INFORMATION TO USERS

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

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

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

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

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

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106

75-10,184

HAYES-ROTH, Frederick Alan, 1947-
FUNDAMENTAL MECHANISMS OF INTELLIGENT BEHAVIOR: THE REPRESENTATION, ORGANIZATION, ACQUISITION AND USE OF STRUCTURED KNOWLEDGE IN PERCEPTION AND COGNITION.

The University of Michigan, Ph.D., 1974
Psychology, general

Xerox University Microfilms, Ann Arbor, Michigan 48106

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED.

FUNDAMENTAL MECHANISMS OF INTELLIGENT BEHAVIOR: THE REPRESENTATION, ORGANIZATION, ACQUISITION AND USE OF STRUCTURED KNOWLEDGE IN PERCEPTION AND COGNITION

by

Frederick Alan Hayes-Roth

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Psychology)
in The University of Michigan
1974

Doctoral Committee:

Professor Walter R. Reitman, Chairman
Professor John H. Holland
Professor David H. Krantz
Professor J. E. Keith Smith

To J. P. R.

ACKNOWLEDGMENTS

I would like to thank the members of my dissertation committee, John Holland, David Krantz, J. E. Keith Smith, and Walter Reitman for their willingness to participate in various aspects of this project, their constant encouragement, and generous accessibility. I am indebted to a number of people including Zenon Zannetos and Clyde Coombs for providing opportunities for me to pursue my research interests. I am intellectually indebted to those whose seminal ideas have been the "training information" for my own discoveries, notably D. O. Hebb, J. H. Holland, J. McCarthy, A. Newell, and F. Rosenblatt.

PREFACE

There are three basic approaches to the functional analysis of intelligence, as represented by experimental psychology, pragmatic projects in artificial intelligence, and formal analytical inquiries into the representation and use of knowledge structures. The current work is certainly most associated with the last approach, but its relevance to workers in all three areas seems significant. This relevance derives from the simplicity, generality, and practicality of the theoretical results presented herein. It is intended that this paper be readable by any consenting adult, regardless of his or her prior level of familiarity with artificial intelligence, cognitive psychology, epistemology, or computer science.

The original learning and representational problem which has motivated my research for several years deserves mention here. When one considers how a natural organism like a poorly developed human neonate can acquire vast knowledge and skills, a very simple problem formulation suggests itself. At every instant in time, the infant's internal state (memory) must reflect the coded representation of the information available to it: sensory information, anxieties, motives, muscular and joint position information, memories of recent events, and so forth. At each succeeding moment, an updated version of these conditions replaces the prior information. Contained in this

information are the neural firing patterns that determine behavior. If one wished to model or simulate the infant's total behavior, one would need only supply the same temporal series of information and interpret appropriate components as corresponding behaviors. Or--and here is the crux of the issue--if one wished to learn to behave as the infant does, one could succeed by remembering what that temporal series of information was and subsequently using it in the obvious way to determine future behavior.

Three observations further constrain this analysis. First, the child itself is obviously in a situation quite similar to the one proposed: it wants to learn to behave in particular stereotyped ways. What it wants is determined by interactions among innate preferences and environmental events, including conditioning stimuli (reinforcements). Whether or not this teleological interpretation has merit, the infant does develop stable behavior sequences which are prima facie evidence of the sort of learning being considered. Second, there is a distinction to be made between the internal representations of external stimulation and those directly associated with the production of cognitive or muscular responses. The problem of learning is primarily concerned with the production of responses appropriate to a variety of stimulus situations. What are controllable, in other words, are internal responses and, as a result, response learning is the crucial concern. Third, no two moments in an orga-

nism's history are likely to be identical. This observation illuminates the infeasibility of a simulation of behavior based on simple information retrieval schemes like the following: change the internal state from one moment to the next by finding the record of a previous experience which is identical to the current one and reinstating the information which is contained by the subsequently recorded one. If no two instantaneous records are identical, it must be necessary to abstract generalized state changing rules which are applicable to classes of situations and, conversely, are insensitive to changes among irrelevant variables.

