there may be more than one optimal solution, we are led to the conclusion that I_0 takes on the same value for different types of optimal designs. More precisely, let I_{0j} be the minimal information required among feasible designs in the jth category. Then:

> Postulate (Minimal Information). The information content I_{0j} for an optimal design in category j is of the same order as that for optimal designs in other categories.

In other words, $O(I_{0j}) = O(I_0)$ for all j, where O(*) denotes the order of *.

There is some support for this hypothesis in the area of shape control for metal bending processes [Hardt 1982]. This paper discusses the tradeoff between different measurement strategies and the attendant complexity of the control model.

The operation of a metal bending process requires knowledge of the moment distribution in the sheet, usually assumed to be proportional to the length along the arc, and the relationship between the curvature and moment for the given workpiece material. These relationships may be determined a priori and incorporated into an elaborate model, or measured in real time and processed through a simpler model. The control schemes are as follows:

1. Open loop control, requiring a detailed model and representing offline control.

2. Hybrid of open loop control of the workpiece shape and online, closed loop identification of plant parameters.

3. Closed loop shape control, using indirect measurements of the workpiece configuration.

For comparable performance among the three control schemes, an increase in the directness of measurements permits a decrease in the complexity of the model representing the process. Moreover, an increase in the operating information entails increased costs but this drawback is countered by the ease of developing a simpler model and of robustness in the face of errors in model structure or calibration.

* * *

The investigation of the Information Minimization Principle and its consequences has extensive theoretical implications for the design of both hardware devices such as servomechanisms and software components such as knowledge systems. Moreover, since information costs comprise a large fraction of the total manufacturing cost [Suh 1984], the results of this type of study have significant

practical implications as well.

Chapter Seven

Conclusion

The overall conclusion to be drawn from this work is that the foundations of mathematics ably serve as foundations for a science of manufacturing. More specifically, set theory and symbolic logic in conjunction with the mathematical constructs they engender, constitute a useful framework for manufacturing science. These structures are instrumental for interpreting qualitative notions with precision, for serving as a unifying framework for specialized theories, and for facilitating efforts toward the computerization of design and manufacturing.

This chapter first summarizes the results of the earlier discussions, then maps out some directions for future research.

7.1 Summary

Chapter 1 presented the need for frameworks for manufacturing science, both in terms of conceptual structures and mathematical constructs to unify and formalize a complex field. The discussion continued with previous research and their shortcomings, then offered a conceptual framework for the study of manufacturing systems along 3 dimensions: formality (informal versus formal), phase (analysis versus synthesis), and quantification (qualitative versus quantitative). In particular the notion of quantification, relating to numbers and equations, was shown to be separable from that of formalization, pertaining to levels of rigor.

Chapter 2 presented the foundations of mathematics. Symbolic logic was introduced informally, then discussed more formally. This was followed by an encapsulation of set theory, which was then used to define the concept of an algebraic structure as an ordered pair <S,R>, where S is a set of objects and R a set of relations defined on S. This definition in turn served as a unifying construct for various structures such as graphs, lattices, and groups. The relationship of homomorphism and isomorphism to the construction and analysis of models was also discussed.

Chapter 3 dealt with the infrastructure for the analysis phase by building on the algebraic constructs

of the previous chapter. The systems framework was presented in algebraic terms, followed by a discussion of system representations in terms of graph, matrix and state space techniques. Since algebraic structures are compatible with the state space approach, they also serve as a unifying groundwork for quantitative theories.

A classification of system models consisting of 3 categories was introduced, according to black, grey and beige boxes. Beige boxes arise when black or grey boxes are brought together to form richer models. The models of automata theory were shown to be grey boxes, while those of control theory and information transmission channels were shown to be generable from black and grey boxes.

Chapter 4 was concerned with the application of the foregoing infrastructure to the analysis phase. A characterization of work activities was presented, followed by the definition of a general manufacturing system in terms of the systems framework. The nature of complexity and its frequent manifestation in hierarchical form was discussed. The need for information to satisfy useful purposes was indicated, followed by a generalization of the classifical information model to take into account some measure of semantic information. Information characteristics were identified in terms of effectiveness, efficiency, and timeliness, followed by a discussion of their impact on manufacturing performance.

The prime importance of information in manufacturing systems highlights the need to model this component adequately. This may be done by drawing on the framework of automata theory, initially introduced in Chapter 3. Some representative results of this unification were presented in the form of theorems.

The applicability of symbolic logic to the synthesis phase was demonstrated in Chapter 5. This was accomplished by providing rigorous versions of the manufacturing axioms based on predicate calculus. The theoretical and practical consequences were shown to be as follows.

1. *Theoretical implications.* The language provides a tool for studying with greater exactitude the relationships among the Axioms and their derivatives. In particular, statements which were previously believed to be corollaries may actually be partitioned into two classes.

    a. One proposition was shown to be a direct consequence of Axiom 1. In fact, it is an alternative statement of Axiom in an informal sense, but is identical in a formal sense.

b. Other propositions, previously believed to be descendants of the Axioms, were proved to belong to a separate lineage. This class of indirect consequences contains propositions, each of which would require the acceptance of at least one other assumption.

2. *Practical implications.* The precise statements make it possible to encode the axioms in software.

   a. The Axioms may be stated in a logical programming such as Prolog.

   b. A modular program schema called the Computerized Axiomatic System (CAS) was developed. Further, the operating modes of CAS were described in the context of a small example.

Chapter 6 addressed the issue of integrating the analysis and synthesis phases of manufacturing at the applied level, in contrast to the mathematical level discussed in earlier chapters. This was done by considering the Information Minimization Axiom as a representative--as well as vital--topic of discussion. Axiom 2 was extended to include explicitly the dynamic case in addition to the static case. Information reduction was discussed in terms of the two major approaches to organization, the hierarchical and layered forms. Moreover the automata-theoretic model presented in Chapter 4 was interpreted in terms of the information minimization principle.

7.2 Future Directions

The benefit of a conceptual framework for manufacturing science is that it suggests future directions for research in addition to providing a skeletal structure on which to hang specialized theories. The framework given in Chapter 1 may be used to infer directions for future research in terms of phase, formality and quantification. More specifically, each of the 8 regions generated by the 3-fold Cartesian product {Analysis, Synthesis} x {Informal, Formal} x {Qualitative, Quantitative}, may be explored systematically.

The research effort in each of these regions may be further directed at a number of levels:

1. Abstract level, consisting of mathematical foundations and models for manufacturing systems. A sample issue lies in the impossibility of constructing an arbitrary manufacturing system.

2. Attribute level, dealing with characterizations of manufacturing systems. An example is the impact of information characteristics on system performance.

3. Empirical level, dealing with case studies and statistical tests in design and manufacturing.

This thesis has focused on Levels 1 and 2. The study at the second level has served to affirm the utility of the general abstract model developed at the first level. In the future, empirical studies at the third level may further validate the conceptual and theoretical studies at the first two levels.

For the purposes of presentation, the discussion which follows is partitioned by phase into analysis and synthesis aspects. A set of representative--but by no means exhaustive--research issues is discussed in each subsection.

7.2.1 Analysis Phase

One important area for future research is to further investigate the relationship between information minimization and functional requirements satisfaction, and the conditions under which the information minimization principle holds. At the empirical level, issues such as the following may be addressed:

1. Do different information factors (e.g. tolerance, surface finish, and process information) have differential impact on process performance?

2. In what ways can information be minimized? Under what conditions is minimization acceptable?

3. How do incomplete state measurements degrade performance?

4. Does information in excess of the minimum improve performance? If so, by how much?

5. What are the implications for adaptive process control (e.g. intelligent injection molding)?

Questions such as these may be addressed suitably by the use of statistical procedures.

To illustrate, consider the first question above. Let process, tolerance, and surface finish information

be denoted respectively as I_p, I_t, and I_s. The corresponding model is

$$P = a*I_p + b*I_t + c*I_s + \text{Error}$$

where P denotes performance. An appropriate measure of performance for a given production line may be $P = R/C$, where R denotes the production rate and C the maximum capacity under 100% facility utilization.

The null hypothesis that relates to the model above is

$$H_0: a = b = c$$

If H_0 turns out to be false, an obvious subsequent question is, "What *are* the relative weights for different kinds of information?" This may be resolved through an estimation procedure for the values of a, b and c in the model above.

An example of a query at the attribute level relates to the following information minimization issue. For a deterministic system, information should be tight in the sense that only the minimal information necessary for the given task should be processed. However, what happens in the nondeterministic case when either the "facts" or the inference rules are not known with certainty? Such cases often arise in knowledge-based systems. The common wisdom in artificial intelligence says that redundancy should be used to combat uncertainty. Some related questions lie along the lines of "How much redundancy is appropriate ?" and "How is the redundancy heuristic related to the Information Axiom ?"

An example of an investigation relating to the attribute and abstract levels pertains to the characteristics of information metrics. For example, under what conditions relating to the underlying product or process, or under what metrical transformations, are the measures invariant? This area of inquiry may benefit from the application of group theory and structure-preserving homomorphisms.

An example of a research area at the abstract level stems from the limitation of automata theory in that it focuses solely on information processing issues. While information may be the single most important component of a manufacturing system, the latter processes materials and energy as well. Hence future work may attempt to generalize automata theory and thereby develop a coherent framework for an industrial plant.

Chapter 4 presented a grey box model of a manufacturing plant as a single automaton. A richer

model would be a beige box model which treats components of the plant, such as manufacturing cells, as distinct but cooperating units. This type of study may draw on the framework of cellular automata.

Cellular automata involves the design and analysis of multidimensional arrays of cells, each of which contains an automaton [Burks 1970; von Neumann 1966]. The focus is on the nature and limits of overall system behavior under stipulated conditions such as the limitation of information flow to adjacent cells. The theory, for example, investigates how self-producing automata might be designed; such models would be especially applicable to the development of a theory for automated manufacturing facilities. Another appropriate framework on which to build, may be the study of parallel computer structures such as those found in data flow architectures.

7.2.2 Synthesis Phase

The Design Axioms have been proposed as appropriate for the design of entire manufacturing systems. An important study at the empirical level would be to determine the validity of this assertion by actually designing a manufacturing facility. Such a study may also be used concurrently to develop auxiliary design rules for the synthesis of large-scale production systems.

The axiomatic approach has been discussed as a valid methodology for all design, including software construction. Moreover, a long-term goal of this field is to build an expert system to dispense axiomatics reasoning. A set of research topics at the empirical and attribute levels, relates to the recursive investigation of these issues:

1. Development of an expert system to use axiomatic principles as generic decision making rules. A system architecture for such a package was presented in Chapter 5.

2. The implications of the Axioms for the design of knowledge-based systems. In particular, the design principles may be applied to the development of effective database architectures and control procedures.

Within the framework of these overall objectives, a variety of specific goals may be addressed. Examples of possible extensions for the future are as follows.

1. *Functions.* The DECsystem-10 Prolog available to us has no log function. Hence the lb function used in the Computerized Axiomatic System of Chapter 5 has been

implemented as a short lookup table. Such a makeshift should be rectified in a real applications system, perhaps by amending the interpreter or by using an implementation offering such a function [Pereira 1984].

More generally, an extension of Prolog to allow for functions would enhance the conceptual simplicity of certain programs, including CAS. In that case the iff predicate, for example, could be encoded as an explicit function rather than a predicate, and thereby permit Axioms 1 and 2 to be written more simply as

```
sup1(X,Y) :-
        feas(X),
        feas(Y),
        Idm(X) > Idf(Y).

sup2(X,Y) :-
        feas(X),
        feas(Y),
        Idm(X) = Idf(Y).
        Ifm(X) < Ifm(Y).
```

2. *Coupling matrix.* We may define idf(X,Idm) as an independence functor which maps a design X into its independence metric Idm. But this functor relies on coupling coefficients which cannot be generated automatically.

   Moreover, the continuous measures of functional independence--reangularity and semangularity--require the determination of analytical relationships between functional requirements and design parameters [Rinderle and Suh 1982]. At this time, the field of artificial intelligence is not sufficiently developed to permit the automatic generation of such equations from design data. In the future, however, we may expect computers to have the capability to determine coupling matrices in analytic form.

3. *Application domains.* Future work should address the issue of determining appropriate domains for prototype expert systems incorporating design axiomatics.

4. *System capabilities.* Over the intermediate horizon, at least, an expert system for axiomatics must serve as an asssistant to a designer. The system can be used to calculate measures such as information content or to determine superiority among alternative

existing designs.

Over the longer term, an expert system should be able to generate alternative designs within a limited domain. The generated designs may then be examined for feasiblility and superiority against the Axioms and their derivatives. The generate-and-test methodology of this second class of systems reflects the long-range goal of the axiomatic approach to design. For problems of meaningful size, however, such an expert system will probably have to wait until the development of suitable mechanisms for encoding knowledge encompassing diverse branches of engineering.

These are examples of investigations at the practical level.

An example of research at the abstract level relates to the further development of design axiomatics. This subject may be pursued by searching for new design principles, then incorporating them into the existing axiomatic framework.

7.2.3 Toward a Unified Theory of Manufacturing

A number of directions for future work have been presented in terms of the analysis and synthesis phases for the sake of discussion. Obviously a comprehensive theory of manufacturing must integrate these two phases as well as the two other dimensions of quantification and formality.

Chapter 6, in fact, has demonstrated how the analysis and synthesis phases may be integrated into a coherent system. This was done by focusing on the Information Minimization Axiom as a vehicle for discussion. An obvious extension is to investigate how such integration may be achieved in terms of the Functional Independence Axiom.

By building on set theory and symbolic logic in conjunction with their derivative structures, this thesis has laid the mathematical groundwork for a theory of manufacturing science. This work has taken one step in the direction of developing a unified theory, but the greater adventure lies ahead.

References

Ackoff, R.L. Towards a behavioral theory of communication. *Management Science*, v. 4, 1957-58: 218-34.

Ackoff, R.L.. Towards a system of systems concepts. *Management Science*, v. 17, 1971: 661-671.

Ackoff, R.L. & R.L. Emery. *On Purposeful Systems.* London: Travistock, 1972.

Aczel, J. & Z. Daroczy. *On Measures of Information and their Characterizations.* NY: Academic, 1975.

Ando, A., F.M. Fisher & H.A. Simon. *Essays on the Structure of Social Science Models.* Cambridge, MA: M.I.T. Press, 1963.

Ansoff, H.I. *Strategic Management.* NY: Wiley, 1979.

Ashby, W.R. *An Introduction to Cybernetics.* London: University Paperback, 1964.

Bard, Y. *Nonlinear Parameter Estimation.* NY: Academic, 1974.

Barocci, T.A., M.K. Primak & K.R. Wever. Productivity improvement programs in mature manufacturing industries. Sloan School of Management, W.P. 1356-82, M.I.T., 1982.

Bartlett, M.S. *Probaility, Statistics, and Time.* NY: Wiley, 1975.

Battani, G. & H. Meloni. Interpreteur du language de programation PROLOG. Groupe de l'Intelligence Artificielle, U.E.R. de Luminy, Marseille, France, 1973.

Birkhoff, G. *Lattice Theory*, 3rd ed. Cambridge, MA: Harvard U., 1963.

Birkhoff, G. & T.C. Bartee. *Modern Applied Algebra.* NY: McGraw-Hill, 1970.

Birkhoff, G. & S. MacLane. *A Survey of Modern Algebra.* NY: McMillan, 1965.

Bittinger, M. *Logic, Proof and Sets.* Reading, MA: 1982.

Bjorke, O. Manufacturing system theory--what is that? *Annals of the CIRP*, v. 22(1), 1973: 157-158.

Boulding, K. General systems theory--the skeleton of science. *Mangement Science*, v. 2, 1956: 197-208.

Brillouin, L. *Science and Information Theory.* NY: Academic, 1960.

Buckley, W., ed. *Modern Systems Research for the Behavioral Scientist.* Chicago: Aldine, 1968.

Bunge, M. *Ontology I: The Furniture of the World.* Boston: Reidel, 1977.

Bunge, M. *Ontology II: A World of Systems.* Boston: Reidel, 1979.

Burks, A.W. *Essays on Cellular Automata.* Urbana, IL: U. Illinois, 1970.

Chow, G.C. *Econometric Analysis by Control Methods.* NY: Wiley, 1981.

Clancey, W.J. Transfer of rule-based expertise through a tutorial dialogue. Dept. of Computer Science, STAN-CS-79-769, Stanford University, Sept. 1979.

Clocksin, W.F. & C.S. Mellish. *Programming in Prolog.* NY: Springer-Verlag, 1981.

Cothier, P.H. Assessment of timeliness in command and control. Laboratory for Information and Decision Sciences, Report LIDS-TH-1391, M.I.T., Aug. 1984.

Courtois, P.J. *Decomposability: Queueing and Computer System Applications.* NY: Academic, 1977.

Davis, M. *Computability and Unsolvability.* NY: McGraw-Hill, 1958.

de Kleer, J. Causal and teleological reasoning in circuit recognition. Ph.D. thesis, M.I.T., Jan. 1979.

Deming, W.E. *Quality, Productivity, and Competitive Position.* Cambridge, MA: M.I.T. Center for Advanced Engineering Studies, 1982.

Dorf, R.C. *Modern Control Systems,* 2nd ed. Reading, MA: Addison-Wesley, 1974.

Fortini, Earlwood T. *Dimensioning for Interchangeable Manufacture.* New York: Industrial Press, 1967: 51.

Futo, I., F. Darvas & P. Szeredi. The application of Prolog to the development of QA and DBM systems. In [Gallaire and Minker 1978: 347-376].

Galbraith, J. *Designing Complex Organizations.* Reading, MA: Addison-Wesley, 1973

Gallager, R.E. *Information Theory and Reliable Communications.* NY: Wiley, 1968.

Gallaire, H. & J. Minker, eds. *Logic and Databases.* NY: Plenum, 1978.

Gallaire, H., J. Minker & J.M. Nicolas. An overview and introduction to logic and data bases. In [Gallaire and Minker 1978: 3-32].

Gellert, W., H. Kustner, M. Hellwich & H. Kastner. *The VNR Concise Encyclopedia of Mathematics.* NY: Van Nostrand Reinhold, 1975: 320-356.

Georgescu-Roegen, N. *The Entropy Law and the Economic Process.* Cambridge, MA: Harvard U., 1971.

Gheorghe, A.V. *Applied Systems Engineering.* NY: Wiley, 1981: 19-27, 42-88.

Ginzberg, A. *Algebraic Theory of Automata.* NY: Academic, 1968.

Glueck, W.F. *Business Policy and Strategic Management,* 3 ed. NY: McGraw-Hill, 1980.

Groover, M. P. *Automation, Production Systems, and Computer-Aided Manufacturing.* Englewood Cliffs, NJ: Prentice Hall, 1980.

Guiasu, S. *Information Theory with Applications.* NY: McGraw-Hill, 1977.

Haken, H. *Synergetics.* NY: Springer-Verlag, 1978.

Hamilton, A.G. *Logic for Mathematicians.* Cambridge, UK: Cambridge U. Press, 1978.

Hardt, D.E. Shape control in metal bending processes: the model measurement tradeoff. In D.E. Hardt, ed., *Information Control Problems in Manufacturing Technology,* Proc. Fourth IFAC Symposium, 1982.

Hax, A. C. & N.S. Majluf. The corporate strategic planning process. Sloan School of Management, W.P. 1396-83, M.I.T., 1983.

Hopcroft, J.E. & J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley, 1979.

Horstmann, P.W. Expert systems and logic programming for CAD. *VLSI Design,* Nov. 1983: 37-46.

Hsu, J.C. & A.U. Meyer. *Modern Control Principles and Applications.* NY: McGraw-Hill, 1968.

Kaneshige, K. Quantitative analysis of axiomatic engineering design. S.M. thesis, M.I.T., May 1981.

Katz, D. & Kahn, R.L. *The Social Psychology of Organizations.* NY: Wiley, 1966: 14-29.

Kim, B.H. Low thermal inertia injection molding. Ph.D. thesis, M.I.T., July 1983.

Kim, S.H. Manufacturing axiomatics with special application to air compressor design. S.M. thesis, M.I.T., Sept. 1978.

Kim, S.H. & N.P. Suh. Application of symbolic logic to manufacturing axiomatics. *Robotics and Computer Integrated Manufacturing,* forthcoming, 1985.

Kneebone, G.T. *Mathematical Logic and the Foundations of Mathematics.* NY: van Nostrand, 1963.

Kowalski, R. *Logic for Problem Solving*. NY: North Holland, 1979.

Kowalski, R. Predicate logic as a programming language. DCL Memo 70, Edinburgh U., Edinburgh, UK, 1974.

Kuhn, T.S. *The Structure of Scientific Revolutions*. Chicago: U. Chicago, 1970.

Kuo, B.C. *Digital Control Systems*. NY: Holt, Rinehart and Winston, 1980.

Kutcher, M. Automating it all. *Spectrum*, v. 20(5), May 1983: 40-43.

Lawson, J.S., Jr. The role of time in a command control system. In M. Athans et al., *Proc. Fourth MIT/ONR Workshop on Distributed Information and Decision Systems Motivated by Command-Control-Communications Problems*, v. 4, Laboratory for Information and Decision Systems, Report LIDS-R-1159, M.I.T., Oct. 1981: 19-60.

Lewis, H.R. & C.H. Papadimitriou. *Elements of the Theory of Computation*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

Longo, G., ed. *Information Theory: New Trends and Open Problems*. NY: Springer-Verlag, 1975: iii-iv.

Luce, R.D. The theory of selective information and some of its behavioral implications. In R.D. Luce, ed., *Developments in Mathematical Psychology*, Glencoe, IL: Free Press, 1960: 5-119.

Luenberger, D.G. *Optimization by Vector Space Methods*. NY: Wiley, 1969.

M'Pherson, P.K. System theoretic concepts of systems engineering. *Advances in Cybernetics and Systems Research*. Proc. European Meeting, Vienna, 1973.

McCree, J. Vented compression molding: a new method for molding particulate materials. S.M. thesis, M.I.T., Jan. 1983.

McEliece, R.J. *The Theory of Information and Coding*. Reading, MA: Addison-Wesley, 1977.

Machlup, F. & U. Mansfield, eds. *The Study of Information: Interdisciplinary Messages*. NY: Wiley, 1983.

Magaziner, I. & R.B. Reich. *Minding America's Business*. NY: Harcourt, Brace, Jovanovich, 1982.

Manheim, M.L. *Hierarchical Structure: A Model of Design and Planning Process*. Cambridge, MA: M.I.T. Press, 1966.

Mattesich, R. Axiomatic representation of the systems framework: similarities and differences between Mario Bunge's world of systems and my own systems methodology. *Progress in Cybernetics and Systems Research*, v. IX, 1982: 51-75.

Mesarovic, M.D. & Y. Takahara. *General Systems Theory: Mathematical Foundations.* NY: Academic, 1975.

Mesarovic, M.D., D. Macko & Y. Takahara. *Theory of Hierarchical, Multilevel Systems.* NY: Academic, 1970.

Mizoguchi, F. Prolog-based expert system. *New Generation Computing*, v. 1, 1983: 99-104.

Nakazawa, H. & N.P. Suh. Process planning based on information concept. *Robotics and Computer Integrated Manufacturing*, v. 1(1), 1984: 115-123.

O'Toole, J. *Making America Work: Productivity and Responsibility.* NY: Continuum, 1981.

Oliveira, E. Developing expert systems builders in logic programming. *New Generation Computing*, v. 2(2), 1984: 187-194.

Organization for Industrial Research, Inc. Group technology CAD/CAM. Waltham, Mass. Undated.

Padulo, L. & M.A. Arbib. *Systems Theory: A Unified State Space Approach to Continuous and Discrete Systems.* Philadelphia: Saunders, 1974.

Pereira, F., ed. C-Prolog user's manual, version 1.5. Dept. of Architecture, U. of Edinburgh, U.K., Feb. 1984.

Porter, M.E. *Competitive Strategy.* NY: Free Press, 1980.

Prigogine, I. *From Being to Becoming: Time and Complexity in the Physical Sciences.* San Francisco: Freeman, 1980.

Prior, A.N. *Time and Modality.* Oxford: Clarendon Press, 1957.

Quastler, H., ed. *Information Theory in Psychology: Problems and Methods.* Glencoe, IL: Free Press, 1955: 9-10.

Rescher, N. & A. Urquhart. *Temporal Logic.* NY: Springer-Verlag, 1971.

Rinderle, J.R. Measures of functional coupling in design. Ph.D. thesis, M.I.T., June 1982.

Rinderle, J.R. & N.P. Suh. Measures of functional coupling in design. ASME Paper 82-PROD-27, 1982.

Roberts, F.S. *Measurement Theory, with Applications to Decisionmaking, Utility, and the Social Sciences.* Reading, MA: Addison-Wesley, 1979.

Robinson, J.A. Machine oriented logic based on the resolution principle. *J. ACM*, v. 12, 1965: 23-44.

Sain, M.K. *Introduction to Algebraic Systems Theory.* NY: Academic, 1981.

Shannon, C.E. & W. Weaver. *The Mathematical Theory of Communication.* Urbana, IL: U. Illinois, 1949.

Shortliffe, E.H. *Computer Based Medical Consultations: MYCIN.* New York: American Elsevier, 1976.

Simon, H.A. *Administrative Behavior,* 2nd ed. NY: McMillan, 1961.

Simon, H.A. On the application of sevomechanism theory to the study of production control. *Econometrica,* v. 20(2), April 1952.

Simon, H.A. *The Sciences of the Artificial.* Cambridge, MA: M.I.T. Press, 1969.

Simon, H.A. and A. Ando. Aggregation of variables in dynamic systems. In [Ando et al. 1963: 64-91].

Strang, G. *Linear Algebra and its Applications,* 2nd ed. NY: Academic, 1980.

Suh, N.P. The future of the factory. *Robotics and Computer Integrated Manufacturing,* v. 1(1), 1984: 39-49.

Suh, N.P., A.C. Bell & D.C. Gossard. On an axiomatic approach to manufacturing and manufacturing systems. *J. of Engineering for Industry,* Trans. ASME, v. 100(2), May 1978a: 127-130.

Suh, N.P., S.H. Kim, A.C. Bell, D.R. Wilson, N.H. Cook, N. Lapidot, & B. von Turkovich. Optimization of manufacturing systems through axiomatics. *Annals of the CIRP,* v. 27(1), 1978b: 383-388.

Suh, N.P. & J.R. Rinderle. Qualitative and quantitative use of design and manufacturing axioms. *Annals of the CIRP,* v. 31(1), 1982: 333-8.

Szolovitz, P. Artificial intelligence and clinical problem solving. Lab. for Computer Science, M.I.T., LCS-TM-140, Sept. 1979.

Tang, A.M., F.M. Westfield & J.S. Worley, eds. *Evolution, Welfare, and Time in Economics.* Lexington, MA: Lexington Books, 1976.

Tice, W.W. The application of axiomatic design rules to an engine lathe case study. S.M. thesis, M.I.T., Feb. 1980.

Trimberger, S. Reaching for the million-transistor chip. *Spectrum,* v. 20(11), Nov. 1983: 100-102.

Tsypkin, Y.Z. *Adaptation and Learning in Automatic Systems.,* trans. by Z.J. Nikolic. NY: Academic, 1971.

van Melle, W., A.C. Scott, J.S. Bennet and M.A. Peairs. The EMYCIN manual. Heuristic Programming Project, HPP-81-16, Stanford University, Nov. 1981.

von Bertalanffy, L. *General Systems Theory.* NY: Braziller, 1968.

von Bertalanffy, L. The theory of open systems in physics and biology. *Science*, v. 111, 1950: 23-29.

von Neumann, J. *Theory of Self-Reproducing Automata*, ed. by A.W. Burks. Urbana, IL: U. Illinois, 1966.

Walker, A. Data bases, expert systems, and Prolog. Report RJ3870 (44067), IBM Research Lab., San Jose, CA, 22 April 1983.

Warfield, J.N. *Structuring Complex Systems.* Battelle Memorial Institute, Monograph No. 4, Columbus, Ohio, Apr. 1974.

Weaver, W. Science and complexity. *American Scientist*, v. 36(4), Oct. 1948: 536-544.

Whyte, L.L., A.G. Wilson, and D. Wilson. *Hierarchical Structures.* NY: American Elsevier, 1969.

Wiener, N. *Cybernetics, or Control and Communication in the Animal and the Machine*, 2nd ed. NY: Wiley, 1961.

Willsky, A.S. Dynamical systems defined on groups: structural properties and estimation. Ph.D. thesis, Dept. of Aeronautics and Astronautics, M.I.T., April 1973.

Wilson, D.R. An exploratory study of complexity in axiomatic design. Ph.D. thesis, M.I.T., Feb. 1981.

Yasuhara, M. Axiomatic engineering design: its concept and procedure. S.M. thesis, M.I.T., Feb. 1980.

Yoshikawa, H. Multipurpose modelling of mechanical systems -- morphological model as a mesomodel. In O. Bjorke and O.I. Franksen, eds., *System Structures in Engineering: Economic Design and Production*, Trondheim, Norway: Tapir, 1978: 594-629.