These observations lead directly to a statement of the fundamental learning problem. At each moment of time, say t, the internal state of the organism consists of information about the contents of memory, sensory stimulation, and the condition of muscles, organs, etc., denoted collectively as S_t, and about the cognitive, muscular, and behavioral responses, denoted R_t, which are made as a result of the occurrence of this internal state. The pair (S_t, R_t), then, is a complete description of the behavior of the organism at time t. In terms of such descriptions, the fundamental learning problem is easily stated:

> Determine a rule (function) T which when applied to S_t produces R_t, regardless of the specific value of t. Symbolically, $T(S_t) = R_t$ for all t.

Like many others, my original approach to this prob-

lem was straightforward. Noting that the number of possible hypothetical rules which one could evaluate for feasibility was astronomical, I attempted to approximate the ideal solution. (The reason for emphasis on the word approximate will become apparent momentarily.) The "obvious" way to approximate the solution was to choose arbitrarily and hopefully small subsets of S_t and R_t and try to relate these subsets by a consistent rule. The vast number of possibilities was no smaller because of this approach, but the complexity of the rules was greatly constrained. This subset-to-subset rule hypothesis generating technique could be effectively controlled to insure that only simple hypotheses were produced. Many simple rules could then be used simultaneously to approximate assumedly more complex rules.

John Holland convinced me that this approach was unsatisfactory. There was no strong a priori reason, he argued, to suppose that behavior rules could be approximated by collections of separate, independent, and probabilistically (less than certainly) valid subrules. He urged me to pursue a frontal attack on the original problem. I interpreted this advice as follows: using all available information in parallel (simultaneously), decompose the (S_t, R_t) pairs into reliable subrules in which the selected complex subsets of S_t and R_t are identified because they are reliably effective models of behavior rule components. Thus, of the two possible

approaches to the problem, approximation based on a priori intuitions or sampling algorithms or decomposition based on a posteriori analysis of the entire set of training information, Holland argued for the latter. We both agreed that decomposition was the only certain way to produce a correct learning procedure, but its nature and feasibility remained to be discovered. Happily, the bulk of this paper is addressed to the exposition of such learning procedures and their application to important and previously unsolved problems.

Another difference between the approach pursued in this paper and more naive attempts at representation of knowledge and learning is the current emphasis on structure. Structure, in the present context, signifies relational connections among data items. Thus, I will be concerned with the representation and acquisition of rules of behavior of the most general type, rules associating relational configurations of objects and events occurring as stimuli to comparably complex response structures.

Much previous research on behavior modeling has been naive regarding structure in either of two ways. First, many things which occur in the world and whose recognition or cognitive manipulation have been extensive studied (e.g. a capital "A"), have rarely been adequately described as complex pattern structures. Second, efforts to induce one structure from another or evolve a new one (e.g. a finite state automaton to perform a particular computation) have

been conducted completely without any informed appreciation of the relationship between structural similarity and functional similarity. Both kinds of issues are considered and analyzed in this paper.

Finally, I would like to acknowledge the role played in my research by good fortune. Both in the intuitive appeal which the fundamental learning problem has always held for me because of its elegant simplicity and in the ultimate tractability of this problem to persistent and stubborn analytical effort do I see tremendous good luck.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGMENTS iii