Yu, F.T.S. *Optics and Information Theory.* NY: Wiley, 1976.

Appendix A

Roots of Industrial Productivity and Implications for Research

It is often useful to differentiate between the state of art and the state of practice. For example, economists speak of the production possibility frontier (PPF) which defines the combinations of maximum output obtainable from a given economy. The PPF is a representation of economic capability, not of actuality: an economy or organization capable of producing at one level under optimum conditions may actually be organized to produce less by sociopolitical or other constraints.

Critics such as O'Toole [1981] have suggested that the major cause of American productivity decline is a degradation in social values and an inflation of worker expectations. This implies that the economy is producing at a point inside the PPF due to motivational constraints. While certain cultures such as those of Japan and Germany seem to engender a high level of motivation, it is doubtful that their culture can be directly transplanted and nurtured in the American milieu. This, however, is an uncertain conclusion as witnessed by the success of industrial plants operated on American soil by Japanese subsidiaries.

Given that psychology and sociology are still inexact sciences at best, it appears that the most assured way to develop a rational foundation for productivity improvement is to focus on technical issues and to expand the PPF. In this way, even if the percentage of waste due to loose social values remains constant, the total output will increase.

The long-run benefit due to expansion of the PPF is the increase in the output per worker through automation. Moreover, automation will release legions of workers from the routine, unsatisfying jobs which lie at the root of the motivational problem, and therefore ultimately the source of the economic malaise. Through the application of computer and robot technology, the problem of worker motivation may be circumvented entirely.

Of course there will be some human costs associated with displacements due to the employment of novel technologies. Despite the vociferous claims of the the self-proclaimed "humanists" who argue against the evils of technological unemployment, however, the displacement costs--and rewards--

seem to be purely at the individual level and/or over the short term. Batteries of economic studies have failed to confirm the hypothesis that new technology spells mass unemployment for the overall economy. In fact, these studies suggest the opposite: new technologies raise productivity and thereby ensure prosperity in the large and in the long run.

In the U.S., for example, the healthiest local economics are those which have embraced the most advanced techologies, such as the silicon areas of California and Massachusetts. The European experience, as illustrated by the early mistakes of Great Britain and France as well as their subsequent efforts at rectification, shows a similar pattern.

In the swift currents of the day, only those who manage to negotiate the technological waves will adapt properly to the new environment. Any group--whether at the level of the individual, the organization or the nation--must exploit the new technologies if it is to maintain a competitive edge in this global economy.

Focus of Research

The issues relating to industrial performance or productivity may be explored from two perspectives: empirical and theoretical. Research to date has concentrated on the empirical approach, the gathering and analysis of data. Less work has been performed from a theoretical perspective.

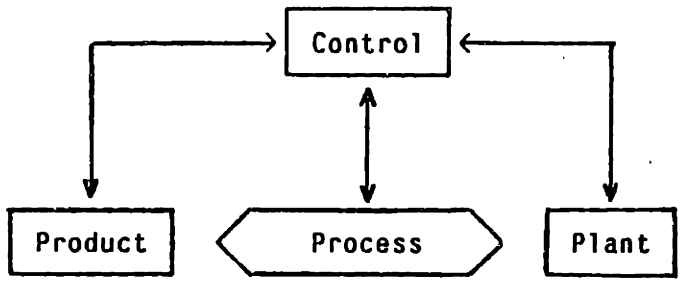
Both the empirical and theoretical approach can be taken at different levels of aggregation, whether in terms of the product, plant, firm, industry, or economy. Productivity, for example, can be measured at any of these levels of aggregation. Productivity at the level of the economy is measured in terms of output value over the number of labor hours, or other similar measures. Productivity at the level of the firm is formulated in terms of return on assets, sales, or equity. Industrial productivity is usually studied using the metrics at the level of the firm or the economy.

Productivity at the plant or product levels tend to rely on monetary metrics. However, the relative simplicity of the situation suggests that a more objective measure such as yield rate could be used.

A Framework for Manufacturing Systems

A manufacturing system may be partitioned into components consisting of the product, plant, process and control. The *product* is the component that ultimately develops into the desired output of the manufacturing system, as specified by a set of functional requirements. The *plant* is the immediate environment which operates on inputs such as energy and materials to produce the

desired product. The plant transforms the input items into the output product through the manufacturing *process*.



This interaction process is mediated by a *control* subsystem which acquires basic information and guides the process by using meta-level information such as those in feedback loops. The basic information pertains to the state of the product (e.g. engine block 003 is at Workstation 9) and plant (e.g. Workstation 10 is inoperative), while meta-level information (e.g. route Job 7 to Workstation 15) mediates the way in which the plant and product interact (e.g. drill four holes at specified locations).

Shortcomings of Previous Programs in Productivity Improvement

One study of productivity improvement projects in manufacturing industries identified 3 classes of programs [Barocci et al. 1982]:

1. Enhancing employee attitudes through quality of work life programs and quality circles: suggestions and bilateral communications between workers and supervisors, based on worker initiative.

2. Increasing product quality through quality circles and statistical quality control.

3. Decreasing cost by reducing overhead and streamlining operations.

Currently, however, programs such as these suffer from several drawbacks:

1. Fuzzy definition of the original problems, program objectives, and success criteria often result in the failure of the project.

2. Success, when it is achieved, is usually marginal. The objectives of one cost reduction

program, for example, was to cut expenses by 3% to 6% per year.

3. No theory exists to guide the formulation or implementation of the productivity improvement programs.

Research on human factors (e.g. worker attitudes) for the purpose of enhancing industrial productivity faces the following limitations:

1. Employee attitudes are based on personal and environmental factors. To the extent that personal factors in attitude formation are based on emotions or whims, no rational theory is at hand, almost by definition. To study nonpersonal sources of productivity variations, it is sufficient to ensure that external factors which affect motivation are controlled or adjusted for. For example, it might be ensured and assumed that the issue of providing clean, safe workplaces has been judiciously addressed by considering the benefits and costs.

2. Increasing levels of automation will continue to reduce the numbers of workers in the industrial sector, thereby reducing the impact of direct labor on production cost. The popular analogy is the reduction of farm workers over the past century due to increases in agricultural productivity.

3. Projects to enhance employee attitudes affect productivity in only a marginal way [Barocci et al. 1982].

4. Even if the entire direct labor cost were to be eliminated, the impact on the final selling price of products would only be about 5 percent (see next section).

If industrial productivity is to be enhanced significantly, our focus of attention must shift from the primary workforce to the secondary level of the engineer and the tertiary level of the manager [Deming 1982; Magaziner and Reich 1982].

Components of Product Cost

Manufacturing cost is the largest component of the selling price in batch-produced goods as shown in the table below [Kutcher 1983]. For our purposes, we may take as given the 20% of the total selling price reserved for corporate profits. The 25% due to administrative expenses and the 4.8% due to

Engineering cost	15%	
Profit	20	
Marketing, Selling, Administrative	25	
Manufacturing	40	
Depreciation & Energy		4.8%
Direct labor		4.8
Indirect labor		10.4
Parts & Material		20.0
Total sales price	100%	

indirect manufacturing labor may well be susceptible to reduction through effective management information systems. We leave these areas, however, to other research efforts.

This still leaves us almost 45% to work with. A procedure that aids the design engineer, whether in product development or manufacturing, will have an impact on the 15% due to engineering cost. In addition a methodology that results in improved products and processes will affect the 29.6% due to material, capital and direct labor requirements.

Of the cost of manufacture, 50% is due to materials, 26% to indirect labor and 12% each to capital and direct labor requirements. These statistics highlight the inadequacy of many productivity improvement programs, such as those directed toward incremental increases in labor output: even if the entire labor content were eliminated, the change in manufacturing cost would be nominal.

For the same reason, automation and flexible manufacturing systems will not by themselves resolve the productivity dilemma. In fact, the use of automation equipment will reduce direct labor costs but increase capital expenditures.

Within the realm of manufacturing costs, the area offering the greatest margin for improvement is in materials. The reduction of material cost may be achieved by introducing smaller and lighter products, eliminating production waste, and substituting cheaper materials. All these tactics depend on the design of products and processes. In fact, proper design can produce quantum reductions in capital and labor requirements, and is therefore the key to enhancing productivity [Suh 1984]. In such industries as electronics and polymer processing, for example, effective process design may improve product yield by an order of magnitude or more.

However design work to date has been characterized by trial and error moderated by personal experience, without the benefit of a theoretical framework for making decisions. The vital role of design in enhancing industrial productivity highlights the need for a scientific approach to the design

of products, processes and software systems. One such methodology is the axiomatic approach. This methodology is consistent with the premise that advancing technology, both in its physical and organizational embodiments, is the main driving force behind industrial productivity.

In summary, the wellsprings of industrial productivity lie in the application of new, efficient technologies to the production function. Hence increased efficiency in the production workforce can only result in incremental gains at best. If our objective is to enhance productivity by several factors or even orders of magnitude, this can be realized only by dramatic technical and organizational improvements. Current research efforts in manufacturing science, including the work reported in this thesis, is directed toward fortifying the technical aspect by establishing a rational scientific base for the manufacturing field.

Appendix B

Decision Making in Design as a Selection among Alternatives

The axiomatic approach to manufacturing has been proposed as a prescriptive paradigm for decision making in design and manufacturing [Suh et al. 1978a]. As a prescriptive model, it differs in form and purpose from descriptive models such as the satisficing [Simon 1961] and behavioral [Ackoff 1957-58] models.

The first part of this appendix deals with the specification of the decision making problem in design as a selection among alternatives, and addresses some general characteristics of the design situation. The latter sections discuss the similarities and differences between the axiomatic approach and the satisficing and behavioral models.

B.1 Genesis of the Design Problem

The general design problem begins with the recognition of a need and its transformation into a set of independent objectives or functional requirements (FR's). The ensuing task is to generate some designs which satisfy the FR's.

The nature of the design obviously depends on the problem domain. For example, a design problem in public affairs may be to develop a schedule of monetary and fiscal policies that will stimulate the economy. A problem in engineering may involve the design of a sedan which will travel over 100 miles per gallon.

Any design is subject to a set of constraints, whether explicit or otherwise. An upper limit on the rate of inflation or a lower limit on mean time to failure are two examples.

The set of functional requirements and constraints completely define the decision problem by specifying the set of acceptable solutions. In addition, by defining an evaluation criterion, it is possible to identify the optimal set of solutions. The evaluation criterion or performance index is a

measure of "goodness" which permits the decision maker to compare the merits of alternative designs.

The optimal set may consist of zero or more multiple elements. The optimal set will be empty when the set of FR's and constraints admits no feasible solutions. It will consist of two or more designs if their performance indices are equal.

B.2 Characteristics of Feasible Solutions

Let F be the set of all possible designs which fulfill the functional requirements and constraints. Let $|F|$ denote the cardinality or number of elements of F. Then $|F|$ may be 0, finite, or infinite. In the following discussion we assume that $|F| > 0$. A *parameter* is a variable which refers to some aspect or attribute of a design. A *continuous* parameter is one which assumes values on a set of continuous intervals rather than discrete points.

Let f_i and f_j be two elements in F. Suppose that f_i and f_j can be described in the same way, in the sense that they can be defined by the same number and type of continuous parameters. Then we may say that f_i and f_j are *parameter-equivalent*, or simply *p-equal*. In the opposite case, we may call the two feasible designs parameter-inequivalent or p-unequal.

> **Example.** Let f_1 be the design for a rotary engine, while f_2 and f_3 correspond to reciprocating engines. Moreover, f_2 and f_3 are identical except for the lengths of the piston cylinders (thereby resulting in different compression ratios). Since the parameters for f_2 and f_3 are of the same type and number, the two designs are p-equal. On the other hand, f_1 is p-unequal to both f_2 and f_3.
>
> **Example.** Let f_1 and f_2 be the designs for two aircraft which are identical in all key aspects except for the size and number of engines. Since the number of engines is a discrete rather than continuous variable, we say that f_1 and f_2 are p-unequal.

A *category* is a set of designs $f_i \in F$ which are p-equal. As with many complex situations [Simon 1969], the set of categories may exhibit a hierarchical structure. This is illustrated in the examples at the end of this section.

Let C^* be a category of F. Then in general there may be infinitely many elements in C^*.

A set S is *convex* if, given two elements a and b in S, the convex combination of a and b is also in S. In other words, given a and b in S, the element

$$c = u * a + (1 - u) * b$$

is in S for all $u \in (0,1)$. When C^* is a convex set consisting of two or more distinct elements, then $|C^*|$ is uncountably infinite.

Example. Suppose that a given car may be fitted with tires of two different radii, R and R', as well as any size in between. Then the cardinality of the category containing this parameter is uncountably infinite.

Although each category may be uncountably infinite, we may show that the number of categories is at most countably infinite.

Theorem. Let $C_1, C_2, C_3, \ldots$ be distinct categories of F. Then the set of C_i's is at most countably infinite.

Proof. All the elements $f_i(j)$ in category C_i have the same number and type of parameters. If another design f_k is p-equal to $f_i(j)$, then it will be included in C_i; otherwise it will belong to another category C_n. Since the number and type of parameters can be enumerat[illegible], there are at most a countably infinite number of combinations of parameters. Each such combination represents exactly one category; hence the number of categories is at most countably infinite.

We can readily see that the categories comprise a collection of mutually exclusive, collectively exhaustive subsets of F.

Theorem. The categories $C_1, C_2, \ldots$ of F define a partition of F.

Proof. Let f_i be a design in F. Since f_i is p-equal to itself, it must belong to some category C^*. Hence the categories cover F.

Now suppose that f_i belongs to two categories, C^* and C^{**}. By definition the elements

in C^* and C^{**} must differ in at least one parameter, whether in type or number. But the type and number of parameters describing f_i are fixed, thereby contradicting the assumption that f_i is in both C^* and C^{**}. Hence the categories are mutually exhaustive.

We see that the set of feasible solutions F is composed of a finite or countably infinite number of categories, each of which may be uncountably infinite. Suppose that F has m categories, for $m \leq \infty$. Let p_i denote the number of parameters in category C_i. Then the total number of parameters may be

$$N = \Pi_{i=1,\ldots,m} p_i$$

Hence F is defined in N-dimensional space, or R^N.

We illustrate these ideas by an example.

Example. A key decision problem in corporate strategy involves the question of growth [Glueck 1980; Hax and Majluf 1983; Porter 1980]. For each division or strategic business unit, corporate mangement must decide whether to (1) Expand, (2) Maintain, (3) Harvest, or (4) Divest the operation.

For each of these alternatives, management must further decide what combination of (1) Administrative, (2) Financial, (3) Marketing, (4) Manufacturing, and (5) Technological resources to deploy. Each category of resource deployment will in turn have numerous parameters. For example, the design of the marketing strategy must resolve the budget to be allocated to different channels such as direct mail, TV advertising, billboards, radio spots, direct sales, magazines, or skywriting. (Elimination of any or all of these channels would correspond to a budget allocation of zero.)

This highly simplified decision problem has 4 * 5 = 20 categories. If we assume 5 parameters for each category, then the solution space F is defined on R^{100}.

Example. Suppose that a particular automobile panel can be fabricated from 4 materials: aluminum, fiberglass, polymer composite, or steel alloy. Moreover, each

material may be worked by 2 different production process such as extrusion or molding.

Hence the combinations of materials and processes give rise to 8 categories. Suppose that there are 5 continuous parameters in each category, such as dimensions, surface roughness, tolerance, or speed of extrusion. Then the solution space is defined on R^{40}.

In general, a design problem is a formidable decision making situation for which exhaustive search is neither practical nor feasible.

B.3 Statement of the Design Problem

It would appear that the decision making process occurs in two stages:

1. The first stage deals with optimization within each category C_i. Let x^i be the vector of parameters for all the designs in C_i, and X^i the space of feasible values of x^i in C_i. Then the task is to determine the optimum value x^* which gives the best design $f_i^* ::= f_i(x^*)$ according to some performance criterion P which is defined on the f_i's.

2. The second stages involves a comparison of the $f_1^*,...,f_m^*$ and the selection of the best overall design f^*.

At each stage the decision maker (DM) extremizes some performance criterion, whether in terms of maximizing benefits or minimizing costs. An example in manufacturing may be the maximization of production subject to capacity constraints. In designing a program of theoretical research, however, the objective may be to maximize expected benefits subject to a budget constraint.

Let M be the set of indices corresponding to the categories of *F*. In other words $M = \{1,2,...,m\}$ where m may be finite or infinite. We let *ext* denote the extremizing operation, whether it be maximization or minimization. Then the decision problem may be stated as

$$\text{ext}_{i \in M} \{ \text{ext}_{x^i \in X^i} \quad P(f_i(x_i)) \}$$

Since we require the x^i to be in X^i, the designs are implicitly restricted to be in the feasible set *F*.

The procedural analogue of this problem statement is:

```
Procedure Extremize

Input FR's and C's.
Generate the set F of all feasible designs.
For each category C_i
  Determine the values x* of the design
    parameters such that f_i is optimal
    according to some performance index P.
  Call this design f_i*.

Compare the f_i*.  The best design f*
  is the solution.
End.
```

No limits have been assumed on the ability of the DM to follow this procedure. This would imply, for example, that every feasible design in F can be generated. Moreover the procedure may be assumed to execute in zero time, since no deadlines or constraints on processing capacity have been mentioned.

An ultra-rational model such as this is often used by economists to simplify problems in microeconomic analyses. A more realistic paradigm, however, is the satificing model of decision making.

Satisficing Model

In reality decision making is subject to various limitations such as

* Distorted perception of the problem.

* Failure to generate any or enough alternative designs.

* Bounds on information processing capability.

* Time constraints.

The cost of the decision maker's time limits the number of hours that may be allocated to a particular problem. In addition there are constraints on the absolute time scale due to changing environmental conditions. For example, a computer company that enters the 32-bit personal computer market one year behind the industry leaders may never be able to acquire enough market share to attain adequate scale economies.

Under limitations such as these, the appropriate decision strategy is to satisfice rather than extremize. This model seems to be applicable to fields as diverse as engineering [Manheim 1966], administration [Simon 1961], and medicine [Szolovits 1979].

In contrast to the extremizing model, the satisficing paradigm cannot be stated so tersely in a single statement. The satisficing model is formalized below as an iterative procedure involving sequential search in the feasible space.

```
        Procedure Satisfice

        Input FR's and constraints.

  New:  Generate a new set G of candidate designs which might
          be feasible.
        Perform a rough analysis on each design f_i in G.

        Case 1: {There is at least one f_i in G which
                 looks promising.}
                Perform a detailed analysis on G.

                Case 1a: {There is no feasible design in G.}
                  Go to Reset.
                Case 1b: {There is at least one feasible design.}
                  Compare performance indices among feasible
                    designs.
                  If the highest index P* exceeds or matches
                    the minimum permissible P*, then
                    the corresponding design is the solution;
                  Else goto Reset.
        Case 2: {There is no design in G which looks promising.}
                Go to Reset.

  Reset:
        If time or other resources are at a critically low
          level, then expand the feasible space by relaxing
          or eliminating some FR's or constraints.
        Go to New.

        End.
```

Manheim [1966] illustrates this procedure by applying it to the design of a highway route. The stepwise procedure for generating alternative selections is as follows:

1. Determine a number of zones which appear promising as highway routes.

2. In each zone, select one or more locations for closer inspection.

3. Develop designs for particular locations.

At each level, the alternatives are selected by (1) Assigning costs to each design activity, and (2) Estimating the construction cost for a highway, based on the available information. This cost is the expected cost of the route that might ultimately be developed if the procedure is taken to completion.

The utility of a particular design is defined by the probability distribution of costs that would result if each plan is followed to completion. The selection criterion is minimal cost based on a decision-theoretic Bayesian analysis.

This procedure specifies (1) The region in which to continue the search, and (2) The stopping rule or satisficing criterion. At each stage decisions must often be made on the basis of incomplete information about the present and expected scenarios concerning an uncertain future. Under such conditions the evaluation subprocedure may be supported by the techniques of statistical decision theory, such as sequential sampling and optimal stopping rules.

The Satisfice procedure is vulnerable to interrupts for various reasons. Chief among these is an exhaustion of resources. When the decision maker has exhausted his temporal or financial resources, for example, then the search procedure must be suspended. If more resources can be acquired, the procedure may resume at the interrupted stage; otherwise it must be aborted. Other interrupts may be traced to changes in the environment or in the perception of the problem situation. The development of a new-generation emission catalyst, for example, may be terminated by the replacement of gasoline engines by electric motors in the automotive market.

Axiomatic Model

The axiomatic decision procedure relates to the way in which the optimal design f^* in F may be attained. Axiom 1 states that the best way to obtain f^* is to identify the set of designs f_i which satisfy the FR's independently. Hence the Independence Axiom presents a criterion for effectiveness, and offers a shortcut to identifying f^* during the design process.

Axiom 2 asserts that, other things being equal, the design with minimal information is superior. The Information Axiom reflects a measure of efficiency relating to the optimal design.

The reduction of information may be examined at different stages in the lifecycle of the design alternative, whether during the specification, fabrication, or operation phase.

1. Design. As the complexity and uncertainty of a decision making task increase, so do the information requirements [Galbraith 1973]. For a given decision problem, however, minimizing the information requirements improves the decision making procedure. Axiom 1 on functional independence is appropriate in this context, since coupled FR's result in interdependencies which may otherwise be ignored when the functions are independently satisfied.

2. Production. Minimizing information requirements such as allowing for lax tolerances increases the performance of manufacturing systems [Nakazawa and Suh 1984].

3. Operation. Other things being equal, a simpler design will exhibit lower operating costs in terms of reliability, availability, and serviceability. The MIT reaction injection molding machine, for example, was designed to increase system availability over previous designs. The ability to control pressure and flow independently allows for the operator to specify these variables immediately rather than groping for the desired combination through trial and error. We may note that this is also an example of maintaining functional independence.

The axiomatic paradigm can be formalized as follows. Let idm(f) be some measure of independence or lack of coupling among the FR's of a design f. Two metrics which have been proposed for this purpose are reangularity and semangularity [Suh and Rinderle 1982]. Let ifm(f) be the information measure of f. Then the axiomatic procedure may be summarized as:

```
min    ifm(f)

such that  (1)  f ∈ {max idm(g)}
           (2)  g ∈ F
```

In other words, the procedure is to select the design with minimal information among the candidates which exhibit maximal functional independence.

This is an idealized model since no limitations are explicitly recognized, such as the ability to generate all designs which exhibit functional independence. A more practical version of the axiomatic model is given below.

```
Procedure Axiomatic
```

```
        Input FR's and constraints.

New:    Generate a new set G of candidate designs which might
          be feasible.
        Perform a rough analysis on each design f_i in G.

        Case 1: {There is at least one f_i in G which
                 looks promising.}
                Perform a detailed analysis on G.

                Case 1a: {There is no feasible design in G.}
                  Go to Reset.
                Case 1b: {There is at least one feasible design.}
                  Compare the feasible designs according to
                    the idm metric.
                  If there is at least one design whose idm value
                    is ≥ the minimum permissible idm*,
                  Then
                    Call this set G';
                    The design in G' with the lowest information
                      requirement is the solution;
                  Else goto Reset.
        Case 2: {There is no design in G which looks promising.}
                Go to Reset.

Reset:
        If time or other resources are at a critically low
          level, then expand the feasible space by relaxing
          or eliminating some FR's or constraints.
        Go to New.

        End.
```

We may note that procedures Axiomatic and Satisfice share some common characteristics. Chief among these may be the iterative nature of specifying the FR's and constraints.

B.4 Relationship of Axiomatics to a Behavioral Model

The behavioral model of communication and decision making deals with the way in which a decision making situation is affected by new communications or items of information [Ackoff 1957-58]. This section first describes the salient aspects of the behavioral model, then discusses it in the context of the axiomatic approach to decision making in design.

Description of the Behavioral Model

A purposeful state S is defined in terms of the following attributes:

1. N, the environment or setting which provides the context for the decision making problem.

2. DM, the decision maker or entity to which purposefulness is attributed.

3. C_i, a course of action; $1 \leq i \leq m$.

4. O_j, a possible outcome or consequence of a course of action; $1 \leq j \leq n$.

5. p_i, the probability that DM will select C_i in a particular environment N. $p_i = p(C_i|I,N)$.

6. E_{ij}, the probability that O_j will transpire if C_i is selected by DM in N. E_{ij} is called the efficiency of C_i for O_j in N, and is given by $p(O_j|C_i,I,N)$.

7. V_j, the value or importance of O_j to DM.

A choice or course of action C_i is said to be *available* in an environment if there is some positive probability p_i that it will be selected by someone. An available choice may have zero probability of being selected by a particular individual under a specific set of circumstances. Then it is not a *potential* course of action for him.

Efficiency is a probability relation between the sets of choices and of outcomes. The outcomes may be given as discrete events such as meeting emission standards, or continuous parameters such as miles per gallon.

Value may be measured in terms of a relative metric regardless of its initial parametrization. This may be achieved by a linear mapping from its original set of values to the interval [0,1], then normalizing to 1. For example, let v_j be the value of outcome j defined to take on a number between 0 and 1. Then the relative value may be defined as $V_j = v_j/(\Sigma_k v_k)$.

Using these concepts, we may now define a purposeful state S in the following way. A decision maker DM is said to be in a purposeful state S in an environment N if the following conditions hold:

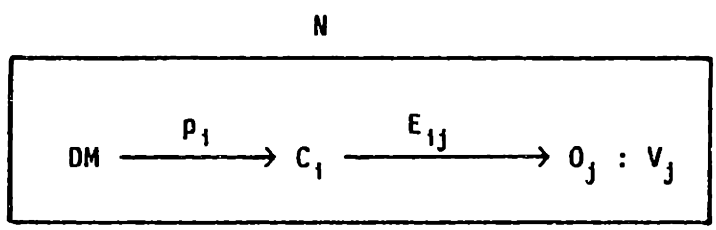
1. There are two or more exclusive courses of action $C_1,\ldots,C_m$ available in N, for $m \geq 2$.

2. Among the courses of action available in N, at least two are potential choices for DM.

3. Among the set of outcomes--defined to be exclusive and exhaustive--there is one (say O_a) for which two of the courses of action (say C_1 and C_2) have some positive but unequal efficiency. In other words, E_{1a} and E_{2a} are both > 0, but not equal to each other.

4. The outcome for which condition 3 holds has some value to DM. That is, $V_a > 0$.

In short, an individual is in a purposeful state if he values something which can be obtained by unequally efficient means. The components of the behavioral model are summarized in the following diagram:



A *state* is defined by the set of parameters DM, p_i, C_i, E_{ij}, O_j and V_j. For a given set of available choices C_i and possible outcomes O_j, the value of a state, V(S), is a function of the probabilities, efficiencies, and values:

$$V(S) = f(p_i, E_{ij}, V_j)$$

One functional form for the value of a state is the expected return:

$$V(S) = \Sigma_{i,j}\ p_i * E_{ij} * V_j \qquad \text{Eq. 1}$$

Suppose that the V_j are defined to lie on the interval [0,1] and to sum to 1. Since p_i and E_{ij} also lie on [0,1], V(S) ranges from a low of 0 to a high of 1.

When a communication is sent or received, the individual's purposeful state changes from the initial state S_1 to the final state S_2. The change in state is due to a change in any or all of the p_i's, E_{ij}'s or V_j's. The value of the communication to the receiver, for example, is

$$\Delta V = V(S_2) - V(S_1) \qquad \text{Eq. 2}$$

where Δ denotes the difference operator. The value of ΔV may be positive, zero or negative.

A communication which changes the probabilities of choice may be said to *inform*; one that affects the efficiences, *instructs*; and one that changes the values of outcomes *motivates*. A given communication may do any or all of these concurrently.

Using Equations 1 and 2, we may write the value of a communication in the form

$$\Delta V = \Sigma_{i,j} \ (p_i + \Delta p_i)*(E_{ij} + \Delta E_{ij})*(V_j + \Delta V_j) \qquad \text{Eq. 3}$$

where i ranges from 1 to m, and j from 1 to n.

By expanding the product, we may recognize the following first order terms:

$$\Delta V_a = \Sigma_{i,j} \ \Delta p_i*E_{ij}*V_j$$

$$\Delta V_b = \Sigma_{i,j} \ p_i*\Delta E_{ij}*V_j$$

$$\Delta V_c = \Sigma_{i,j} \ p_i*E_{ij}*\Delta V_j$$

These terms represent the value of information, instruction, and motivation, respectively.

The higher order terms ΔV_{ab}, ΔV_{ac}, ΔV_{bc}, and ΔV_{abc} may be defined similarly. For example,

$$\Delta V_{ab} = \Sigma_{i,j} \ \Delta p_i*\Delta E_{ij}*V_j$$

gives the value of the joint effect of information and instruction (not the sum of their independent effects).