PREFACE iv

LIST OF TABLES xvi

LIST OF ILLUSTRATIONS xvii

LIST OF APPENDICES xviii

LIST OF ABBREVIATIONS xix

CHAPTER 1: THE NATURE OF INTELLIGENCE 1

1.1 Chapter Overview 1

1.2 Organism and Environment: The Learner and the Trainer 1

1.3 Defining Intelligence 7

1.4 The Proposed Approach to Modeling Intelligent Behavior 14

1.5 Examples of Intelligent Behavior Modeled in this Paper 17

1.6 Objectives 18

CHAPTER 2: PERCEPTION AND COGNITION: REVIEW OF RELEVANT LITERATURE 21

2.1 Chapter Overview 21

2.2 The Perception Problem: Pattern Representation, Learning, and Recognition .. 22

2.3 The Cognition Problem: Rule Representation, Acquisition and Use in the Control of Behavior 35

CHAPTER 3: THE STRUCTURAL REPRESENTATION (SR) STIMULUS-DRIVEN COMPUTATION SYSTEM 46

3.1 Chapter Overview 46
3.2 Parameterized Structural Representations (PSRs) of Events 46
3.3 Representation of Ideals (Patterns, Concepts, Prototypes) 56
3.4 Representing Behavior Rules: Contingency-Response Transformations (TXs) 64
3.5 Principles of Computing in the SR System ... 68
Encoding 71
Differentiated memories, communication, and reconstruction 72
Competitive response classes: mechanisms of choice 74
Cooperative response classes: the realization of disjunction 76
Hierarchies and goal-oriented control structures 78
Chains or programs 80
The integrality of memory and the transformation of structure 82
Untreated aspects of memory control ... 84
3.6 Relationship Between the SR System and Other Computing Frameworks 85
CHAPTER 4: PERCEPTION AND PATTERN LEARNING 88
4.1 Chapter Overview 88
4.2 The Recognition Problem 88
4.3 The Learning Problem Conceptualized 90

4.4 An Overview of the Learning Problem Solution 93
4.5 Solving the Pattern Learning Problem Using Interference Matching, a Mechanism of Abstraction 94
Categorical behavior 94
Models 95
Partial-matching using the IM Procedure 96
Schematic classification theory 99
4.6 Recognition Using Interference Matching 115
4.7 More Detail on PSRs for Matching: Uniform Representations 116
4.8 Feature Coding + Pattern Learning + Pattern Recognition + Consciousness = Perception ... 119
CHAPTER 5: DIRECT MEMORY CONTACT MATCHING FOR IMMEDIATE BEHAVIOR 123
5.1 Chapter Overview 123
5.2 The Fundamental Stimulus-Response Idea: Stimulus-Driven Computation 124
5.3 Recognition as a Problem of Graph Matching . 126
5.4 Criteria for an SDC Memory 127
5.5 Implications of the Subordinate Congruence Memory Design Criterion 130
5.6 SANER Nets 131
5.7 Canonical Abstraction Powersets 137
5.8 Conclusions: Powersets and Matching 140
CHAPTER 6: SUBSTITUTION TRANSFORMATIONAL BEHAVIOR AND ITS ACQUISITION 143

6.1 Chapter Overview 143
6.2 Transformational Grammar as a Typical Problem 144
6.3 Representing Transformational Rules in the SR System 148
6.4 The Learning Problem: Grammatical Inference 150
Prediction-matching: the first step in rule learning 151
Generalization-matching: the second step in rule learning 153
6.5 Generality of the Prediction-Plus-Generalization-Matching Procedure 155
CHAPTER 7: PROGRAMMATIC BEHAVIOR AND ACQUISITION: LANGUAGE LEARNING 158
7.1 Chapter Overview 158
7.2 Outstanding Top-Down and Hybrid Language Understanders 158
7.3 Top-Down Understanding: "Pick up a big red 162
block" 162
7.4 SDC as a Natural, Efficient, Learnable Basis for Bottom-up Understanding 167
Representation of a bottom-up understander 167
(1) An SDC understanding system is simpler and more efficient than the alternatives 171
(2) The comparability of temporal

behavior sequences of SDC and its alternatives 173
(3) SDC systems are easily extended ... 174
(4) Learning language processing rules in an SDC system 175
7.5 Reconstruction, Supplementary Information, and Trial-and-Error Learning 177
CHAPTER 8: EXPANDING KNOWLEDGE 185
8.1 Chapter Overview 185
8.2 Creating New Predicates: The Representation of Patterns 186
8.3 Learning Concept Domains and Categories 190
8.4 Generalization over Members of the Same Category 196
8.5 Learning to learn 198
Attentional contol 199
Problem solving heurisitics..... 200
Problem solving algorithms 201
8.6 Summary 202
CHAPTER 9: CONCLUSIONS 203
9.1 Chapter Overview 203
9.2 Mechanisms of Intelligent Behavior 203
9.3 Major Deficiencies of the Current Work 207
9.4 Suggested Guidelines for Future Research ... 209
CHAPTER 10: EPILOGUE: SPECULATIONS ON NEURAL INTERFERENCE PROCESSES AS NATURE'S SOLUTION TO THE FUNDAMENTAL PROBLEM 212