With this notation, Eq. 3 may be written as

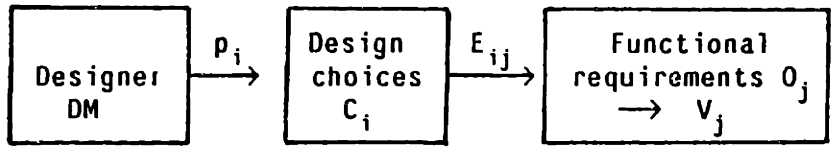
$$\Delta V = \Delta V_a + \Delta V_b + \Delta V_c + \Delta V_{ab} + \Delta V_{ac} + \Delta V_{bc} + \Delta V_{abc}$$

thereby isolating the contributions to change in value as the sum due to independent and interrelated effects of information, instruction and movtivation.

Relationship to Axiomatics

In axiomatics, we may identify the designer as DM, the candidate set of designs as C_i, and the functional requirements or operating behaviors as O_j. The relative importance or priorities of the functional requirements determine the V_j. The p_i define the probability of selecting specific designs, while the E_{ij} refer to the likelihood with which a particular design will yield the desired behaviors.[1]

[1]The decision maker initially selects design C_i in view of the functional requirements. Each C_i then has probability E_{ij} of satisfying functional requirement O_j.



The set of constraints determine which designs C_i are feasible, by assigning $p_i = 0$ if the ith choice is unacceptable. Hence a constraint is a communication that informs.

The Axioms may be considered to be messages which redefine the parameters relating to the design activity. Axiom 1 on functional independence is a statement about how certain design choices C_i affect the likelihood of yielding the desired operating characteristics O_j. Hence the first Axiom affects the E_{ij}, and is therefore a communication that instructs.

Axiom 2, in contrast, calls for a minimization of information as long as Axiom 1 is satisfied. Hence it affects the selection probabilities p_i and is therefore a communication that informs.

The propositions listed below also affect the probabilities of selecting particular designs, and are therefore communications that inform:

* Integrate functional requirements (FR's) in a single component if the functions can be satisfied separately.

* Everything else being equal, conserve materials.

* Minimize the number and complexity of part surfaces.

* If a solution satisfies more FR's and constraints than those originally specified, the part or process may be overdesigned.

* Use standard or interchangeable parts whenever possible.

* Part count is not a measure of productivity.

* Order the constraints by priority, and minimize their number.

The following propositions pertain to the likelihood of attaining the FR's depending on the choice of

design. Since they affect the efficiences E_{ij}, they are communications that instruct.

* A part should be a continuum if energy conduction is important.

* If weaknesses cannot be avoided, separate parts.

In a given iteration of the design process, the FR's are defined to be the set of independent specifications, all of which must be satisfied. Hence each FR has the same weight:

$$V_j = V(FR_j) = 1/n$$

where n is the number of FR's. When a given set of FR's cannot be satisfied subject to the constraints, they must be ordered by priority and culled. This procedure has the effect of reducing at least one V_j so that it becomes $< 1/n$, while at least one other V_k becomes $> 1/n$. Hence the following propositions pertain to state transitions between two iterations of the design process. By affecting the V_j's, they are communications that motivate:

* Order the FR's according to priority.

* Minimize the number of FR's.

* Satisfy the FR's in order of importance.

An example of a meta-level communication which transcends any particular component of the behavioral model is the following:

* There may be several optimal solutions.

B.5 Summary

Decision making in design may be viewed as a selection among candidates in a set of feasible solutions. The design problem is formalized as a multistage problem consisting of the optimization of continuous parameters or variables within a set of discrete categories. The general decision problem, as well as the satisficing and axiomatic procedures, have been discussed.

In addition the axiomatic approach to design has been presented in the context of a behavioral model of communication and decision making. The axiomatic principles were shown to correspond to different components of the decision making model. In reference to the behavioral model, the

Axioms and their derivatives may be classified as *informing, instructing, motivating,* or *meta-level* communications. In particular, Axiom 1 on functional independence is seen to be a communication that instructs, while Axiom 2 on information minimization is one that informs.

Appendix C

Background on Measures of Functional Independence

Functional coupling refers to the degree of interdependence among design parameters and their relationships to the functional requirements [Kim and Suh 1985; Rinderle 1982; Rinderle and Suh 1982; Suh and Rinderle 1982]. Let $FR = (FR_1, \ldots, FR_m)^T$ denote the vector of functional requirements and $DP = (DP_1, \ldots, DP_n)^T$ the set of design parameters. All the FR's and DP's are related to each other by the coupling matrix C, whose elements are defined by the relation

$$C_{ij} = \frac{\partial(FR_i)}{\partial(DP_j)}$$

where ∂/∂ denotes the partial derivative operator.

In matrix form, the relationship is

$$FR = C * DP$$

Here * denotes the multiplication of a matrix with a vector. If we partition C into its column vectors $\langle C_1, \ldots, C_n \rangle$ then the relation becomes

$$FR = C_1 * DP_1 + C_2 * DP_2 + \ldots C_m * DP_m$$

where * denotes vector multiplication.

C.1 Reangularity

Suppose that two design parameters DP_j and DP_k result in the same set of relative changes in the FR vector. This means that the column vectors C_j and C_k are proportional in algebraic terms. Since we can change FR by varying either DP_j or DP_k, one of them can be discarded. (In the language of linear algebra, we are seeking the minimum spanning basis for the vector space defined by FR.)

When DP_j and DP_k are collinear, it is easy to see that the design is poor. The case where DP_j and DP_k are orthogonal is clearly the best situation for obtaining specific values for FR. What happens in the general case when DP_j and DP_k are neither collinear nor orthogonal, but form some arbitrary

angle t between 0 and 90 degrees?

To take care of all situations, we may define a measure of orthogonality among the design parameters with the formula

$$R = \Pi_{i,j} \quad |sin \langle C_i, C_j \rangle| \qquad \text{Eq. 1}$$

where i ranges from 1 to n-1 and j from i+1 to n. That is, R is given by the product of the absolute values of sines of angles between all pairs of the column vectors. The R measure is called *reangularity* (for *reg-angle-arity*, or *right-angle-quality*).

It is clear that R takes on values on the closed interval from 0 to 1. R = 0 when one or more pairs of the design vectors C_i are parallel, and increases gradually to a maximum value of R = 1 when all the vectors are mutually orthogonal.

For ease of computation, we may write R in terms of the coupling matrix elements C_{ij}. First we note that

$$|sin \langle C_i, C_j \rangle| = [(sin \langle C_i, C_j \rangle)^2]^{1/2} = [1 - (cos \langle C_i, C_j \rangle)^2]^{1/2}$$

Now

$$(cos \langle C_i, C_j \rangle)^2 = \frac{(C_i * C_j)^2}{[\|C_i\| * \|C_j\|]^2}$$

where ||*|| denotes the Euclidean norm. Hence we may write Eq. 1 as

$$R = \Pi_{i,j} \left\{ 1 - \frac{(\Sigma_k \; C_{ki} * C_{kj})^2}{(\Sigma_k \; C_{ki}^2) * (\Sigma_k \; C_{kj}^2)} \right\}^{1/2}$$

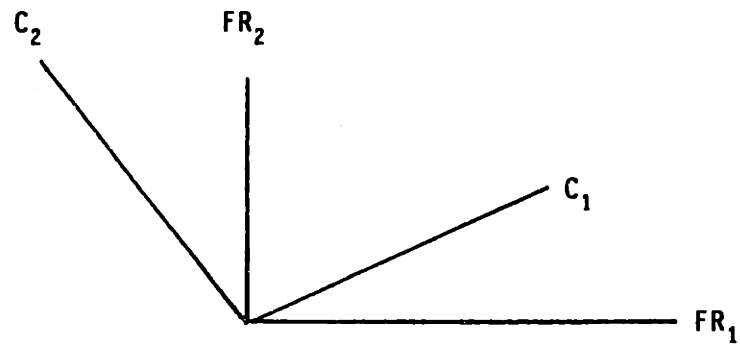
where i = 1,...,n-1 and j = i+1,...,n.

Example 1. Consider the design example

$$FR = \begin{vmatrix} 2 & -1 \\ 1 & 2 \end{vmatrix} * DP$$

For this 2-dimensional case, only one pair of column vectors need be treated in calculating R:

$$R = \left\{ 1 - \frac{(2 * (-1) + 1 * 2)^2}{(2^2 + 1^2) * ((-1)^2 + 2^2)} \right\}^{1/2} = 1$$



Since the vectors C_1 and C_2 are orthogonal as shown in the diagram below, the reangularity takes its maximal value of 1.

C.2 Semangularity

Reangularity is an important measure of functional independence but by itself is inadequate. It takes into account the orthogonality among the design parameters but says nothing about their "alignment" to the functional requirements. An example will clarify this point.

Example 2. Example 1 in the preceding section yielded the calculation R = 1. In the design situation defined by

$$C = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}$$

we obtain R = 1 also. According to the reangularity criterion, both cases are equally desirable.

In the first example, however, a designer must in general tweak both DP_1 and DP_2 and take into account their interactions before he can obtain the desired values of FR_1 and FR_2. In the second example, the adjustments are straightforward. In algebraic terms, the design problem is simplified when each diagonal term is large compared to other terms in the respective columns.

In geometric terms, the second case is superior to the first because the column vectors C_1 and C_2 are collinear respectively with the functional requirement axes FR_1 and FR_2.

Semangularity (for *sem-angle-arity* or *same-angle-quality*) is a measure which takes into account this

notion of "alignment". It is defined as

$$S = \Pi_{j=1,\ldots,n} \frac{|C_{jj}|}{\|C_j\|} = \Pi_{j=1,\ldots,n} \frac{|C_{jj}|}{[\Sigma_k (C_{kj}^2)]^{1/2}}$$

where $|C_{jj}|$ denotes the absolute value of the jth diagonal element, and $\|C_j\|$ denotes the Euclidean norm of the jth column vector. Note that the $\|C_j\|$ are normalizing factors that render the S metric immune to scale changes in the DP_j's.

The S metric takes on values from a low of 0 to a high of 1. $S = 0$ if one or more of the diagonal elements are 0 (meaning that the jth design parameter has no effect on the jth functional requirement). $S = 1$ if C is a diagonal matrix in which all diagonal elements are nonzero and all off-diagonal elements are zero.

Example 3. The semangularity for Example 1 from the previous section is given by

$$S = \frac{2}{[2^2 + 1^2]^{1/2}} * \frac{2}{[(-1)^2 + 2^2]^{1/2}} = 4/5.$$

On the other hand, the semangularity for Example 2 is

$$S = \frac{1}{[1^2 + 0]^{1/2}} * \frac{1}{[0 + 1^2]^{1/2}} = 1.$$

Since S is greater for the second example than for the first (while $R = 1$ for both), we conclude that the second design case is superior.

* * *

NOTE. Early work in functional coupling refers to metrics called *skew* and *alignment.* Skew was essentially the same quantity as reangularity. The difficulties with the old definition of alignment as a measure of coupling included its non-intuitive behavior and the fact that it could be interpreted only in conjunction with reangularity. Hence alignment was abandoned in favor of reangularity and semangularity.

Appendix D

Background on Measures of Information

The pervasiveness and importance of information may be paralleled only by energy. It has been suggested that the great inventions are those that transform, transmit and store energy--such as fire, wheels, windmills, engines and rockets--or perform similar operations on information--such as speech, writing, printing, photography, television and computers [Aczel and Daroczy 1975: 1].

The equivalence of different manifestations of energy such as heat, electricity and chemical energy was recognized around the middle of the nineteenth century. This development led to the measurement of energy in a common measure. A similar attempt was made a century later, concerning information: a measure of information was developed to capture the essense or structure of information, without regard to its meaning [Brillouin 1960; Quastler 1955; Shannon and Weaver 1949].

In everyday language, we use the term *information* in two different ways [Elstob 1980; Shannon and Weaver 1949]:

1. *Semantic information*, or *s-info*. This pertains to the meaning of a message or form.

2. *Physical information*, or *p-info*. This relates to reduction in uncertainty.

P-info deals with the amount of structure, order or pattern in an object or system, and with the retention of form as it is transformed, transmitted and stored. S-info, in contrast, is a subjective quantity which depends on the observer. These two types of information are related in at least one sense; any item of s-info must be conveyed on some manifestation of p-info. Information theory, which we also call *classical information theory* (CIT), deals strictly with p-info.

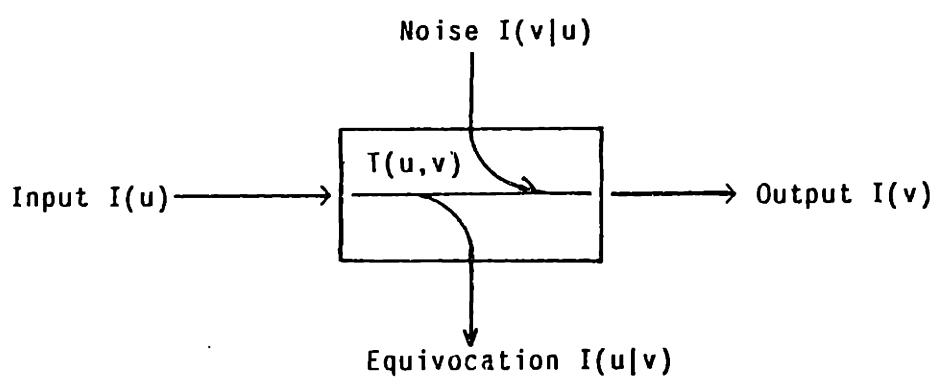
The first section of this appendix gives an informal introduction to CIT. This is followed by a more formal presentation plus a discussion of the information metric and its relationship to physical entropy. Since this formal construct often arises in design axiomatics, its use is illustrated through an application to tolerance specifications.

D.1 Informal Introduction

Information transmission occurs in physical systems which may be called communication systems. Such a system consists of 3 subsystems: a source, a channel, and a destination.[2] The *source* or *stimulus* produces a set of conditions which may be interpreted as *symbols*, *messages* or *events*. These messages travel over a *channel* and are received at the *destination* as an *output* or *response*.

In traversing the channel, a message may be subject to disturbances called *noise* or *ambiguity*. This quantity is a reflection of the uncertainty in the response when the stimulus is known. An example of a noise source is the signal distortion due to random thermal fluctuations in an copper wire. The *equivocation*, on the other hand, is that part of the stimulus information which is lost in transit: in other words, the uncertainty of the stimulus when the response is known. The *transmitted* information is the quantity that is conveyed by the channel when any existing correlation between stimulus and response is accounted for.

The relationships among these quantities are depicted in the diagram below.