10.1 Chapter Overview 212
10.2 Speculations on Neural Representations 212
10.3 Causal Regression and Learning 215
10.4 The Hypothesized Nature of Neural Computation 217
10.5 Conclusions 219
APPENDIX 221
BIBLIOGRAPHY 279

LIST OF TABLES

Table 1 100

LIST OF ILLUSTRATIONS

Figure 1 4

Figure 2 50

Figure 3 51

Figure 4 52

Figure 5 58

Figure 6 60

Figure 7 62

Figure 8 69

Figure 9 91

Figure 10 99

Figure 11 133

Figure 12 145

Figure 13 146

Figure 14 147

Figure 15 164

Figure 16 233

Figure 17 241

LIST OF APPENDICES

APPENDIX I: SPACE LIMITED INTERFERENCE MATCHING MODEL BUILDER (SLIMMOB)

A procedure for the generation of abstractions of structured (relational) event descriptions which is operable even when computing space is limited 221

APPENDIX II: SCHEMATIC CLASSIFICATION PERFORMANCE MEASURES

The derivation of a Bayesian posterior expected gain measure which estimates the expected value of schematic classification rules of the sort, "If a stimulus S has characteristic T, classify S in class C" 226

APPENDIX III: CANONICAL FORMS OF PARAMETERIZED STRUCTURAL REPRESENTATIONS

A procedure for the successive numbering of each parameter in a parameterized structural representation enabling each parameter to be replaced by a distinct constant, thereby obviating combinatorial isomorphism procedures 231

APPENDIX IV: A FORMAL ANALYSIS OF THE PRINCIPAL RESULTS

Mathematical proofs of the computational universality of the SR system and effectiveness of the interference matching procedure applied to uniform representations for abstracting pattern characteristics and substitution transformations from training information, and the relationship between the SR system and the first-order predicate calculus 243

LIST OF ABBREVIATIONS

$\in$ is an element of

$|x|$ absolute value of x or cardinality of the set x

$\{x : P(x)\}$ the set of all x such that P(x) is true

$X \sim F$ the random variable X is distributed as F

$X - Y$ the set difference of X and Y (all members of X not contained in Y)

$\binom{a}{b}$ the combinatorial (binomial coefficient) of a things taken b at a time

S (*) T means the representation S matches (contains as a subrepresentation that of) T

$\Rightarrow$ logically implies

& logical conjunction

$\sim$ logical negation

$\Leftrightarrow$ if and only if

$\equiv$ is identical, equivalent

Chapter 1: THE NATURE OF INTELLIGENCE

1.1 Chapter Overview

Whatever the offenses risked, no distinction is made here between natural organisms and artificial information processing systems like computers. For the current purposes, there are only two "beings" of importance, the organism and its environment. Intelligence is defined in terms of what knowledge the organism possesses, desirability of behaviors based on that knowledge vis-a-vis systematic and variable environmental reinforcement, the organism's adaptive capacity to select preferred behaviors over less rewarding ones, and its ability to acquire new knowledge in appropriately structured training environments. Behavior is described in terms of two basic mechanisms, representation and computation. A brief outline of the structural approach to the representation of knowledge and behavior which is used throughout the paper is presented. Several important exemplary problems which are analyzable in this framework are then suggested, including complex pattern learning and recognition, substitution transformational rule acquisition and behavior, and general programmatic rule acquisition and behavior (e.g. language understanding). Finally, the objectives and non-objectives of the paper are enumerated.

1.2 Organism and Environment: The Learner and the Trainer

The organism is viewed in the current paper as an information processing system which interacts with (manipulates and is affected by) another entity, the environment. The organism's behavior is a term encompassing all of its internal and external responses over time. These responses are events which occur whenever the combination of internal and external conditions are such that some aspect of the organisms's overall physical state changes. Behavior will be analyzed in terms of its abstract analogue, computational systems of equivalent functional complexity.

Particular environmental events are assumed to produce desirable experiences for the organism, while other events are assumed to produce undesirable consequences. The successful (happy) organism is one which modifies its behaviors in whatever way possible to increase the likelihood and magnitude of desirable experiences. Equivalently, a successful self-modifying computing system is one whose behavior increases the likelihood and magnitude of those experiences arbitrarily deemed "good" which are administered to it by an external agent, the trainer. Such a computing system provides a model (scientific theory) of the comparably successful learning behavior in natural organisms in so far as it explains previously observed phenomena and predicts yet untested characteristics of that behavior. Thus, the study of intelligent information processing systems, the central topic of this paper, is of direct relevance to the psychology of natural systems

(Newell, 1970; Moore & Newell, 1973).