The mathematical definition of information is based on the reduction of uncertainty. The simplest situation incorporating uncertainty is a choice between two alternatives. This uncertainty is a maximum when the two alternatives are equally likely. When a choice is made between two such alternatives, we say that 1 *bit* (for *b*inary dig*it*) of information has been conveyed.

Suppose that a choice is made among m equally likely alternatives, where $m = 2^n$. The $n = lb(m)$ bits of information have been transmitted, where *lb* denotes the binary logarithm, or log to base 2. Occasionally the natural log is used instead of the binary log; in that case the units of the information

[2] Sometimes two additional components, a *transmitter* and a *receiver*, are included to respectively encode and decode the messages into a form compatible with the channel. For example, if the source and destination are two persons, while the channel is a telephone line, then the transmitter is a telephone mouthpiece and the receiver is an earpiece.

measure are called *nits* or *nats*.

More generally, suppose that a particular event u has probability $p = p(u)$ of occurring. Then the information value of u, called the *surprisal* of the event, is given by $h = lb(1/p)$. As before, the units of h are bits.

Given a sequence of events $u_1,...,u_m$, the *information measure* of the sequence is given by the average of the surprisals:

$$I = \Sigma_{i=1,...,m}\ p_i h_i = -\ \Sigma_{i=1,...,m}\ p_i\ lb\ p_i$$

I is also called *average information*, *uncertainty*, or *entropy*.[3] If there are m events, all of which are equiprobable, then $p_i = 1/m$ for all i. In this case I takes its maximum value of

$$I_{max} = \Sigma_{i=1,...,m}\ p_i\ h_i = lb\ m$$

That this is in fact the maximum value for I, may be verified by the method of Lagrange multipliers under the constraint that the probabilities should add to 1, i.e. $\Sigma_i\ p_i = 1$.

D.2 Formal Presentation

This section presents a derivation of the classical information measure beginning from a number of constraints that such a measure should satisfy. Let $U = \{u_1,...,u_m\}$ be the set of input symbols or events being conveyed into the channel, and $V = \{v_1,...,v_m\}$ the output set. The prior probability of selecting the ith input (i.e. having the source in state s_i) is given by $p_i = p(u_i)$.

The assumptions on the information measure are specified next [Luce 1960: 23].

Postulate. The classical information measure satisfies the following properties:

1. Independence of alternatives. The amount of information transmitted by selecting u_i is a real number depending only on p_i, not on p_j for $j \neq i$. That is, the amount of information due to the selection of u_i is given by some function $f(p_i)$.

2. Continuity. $f(p_i)$ is a continuous function of p_i, since a small change in p_i should yield only a small change in the information transmitted.

---

[3] Entropy is the term used in statistical mechanics to denote the randomness of a physical system such as a volume of gas molecules. If m alternative states are possible, then the greatest possible value of entropy, denoted S, occurs when each state is equally likely. That is $p_i = 1/m$ for all i and $S \propto lb(1/p) = lb\ m$, where $\propto$ denotes proportionality.

3. Additivity. If two *independent* selections u_i and u_j with respective probabilities p_i and p_j are made, then the information due to the joint event $\langle u_i, u_j \rangle$ having probability $p_i p_j$ is the simple sum of the individual selections. That is,

$$f(p_i p_j) = f(p_i) + f(p_j)$$

4. Scale. A selection with probability 1/2 will convey 1 bit of information; that is, $f(1/2) = 1$.

If n is an integer, then repeated use of the third condition implies

$$f(p^n) = n\, f(p)$$

Letting $q^n = p$ in this equation yields

$$f(p^{1/n}) = f(q) = (1/n)\, f(q^n) = (1/n)\, f(p)$$

If k and n are integers, then the combined result is

$$f(p^{k/n}) = (k/n)\, f(p)$$

Let x be any number. We can select k and n to approximate x arbitrarily closely by the quotient k/n, so that the continuity of f implies

$$f(p^x) = x\, f(p)$$

We choose to let $x = -\,\mathrm{lb}\, p$ (where lb denotes the log to base 2), or equivalently $(1/2)^x = p$, obtaining

$$f(p) = f((1/2)^x) = x\, f(1/2) = -\,\mathrm{lb}\, p$$

This quantity is called the *surprisal.* Finally, the *average information of a message stream* or the *expected information transmitted by a source* with probability distribution $\{p_1,\ldots,p_m\}$ is[4]

$$I = -\,\Sigma_i\, p_i\, \mathrm{lb}\, p_i$$

The average information of a message stream going into, out of, between, and through the channel are given as follows.

Definition. Let $U = \{u_1,\ldots,u_m\}$ be the set of input messages into a channel, and V =

---

[4]Despite the formal equivalence of information and statistical entropy in many situations, they appear to be fundamentally different quantities. Section D.4 discusses their relationship further.

$\{v_1,\ldots,v_m\}$ be the set of output messages. Moreover p_i is the probability of occurrence of u_i and p_j the occurrence of v_j, while p_{ij} is the joint probability of occurrence of u_i and v_j.

1. The *average input information* going into the channel is

$$I(u) = -\Sigma_{i=1,\ldots,m}\ p_i \text{ lb } p_i$$

2. The *average output information* coming out of the channel is

$$I(v) = -\Sigma_{j=1,\ldots,m}\ p_j \text{ lb } p_j$$

3. The *average joint information* of input and output, due to the correlation between stimulus and response, is

$$I(u,v) = -\Sigma_{i,j=1,\ldots,m}\ p_{ij} \text{ lb } p_{ij}$$

4. The *average transmitted information* from input to output is

$$T(u,v) = I(u) + I(v) - I(u,v)$$

This quantity reflects the amount of information conducted through the channel, when adjusted for the average amount of coupling which may be computed in advance.

In addition to these quantities, information may be distorted in transit through the channel. Spurious information impinging on the channel is called *noise*, while information lost between input and output is called *equivocation*. In the definition below, we use the notation $p_{j|i}$ to mean $p(x_j|x_i)$, i.e. the probability of occurrence of x_j given x_i, for some variables x_i and x_j. By the definition of conditional probability, $p_{j|i} = p_{ij}/p_i = p_{ij}/\Sigma_j\, p_{ij}$.

Definition.

1. The *conditional information* of the output given the input u_i is given by

$$I(v|u_i) = -\Sigma_j\, p_{j|i} \text{ lb } p_{j|i}$$

2. The *average* or *expected conditional information* of the output given the input is obtained by summing over information components due to the inputs:

$$I(v|u) = \Sigma_i\, I(v|u_i) = -\Sigma_{i,j}\ p_{ij} \text{ lb } p_{j|i}$$

This quantity, called *noise*, is also equal to

$$I(v|u) = I(v) - T(u,v)$$

which can be shown by taking logarithms from the inequality $p_{j|i} = p_{ij}/p_i$, and summing over i and j.

All indices i and j run from 1 to m unless otherwise specified.

Similar remarks apply to equivocation, resulting in the following definition.

Definition. Information lost from the channel, called *equivocation*, is given by

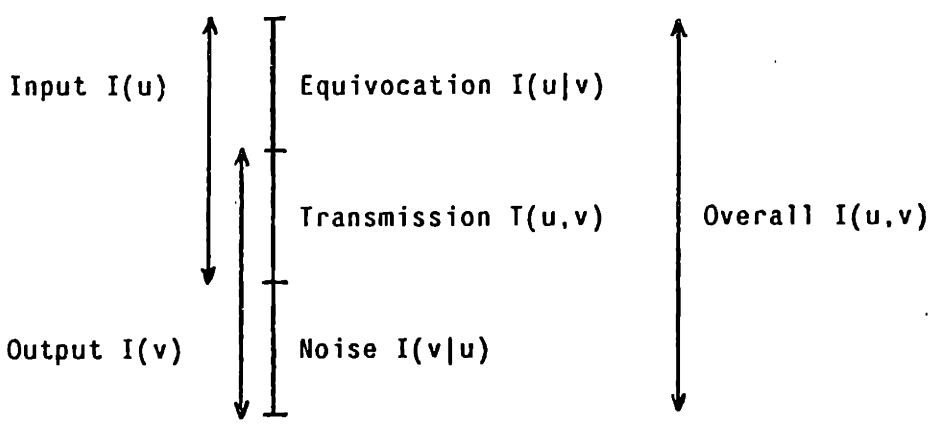
$$I(u|v) = -\Sigma_{i,j=1,\ldots,m}\ p_{ij}\ \mathrm{lb}\ p_{i|j}$$

This can be shown to equal

$$I(u|v) = I(u) - T(u,v)$$

In other words, equivocation is the average information still required to identify the input, after the output has been received. On the other hand, noise is the average information required to identify the output after the input has been specified.

The relationship among these quantities is represented pictorially below.



To illustrate, we consider a number of special cases.

1. *Perfect correlation between inputs and outputs.* Here the channel is perfect, having no noise and no loss. Then $p_{ij} = p_i = p_j$ for $i = j$, and $p_{ij} = 0$ for $i \neq j$. The consequence is that $I(u,v) = I(u) = I(v)$. This implies that the transmitted information $T(u,v)$ reaches its maximum value of $T(u,v) = I(u) = I(v)$. In other words, the input, output and

transmitted information are all equal for the perfect channel.

2. *Independence of input and output.* In the case of zero correlation between input and output, $p_{ij} = p_i p_j$ and $I(u,v) = I(u) + I(v)$. This imples that $I'(u,v) = 0$: no information is transmitted when the stimulus and response are uncorrelated.

3. *Constant response.* Here $I(v) = I(v|u) = I'(u,v) = 0$, and $I(u) = I(u|v) = I(u,v)$. As in Case 2 above, no information is transmitted.

Some derivative concepts apply to relative information and redundancy.

Definition. Let I' be the observed value of information for a source with messages $\{u_1,...,u_m\}$, while $I_{max} = lb(m)$ is the maximum possible value of information.

1. The *relative information* or *relative entropy* of the message source is given by

$$r = I' / I_{max}.$$

2. The *redundancy* of the message source is given by the complementary quantity

$$R = 1 - r.$$

Channel capacity is a measure of the information-transmitting capability of a channel over the long run.

Definition. Let $\{u_1,...,u_m\}$ be a set of messages, each having equal probability of occurrence, impinging on a channel. Moreover, let $N(\tau)$ be the number of messages conveyed through the channel over a time period τ. The the *channel capacity* is given by

$$C = \lim_{\tau \to \infty} [lb\ N(\tau]/\tau$$

That is, C is the information transmitted per unit time when averaged over a long period.

D.3 Discussion of the Classical Information Measure

The classical information measure, in spite of its generality and usefulness, has its limitations. The main limitation springs from its obliviousness to semantic information. In addition, although the classical measure is often used as an indicator of the degree of organization, it should be considered only a proxy rather than a direct measure [Bunge 1979: 272]:

1. Information is not defined on a set of systems, or on the states of a system, but rather on system-message pairs. That is, I: {Systems}x{Messages} -> $\mathbb{R}$. Hence information is not an intrinsic property of a system but is a mutual property of a system and a message.

2. A system is neither predictable nor unpredictable in itself. Whatever can be predicted is done so through theories and facts. Hence the degree of predictability is defined on the pairs {Systems}x{Theories} rather than on the set {Systems} alone.

3. Prediction is not tantamount to deterministic prediction. If a system is stochastic--as exemplified by radioactive decay in atoms or interactions in social groups--then its behavior can be predicted statistically. For example, transition matrices and standard deviations may be determined. Moreover, a stochastic system such as a probabilistic automaton is nondeterministic by construction.

The classical information model is too restrictive, and a more general theory of information should be sought [Longo 1975: iv]. This goal is addressed to some extent by the generalized information metric discussed in Chapter 4, as well as the behavioral model of decision making given in Appendix B. Both these theories take into consideration some measure of semantic information.

D.4 Information versus Entropy

Quantum mechanics asserts that a confined particle may only take on discrete values of energy, which we call eigenstates. The proportion of particles in the ith eigenstate may be denoted as f_i, for i = 1,...,m. If we know the distribution of particles in the respective eigenstates, then

$$I = - \Sigma_i \; f_i \, \mathrm{lb} \, f_i = - (1/\ln 2) \; \Sigma_i \; f_i \ln f_i$$

gives the information available about the system.

From statistical mechanics, we know that the entropy of the ensemble is given by

$$S = - k \; \Sigma_i \; f_i \ln f_i$$

where k is Boltzmann's constant. Hence in this case, entropy and information are directly proportional:

$$S / I = k \ln 2$$

Because the classical information metric has the same equational structure as entropy, some authors believe them to be identical. However, information and entropy have different physical interpretations and are not always related so simply as in the case above.

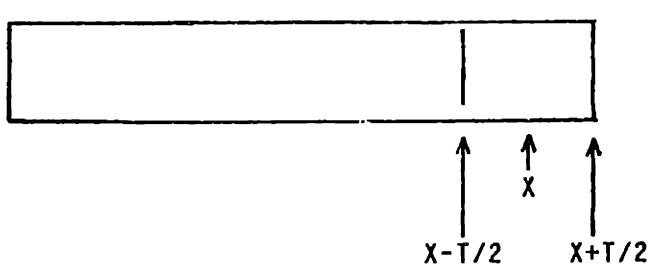
Entropy is a physical quantity inherent in the system and may therefore be called a first-order quantity. Information, however, is a second-order quantity whose value depends on the observer.

Brillouin [1960] has shown that obtaining information requires the expenditure of work or useful energy, which we know from thermodynamic considerations to imply an increase in entropy. Hence extracting information about the state of a system entails an increase in entropy even when the state of the system is otherwise unaffected. Moreover, entropy may increase without any yield of information; an example is found in the mixing of two gases in an opaque, sealed container, when the membrane separating them is broken. Hence information and entropy seem to be related but nonequivalent.

D.5 Tolerance Information

This section offers a brief note on tolerance information for referential purposes, since the concept arises frequently in discussions of design axiomatics. The interpretation of information given here is consistent with the Generalized Information Model of Chapter 4, which is based on the likelihood of satisfying functional requirements.

Consider the axial dimension in specifying a cylinder for fabrication.



Suppose that the desired length is X, with tolerance ± T/2. If we assume a uniform probability distribution on the interval [0,X+T/2], then the probability of obtaining the desired outcome is just

$$\frac{T}{X + T/2} \simeq \frac{T}{X}$$

The approximation T/X follows from the mild assumption that the tolerance is much smaller than the dimension, i.e. $T \ll X$. The associated information is given by the logarithm of the probability of success, or $I = lb(X/T)$.

Appendix E

Some Theorems on Axioms 1 and 2

This appendix builds on the discussions in Appendices C and D, and presents some theorems related to measures of functional independence and information. The first section proves a number of results relating to the modularity and invariance of functional independence measures. This is followed by a decision making procedure for using the independence measures when the number of design parameters may or may not equal that of the functional requirements.

The second section presents two theorems relating to information requirements. First, the information required to traverse an interval is shown to be least when the traversal is completed in one step rather than two or more. Second, a proof is given for the proposition that coupling increases information requirements. This proposition may represent the most important link between Axioms 1 and 2.

E.1 Theorems on Measures of Functional Independence

Modularity of Independence Measures

For a coupling matrix C which can be partitioned into square submatrices which are nonzero only on the main diagonal, we can derive simple formulas for the reangularity and semangularity of C in terms of those for the submatrices.

> Theorem (Modularity of Independence Measures). Suppose that a coupling matrix C can be partitioned into square submatrices which are nonzero only along the main diagonal. Then the reangularity and semangularity for C are equal to the products of their corresponding measures for each of the nonzero submatrices.

In other words, suppose that the N x N matrix C can be written in the form of a matrix as shown below. Here each C_i is a square submatrix of order N_i x N_i, and the 0's denote square submatrices composed of 0 values. Obviously $N_1 + N_2 + ... + N_K = N$.

$$C = \begin{vmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \cdot\cdot & \cdot\cdot & \cdots & 0 \\ 0 & 0 & 0 & C_K \end{vmatrix}$$

For such a matrix C, the reangularity R and the semangularity S are given by

$$R = R_1 * R_2 * \ldots * R_K$$

$$S = S_1 * S_2 * \ldots * S_K$$

where the R_i and S_i factors are the respective measures of reangularity and semangularity for submatrix C_i. The proof of the theorem follows.

Proof. Let c(i) denote the ith column vector of the C matrix, and c(i,j) the element of C corresponding to the ith row and j^{th} column. The reangularity R is defined as [Rinderle 1982; Rinderle and Suh 1982; Suh and Rinderle 1982]:

$$R = \Pi_{i,j}\ [1 - A(i,j)]^{1/2} \qquad \text{Eq. 1}$$

where the product is taken over the indices i = 1,2,...,N-1 and j = i+1,...,N. Each A(i,j) term denotes the square of the cosine of the angle formed by column vectors c(i) and c(j); that is,

$$A(i,j) = [c(i) * c(j) / \|c(i)\| * \|c(j)\|]^2$$

Here c(i) * c(j) denotes the inner product of the ith and j^{th} column vectors:

$$c(i) * c(j) = c(1,i)*c(1,j) + c(2,i)*c(2,j) + \ldots + c(N,i)*c(N,j)$$

while ||c(i)|| denotes the Euclidean norm of the ith column:

$$\|c(i)\| = [c(1,i)^2 + c(2,i)^2 + \ldots + c(N,i)^2]^{1/2}$$

The semangularity S is defined as

$$S = \Pi_i\ |c(i,i)|/\|c(i)\| \qquad \text{Eq. 2}$$

where the index i runs from 1 through N.

To calculate the reangularity for the coupling matrix C, we must calculate the inner

products c(i)*c(j) as the numerators of the A(i,j) factors. If c(i) and c(j) correspond to the same submatrix, say C_1, then the inner product is

$$c(1,i)*c(1,j) + \ldots + c(N_1,i)*c(N_1,j)$$

since the elements in rows N_1+1 and lower are all 0. But this is the same expression as the inner product c(i)*c(j) for calculating R_1 corresponding to submatrix C_1.

Now suppose that c(i) and c(j) correspond to different submatrices, say C_1 and C_2. But the 0 elements in c(i) and c(j) are positioned in such a way that each term of their inner product is zero. That is,

$$c(1,i)*c(1,j) = c(2,i)*c(2,j) = \ldots = c(N_1,i)*c(N_2,j) = 0.$$

Since c(i)*c(j) = 0, the corresponding angular measure is A(i,j) = 0.

From Eq. 1, we see that this factor (within the square brackets) equals 1, and therefore has no effect on the reangularity R. In other words, the inner products of columns vectors corresponding to different submatrices have no impact on R. Hence the reangularity R for the whole matrix C can be written as the product of reangularities for the nonzero submatrices:

$$R = R_1 * R_2 * \ldots * R_K$$

We can proceed analogously for the semangularity S. Consider the semangularity corresponding to any submatrix, say S_1 for C_1. S_1 is given by the product

$$S_1 = u(1) * u(2) * \ldots * u(N_1)$$

where each u(i) is defined as

$$u(i) = |c(i,i)| / [c(1,i)^2 + \ldots + c(N_1,i)^2]^{1/2}$$

Since the elements $c(N_1+1,i)$ and lower are all 0, the expression u(i) corresponding to the ith column of C_1 is equal to the u(i) for the ith column of the full matrix C.

A similar statement may be made for any column corresponding to any submatrix C_j.

Hence the semangularity S for the whole matric C equals the product of the semangularities for each of the submatrices:

$$S = S_1 * S_2 * \ldots * S_K$$

This completes the proof of the Modularity Theorem.

Effective Reangularity for a Scalar Coupling Element

The indices in Eq. 1 imply that at least 2 column vectors are required for the calculation of reangularity. The following corollary, however, shows that the effective reangularity may be defined in a simple way.

Corollary (Effective Reangularity for a Scalar). The effective reangularity R for a scalar coupling "matrix" or element is 1.

Proof. Consider the following coupling matrix C, where D is an M x M submatrix with M > 1, and d is a scalar "submatrix":

$$C = \left| \begin{array}{c|c} D & 0 \\ \hline 0 & d \end{array} \right|$$

From the preceding Modularity Theorem, the reangularity for C is given by $R = R_D * R_d$, where R_D and R_d denote the reangularity measures for D and d, respectively.

Now consider the angular measure $\Lambda(i,M+1)$ defined in connection with Eq. 1. Since the first M components of the last column are all 0, the value of $\Lambda(i,M+1)$ is 0. Hence only the $\Lambda(i,j)$ corresponding to the first M factors contribute to R. In other words, $R = R_D$. By relating the last two equations for R, we obtain $R_d = 1$. Hence the effective reangularity for a scalar coupling component is 1.

Invariance Under Reordering

The following theorem states that the functional requirements (FR's) and their associated design parameters (DP's) may be interchanged in the FR and DP vectors without changing the reangularity and semangularity of the coupling matrix. Note that the proof is for the general N x N coupling matrix; it is not restricted to, say, triangular matrices.

Theorem (Invariance). Reangularity and semangularity for a coupling matrix C are

invariant under alternative orderings of the FR and DP variables, as long as the orderings preserve the association of each FR with its corresponding DP.

To illustrate, take the simple 2 x 2 case given by

$$x = a * p + b * q$$
$$y = c * p + d * q$$

where the design parameter p is most closely associated with the functional requirement x, and the parameter q with requirement y. Then the coupling matrix is given by

$$C = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

Suppose we now interchange the order of the equations while still ensuring that the FR's and DP's correspond as before. Then:

$$y = d * q + c * p$$
$$x = b * q + a * p$$

The corresponding coupling matrix is

$$C' = \begin{vmatrix} d & c \\ b & a \end{vmatrix}$$

The invariance theorem says that the values of the reangularity and semangularity for C' are the same as those for C.

Proof. The proof will be given in 3 steps:

1. The reordering of the FR's and DP's corresponds to one row and one column interchange of the elements of the coupling matrix. Moreover, it makes no difference whether the rows are interchanged first, then the columns, or vice versa.

2. Reangularity R is unaffected by any reordering.

3. Semangularity S is unaffected by any reordering.

Step 1.

Suppose that a functional requirement FR* is the ith element of the FR vector, while

the corresponding primary design parameter DP* is the ith element of the DP vector. When we move FR* to the jth position in the FR vector, we must also move DP* to the jth position in the DP vector in order to maintain correspondence between these variables (otherwise the whole concept of semangularity, or correspondence between any FR_k and DP_k, is meaningless). But relocating FR* corresponds to an interchange of rows i and j in the N x N coupling matrix C, while relocating DP* corresponds to interchanging columns i and j.

From linear algebra, we recall that such interchange operations can be represented by pre- and post-multiplication by elementary matrices. Let E(i,j) denote the elementary matrix obtained from the N x N identity matrix by interchanging rows i and j. Then the product E(i,j) * C results in an interchange of rows i and j in C, while C * E(i,j) results in an interchange of columns i and j. The compound operation of interchanging both rows and columns is effected by E(i,j) * C * E(i,j) = C'.

By the associative property of matrix multiplication,

[E(i,j) * C] * E(i,j) = E(i,j) * [C * E(i,j)]

Hence it is immaterial whether the rows are interchanged first, then the columns, or vice versa.

Step 2.

Let c(i) and c(j) denote the ith and j^{th} column vectors of C. What are the effects of the row and column interchanges on these vectors?

* Effect 1. Interchanging the columns results in a swapping of the c(i) and c(j) vectors. Other columns c(k), for k unequal to i or j, are left unchanged.

* Effect 2. Interchanging the rows results in a reordering of the ith and j^{th} elements

in all N column vectors.

Let c(m) and c(n) denote arbitrary columns of the coupling matrix. From Eq. 1, the angular measure between any two columns is defined as

$$A(m,n) = [c(m) * c(n) / \|c(m)\| * \|c(n)\|]^2$$

The indices in Eq. 1 imply that m < n. Without loss of generality, then, we can assume that i < j in the following discussion.

Three cases may be identified, depending on the relationships between i, j, m, and n.

1. *Neither m nor n equal to i or j.* From Effects 1 and 2 above, the inner product as well as the Euclidean norms of c(m) and c(n) remain unchanged under the transformation of the coupling matrix from C to C'. Hence A(m,n)', the value of A(m,n) for the transformed coupling matrix C', remains the same as A(m,n) for the original matrix C.

2. *m = i and n = j.* Then the positions of the column vectors are reversed, but their inner product and norms remain the same. Hence A(m,n)' = A(i,j)' = A(j,i).

3. *Exactly one of m or n equals i or j.* Without loss of generality, suppose that m is equal to i, and n unequal to j. From Effects 1 and 2 above, the new i^{th} column contains the same elements as the old j^{th} column, and all the elements (c(k,i)',c(k,n)') between columns i and n are paired as before. Since A(i,n)' = A(j,n), every occurrence of A(i,n)' may be replaced by A(j,n), and A(j,n)' by A(i,n).

In short, there corresponds to every A(m,n)' for C' exactly one A(k,l) in C and vice versa. As a result, the value of the product in Eq. 1 remains the same. So we see that reangularity is invariant under alternative orderings of the (FR_k,DP_k) pairs.

Step 3.

From Eq. 2, we see that the semangularity S is defined as the product of factors of the

form $u(k) = |c(k,k)|/||c(k)||$. We may identify 2 cases.

1. *k unequal to i or to j.* From Effect 2 in *Step 2* above, we see that c(k,k) will remain as the kth element of c(k). Since the new c(k) contains the same set of elements as before, ||c(k)|| is unaffected. Hence u(k) remains the same.

2. *k = i or k = j.* Without loss of generality, we assume that k = i. From Effects 1 and 2 above, we see that the new c(k,k) [= c(i,i)] will equal the old c(j,j). Also, the new ||c(k)|| will equal the old ||c(j)|| since they contain the same set of elements. Therefore the new u(k) will equal the old u(j).

In summary, the only effect of the transformation from C to C' is that the values of u(i) and u(j) are swapped. Hence semangularity is invariant under alternative orderings of the (FR_k, DP_k) pairs.

This ends the proof of the Invariance Theorem.

Design Procedure for Functional Independence

The design methodology relating to the Functional Independence Axiom is given in the following procedure. We let m equal the number of functional requirements and n the number of design parameters. Then the design procedure is as follows.

1. Identify FR's.

   a. List candidate FR's.

   b. Minimize number of FR's.

2. Identify DP's.

   a. List candidate DP's.

   b. Minimize number of DP's.

3. Determine coupling matrices.

   a. Write FR's in terms of DP's.

   b. Generate alternative coupling matrices.

If m = n, then:

Calculate numerical values of all the coupling matrices, as well as R and S. Try various orderings of the DP variables.

If m < n, then: {Design is underspecified}

Select alternative sets of m DP's as active parameters, using the remaining DP's as passive parameters. Calculate coupling matrices for various values of the passive parameters. For each set of m DP's, try various orderings of the DP variables.

If m > n, then: {Design is overspecified}

Eliminate some FR's or add new DP's.

4. Check for termination. If R and S are satisfactory, then stop. Else go to Step 1 with a new set of candidate FR's and/or DP's.

In Case 3b, only the orderings of the DP's are changed, not those of the FR's. This corresponds to reordering the columns of the coupling matrix C. According to the Invariance Theorem, the values of R and S are unaffected by paired interchanges of FR and DP variables. For example, if FR_1 and DP_1 are changed to FR_3 and DP_3, respectively, then R and S remain the same.

E.2 Theorems on Information

In an uncoupled design, each functional requirement can be met independently by adjusting the corresponding design parameter. In this way the desired solution can be readily obtained.

In a coupled system, however, each design parameter DP_j affects two or more FR's. It is no longer possible to determine the optimal value of DP_j in one stroke. When DP_j is adjusted to satisfy a functional requirement FR_m, some other function FR_n will be affected also. This situation may be rectified by adjusting another DP_k; since the system is coupled, however, FR_m will now be thrown out of kilter. If a feasible solution satisfying the FR's exists, it will in general be obtained only after a process of iteration and refinement of the DP's.

What is the impact of this iterative procedure in terms of information requirements? We would expect the information requirements to increase, and in fact we can show this to be the case.

Geometric information is defined as the logarithm of the range of a parameter divided by its tolerance. Because the set of functional requirements are specified within certain tolerances, the proper design parameters have their own tolerance bands. Since the logarithm of any quantity exceeding 1 increases less slowly than the quantity itself, we can show that the information required to traverse a dimension is minimal when the distance is covered in a single step or iteration rather than two or more.

Suppose, for example, that a design parameter DP_j must assume a value X_j within a tolerance band of width T. Then the information required for this specification is least when the proper value X_j is attained in a single iteration or "jump". In a coupled design, however, the optimal values X are obtained only after trial and error rather than known a priori.

These ideas are stated formally in the two theorems which follow. But first we introduce some definitions relating to the segmentation of an interval.

Definition. Let $S = \{X_i \mid X_i < X_{i+1} \text{ for } i = 0,1,\ldots,n-1\}$ be a set of real numbers. Then

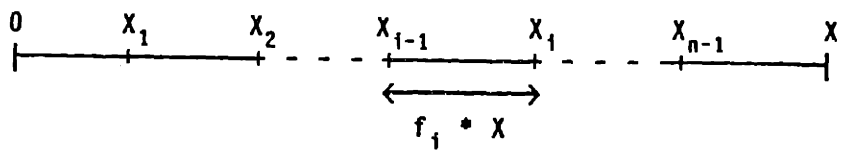
1. The set S is called a *segmentation* of the interval $[X_0,X_n]$.