The principal elements of the adaptive (learning) information processing system are illustrated in figure 1. The major components considered here include sensors, memory/logic, motivational control, and effectors. Sensors are instruments which respond to environmental stimultation (energy) and internal (e.g. biological) conditions by generating corresponding information patterns of some sort; typically, these are time series of neuronal action potentials or graded potentials in natural systems and electrical current changes in artificial systems. Sensors are the input devices of the organism's information processing system.

The memory/logic unit controls behavior. Logic rules the temporal sequence of changes in memory. Memory retains information about previous and current information patterns and rules describing related responses which the organism can currently make, has previously made, or might make in the future to such stimulation.

Traditionally, information processing systems are viewed as mathematical structures in terms of a set of possible states (discriminable overall physical conditions) and rules relating the orderly transition from one state to the next (Arbib, 1964; Minsky, 1967; Hopcroft & Ullman, 1969). One difficulty with this approach is an implicit confusion of the two components of state: memory of stored information and memory (determination) of the

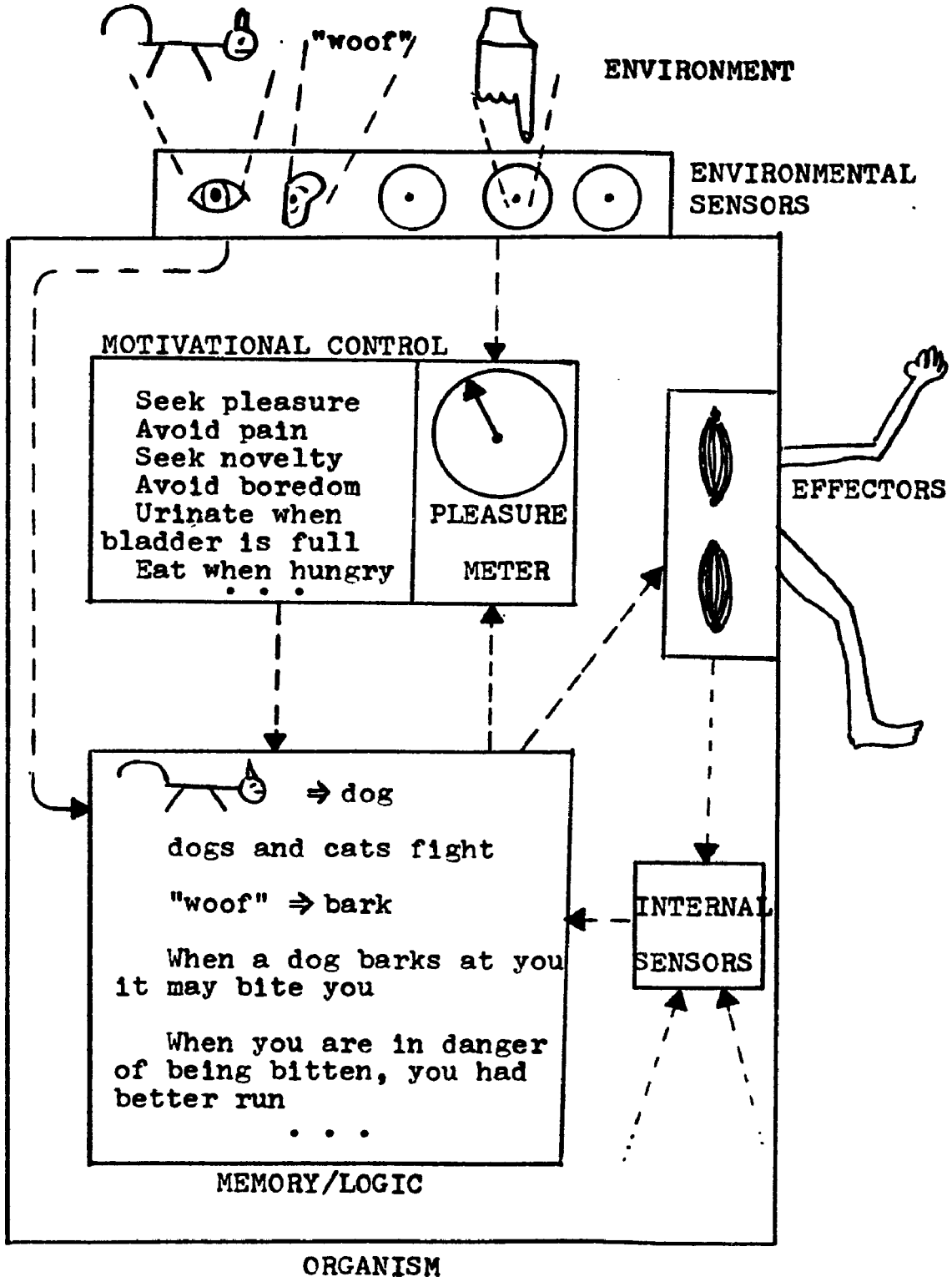


Figure 1. A schematic diagram of the principal components of the organism's information processing system: sensors (internal and external, e.g. eye, ear, tactile sensors), motivational control, effectors, and memory/logic.

next state alternatives. A state is defined both in terms of past and future potential transitions and in terms of the previous input information presented and stored. A second difficulty results from the virtual impossibility of representing the states of any large scale information processing system in anything like a table of possibilities. There are simply too many of them. Moreover, states are apparently highly organized and decomposable because of the enormous amount of structure inherent in the data which they are used to represent. In traditional analytical systems, there is no apparent way to exploit such structure. A framework without these difficulties is proposed here. It is functionally equivalent to the traditional one but is more intuitive and more useful for the current analytical purposes.

The organism is supposed to possess memory capacity, for example in terms of an unlimited supply of very large, primarily empty mailboxes or unexposed photographic plates, and the memory of the organism at any time is simply the current contents of these storage units. The logic of the organism is a set of rules (contained in memory) which dictate how the contents of memory are to be changed whenever particular contingencies occur. All contingencies are statements (propositions) about the contents of memory. Some contingencies are descriptions of current sensations; some describe currently active ideas or inactive ones representing previously learned facts; others describe current