2. An interval $[X_i,X_{i+1}]$ between two consecutive points in S is called a *segment.*

3. The largest possible interval $[X_0,X_n]$ on S is called the *entire interval,* or simply *interval.* Further, when $X_0 = 0$ and $X_n = X$, the interval $[0,X]$ will also be called simply the *interval* X when no ambiguity arises.

4. The fraction $f_i = [X_i - X_{i-1}]/X$ of interval X is called the ith *segment weight.*

The relationship among the segments and weights are depicted below.



We now define the notion of relative tolerances.

Definition. Consider a dimension or parameter whose range is X and tolerance is T. Then

1. The fraction $t = T/X$ is called the *relative tolerance of interval X.*

2. If X has segment weight f_i and tolerance T_i on the ith segment, then the ratio $t_i = T_i/[X_i - X_{i-1}]$ is called the ith relative tolerance of segment i.

In order to talk about realistic situations, we need to stick to cases where the tolerance on an interval is not too large compared to each segment.

Definition. A segmentation $S = \{X_0,...,X_n\}$ is said to be *reasonable* if the product of the segment weights exceeds the relative tolerance raised to the $(n - 1)^{th}$ power:

$$\Pi_{i=1,...,n} f_i > t^{n-1}$$

This definition will be used in the next theorem to filter out pathological partitions. An example of a pathological partition is one in which the length of each subinterval is as small as the tolerance.

Example. Suppose that a range X is segmented in such a way that the length of each subinterval equals a constant relative tolerance. Since $f_i = t$, we obtain

$$\Pi_{i=1,...,n} f_i = t^n < t^{n-1}$$

where the inequality follows from the fact that $t < 1$. According to the previous definition, the partition is not reasonable.

Now we introduce the first theorem which states that the information requirement for traversing an interval in multiple steps is more demanding than bridging the gap in a single step.

Theorem (Scale Economy). Let $S = \{X_0,...,X_n\}$ be a reasonable segmentation along a parameter whose interval is X and tolerance is T. The geometric information required to traverse X is minimum when S defines only one segment (i.e. $n = 1$).

Proof. Let $t = T/X$ be the relative tolerance, and $\{f_1,...,f_n\}$ be the segment weights of $\{X_0,...,X_n\}$. The information due to traversing the distance X on one jump (i.e. $n = 1$ and $S = \{X_0,X_1\}$) is given by the log of the range over the tolerance:

$$I_1 = lb\,(X/T) = lb\,(1/t)$$

where *lb* denotes the binary log, or logarithm to base 2.

To compare the information requirement for covering the interval X in n steps, we may consider two cases depending on assumptions about the tolerance defined on each subinterval.

Case 1. Proportional tolerance.

We assume that the tolerance T_i for the ith segment is proportional to the length of the subinterval; that is, $T_i = f_i * X$. The information needed for n partitions is given by

$$I_n = \Sigma_{i=1,\ldots,n} \text{lb}\,(X_i/T_i) = n * I_1$$

since $X_i/T_i = (f_i * X)/(f_i * T) = X/T = 1/t$. We see that information increases linearly with n, the number of partitions, when the tolerances are proportional.

Case 2. Constant tolerance.

We assume that the tolerance for each subinterval equals the overall tolerance; namely, $T_i = T$.

The information needed to cover X in n steps is

$$I_n = \Sigma_{i=1,\ldots,n} \text{lb}\,(X_i/T) = \Sigma_{i=1,\ldots,n} \text{lb}\,(f_i/t) = \text{lb}\,(\Pi_{i=1,\ldots,n} f_i/t^n)$$

Since $\{X_0,\ldots,X_n\}$ is assumed to be a reasonable partition, we know that $\Pi\, f_i > t^{n-1}$ and conclude that $I_n > I_1$.

This completes the proof of the theorem.

The next example is an illustration of Case 2 in the preceding theorem.

Example. Let X = 100 be the range of a dimension, and T = 1 the tolerance. If the segmentation of interval X is {0,50,100}, then the corresponding weights are $f_1 = f_2 = 0.5$. The relative tolerance of X is t = 1/100.

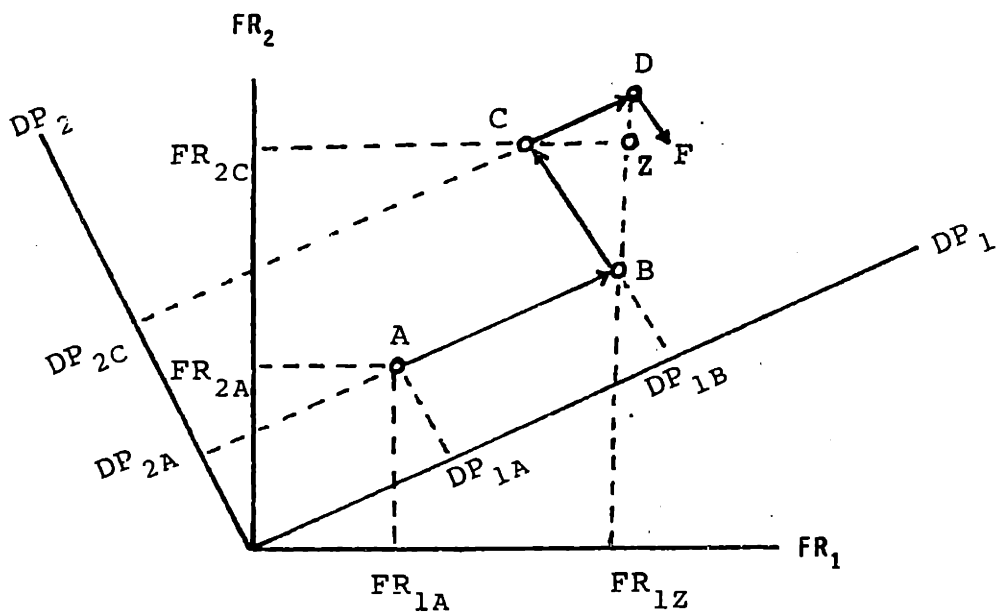
Since $f_1 * f_2 = .25 > t^{2-1} = .01$, the Scale Economy Theorem applies. In fact, the

information required for a single jump is I_1 = lb (1/.01) = 6.6 bits; the value for two jumps is I_2 = 2 * lb (.5/.01) = 11.3 bits. Hence traversing X in a single jump is more information-efficient than using two steps.

The next theorem relates coupling and information content.

Theorem (Coupling). Coupling increases information requirements.

Proof. We first consider the simplest coupling situation: the 2-dimensional case in which the FR's and DP's have a linear dependence and whose origins coincide. We assume, however, that the values of the coupling coefficients are unknown (otherwise we could simply solve for the DP's algebraically).



Let A and Z denote the starting and destination points, respectively. Suppose we move from A to Z by the following sequence of operations:

1. Increase DP_1 from DP_{1A} to DP_{1B}, until FR_1 reaches FR_{1Z}.

2. Increase DP_2 from DP_{2A} to DP_{2C}, until we reach FR_{2C}. However, FR_1 is now unsatisfactory.

3. Increase DP_1 again, thereby moving from point C to point D. But we have

overshot FR_2.

4. Decrease DP_2 to move from point D to point F. Now we have exceeded the FR_1 requirement.

5. And so on, until we finally converge to Z within the tolerances defined for FR_1 and FR_2.

The excess information required for this procedure may be traced to 3 factors:

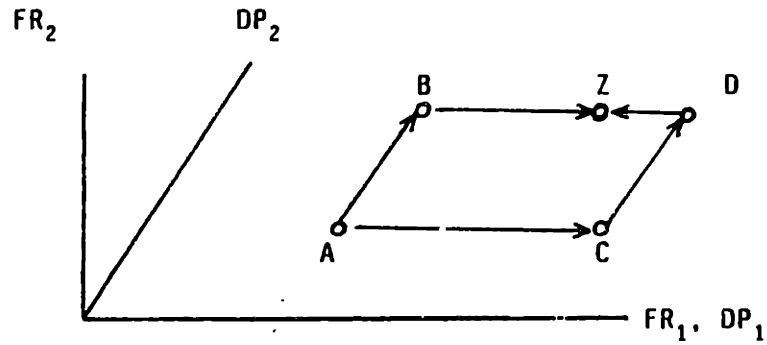
1. *Overshooting.* For example, the move from point C to point D in order to satisfy FR_1 results in overshooting the FR_2 requirement. This must be corrected by invoking DP_2 again.

2. *Retrograding.* The move from B to C in order to satisfy FR_2 has the side effect of disturbing FR_1, whose objective had been attained. Hence FR_1 must be adjusted again.

3. *Scale Economy.* Even when neither of the previous two conditions apply, the process of iterative adjustment increases information requirements, as indicated by the Scale Economy Theorem. Since the exact relationship between the FR's and DP's is assumed to be unknown, the likelihood of selecting the proper DP values in a single step is negligible. When the proper DP values are attained in two or more steps, the information requirements increase even when no overshoots or retrogrades occur.

For a quasi-coupled system, only a fortuitous sequence of DP adjustments will result in minimal information requirements.

On the diagram, adjusting DP_2 first, then DP_1, will result in a simple path from A to B to Z. However, adjusting DP_1 first will later require backtracking along the FR_1/DP_1 axis, to move from D to Z.



These situations correspond to the simplest case of two dimensions and linearity among the FR's and DP's. For higher dimensions and/or nonlinear relationships, the problems of backtracking, retrograding and scale economy cannot become any more tractable. Hence we may conclude that coupling increases information requirements.

In view of the Coupling Theorem, we might say that Axioms 1 and 2 are partially interdependent, being neither collinear nor orthogonal.

Appendix F

Background for Prolog

Prolog is based on a subset of first-order predicate calculus which allows only Horn clauses. It has a built-in theorem prover which uses a top-down, depth-first search strategy.

F.1 Propositions and Horn Clauses

We introduce some terminology which begins with primitive notions and leads up to the concept of a Horn clause.

Definition.

1. A *term* is one of the following:

   a. A constant symbol.

   b. A variable symbol.

   c. A function expression of the form $f(t_1,...,t_r)$, where f is an r-ary function symbol and $t_1,...,t_r$ are terms.

   No other objects are terms.

2. An *atom* or *atomic formula* is an expression of the form $P(t_1,...,t_n)$, where P is an n-ary predicate symbol and $t_1,...,t_n$ are terms.

3. A *literal* is an atomic formula, P, or its negation, ~P.

4. A *clause* is a set of literals $\{B_1,...,B_n,\sim A_1,...,\sim A_m\}$.

5. The subset of literals $\{B_1,...,B_n\}$ is the *conclusion* or consequent of the clause, while the subset $\{\sim A_1,...,\sim A_m\}$ is the *hypothesis* or antecedent.

6. The *empty* or null clause corresponding to $m = n = 0$ is denoted by $\square$.

7. A proposition in *clause form* is a set of clauses $C = \{C_1,\ldots,C_q\}$.

8. A *Horn clause* is a clause $C = \{B_1,\ldots,B_n, \sim A_1,\ldots,\sim A_m\}$ in which $n = 0$ or 1. A Horn clause may be further categorized as follows:

   a. $m = n = 0$. Then $C = \square$, denoting the *empty* clause.

   b. $m = 0, n = 1$. Then $C = \{B\}$, called an *assertion* or a *unit clause.*

   c. $m > 0, n = 0$. Then $C = \{\sim A_1,\ldots,\sim A_m\}$, called a *goal* or *goal sequence.*

   d. $m > 0, n = 1$. Then $C = \{B,\sim A_1,\ldots,\sim A_m\}$, called a *rule of inference.*

Prolog permits only Horn clauses.

F.2 Interpretation of Clause Form

Suppose that the clause $\{B_1,\ldots,B_n,\sim A_1,\ldots,\sim A_m\}$ contains variables $x_1,\ldots,x_k$. Then this clause is interpreted in Prolog as

$$(\forall x_1)\ldots(\forall x_k)(B_1 \vee \ldots \vee B_n \text{ if } A_1 \,\&\, \ldots \,\&\, A_m)$$

The empty clause, $\square$, corresponding to $m = n = 0$ is viewed as a contradiction.

The rationale for this interpretation is as follows. Without loss of generality we consider Horn clauses for which $n = 1$. In the expression

$$B \text{ if } A_1 \,\&\, \ldots \,\&\, A_m$$

the word *if* may be replaced by the left-arrow symbol, <-. The result is

$$B \leftarrow A_1 \,\&\, \ldots \,\&\, A_m$$

By the properties of functors and connectives, an expression of the form $B \leftarrow A$ is equivalent to $B \vee (\sim A)$. Consequently the previous expression may be written as

$$B \vee \sim(A_1 \,\&\, \ldots \,\&\, A_m)$$

By De Morgan's law for moving negation inwards, the result is

$$B \vee \sim A_1 \vee \ldots \vee \sim A_m$$

This disjunction of literals constitutes the set $\{B, \sim A_1, \ldots, \sim A_m\}$ which was the clause we started with.

A Horn clause of the form $\{B, \sim A_1, \ldots, \sim A_m\}$ is written in Prolog as

```
B :- A1,....,Am.
```

where the compound symbol, :-, stands for the left arrow, <-. A common convention is to use words beginning with capital letters to denote variables, and those starting with lower-case letters to stand for constants.

F.3 Procedure Interpretation

A simple way to view Prolog is as a procedural interpreter for Horn clauses. Procedures are specified as Horn clauses whose calls are controlled through pattern-matching by a theorem prover using the unification algorithm [Robinson 1965]. The arguments of the literals may be viewed as input and output parameters of procedure calls.

A clause in the form of a goal sequence,

```
:- A1,....,Am.
```

may be regarded as a series of procedure calls in which each literal A_i is in turn interpreted as a procedure call.

A rule of inference,

```
B :- A1,....,Am.
```

may be regarded as a *procedure declaration*, where B is the *head* of the procedure. The procedure calls represented by the A_i constitute the *body* of the procedure.

Assertions of the form

```
B.
```

may be viewed as statements of *fact* or as procedures with empty bodies. Finally the empty clause, □, may be regarded as the STOP instruction.

There are two classes of procedures in Prolog: (1) built-in procedures provided by the interpreter and (2) user-defined procedures specified by the programmer. Prolog, like LISP, has the capability to modify procedures. This built-in capability allows a program to change its set of clauses and thereby modify its own behavior specification in real time.