conditions of body limbs and effector muscles. Responses to recognized occurrences of specific contingencies may produce observable behaviors through the intermediate control of effectors (e.g. muscles) which physically affect the external environment or they may produce internal changes of a more symbolic kind. In short, the input, processing, and output functions of the sort of organisms considered here are associated with the sensors, memory/logic, and effectors respectively. These three elements are essential to any information processing system which is responsive to variations in its environment.

To provide an account of the mechanisms of adaptive behavior, it is necessary to associate a motivational control component with the information processing system. The organism must have some means (a pleasure meter) of measuring the goodness, desirability, or pleasurability of alternative situations if it is supposed to exhibit selectivity in its behavior towards rewarded experiences. Most natural systems are easily conditioned by a variety of response-contingent reinforcement schedules. Reinforcers are simply the external correspondent of internal information patterns producing differential desirability evaluations and resultant response preferences. The need for providing an artificial learning system this sort of motivational control is obvious. Any computing system does only what it is programmed to do; one might say it has a very strong preference to choose the behavior it has been

programmed to perform. In the case of an artificial information processing system like this, the distinction between the motivational control and memory/logic components might be viewed as somewhat contrived. Nevertheless, the function played by motivational control is easily separated from the cognitive functions of interest, perceiving, reasoning, planning, and acting. It is because the major focus of this paper is a functional analysis of such cognitive behaviors that the distinction between memory/logic and motivational control is proposed.

In sum, the organism is viewed as possessing the capacity to receive energy, encode it as information, cognize about it or compute, issue responses, and experience some measure of the desirability of its physical condition in an environment. How these functions are combined to create intelligence is the subject of the following section.

1.3 Defining Intelligence

In this section the concept of intelligence is defined in terms of the sorts of behaviors it makes possible. The main ideas here have been previously explored in greater detail elsewhere (Hayes-Roth, 1971, 1973a). Because an appreciation of intelligent behaviors requires an understanding of knowledge, its expansion, and its use in adaptive behavior, these notions are discussed first.

Knowledge is defined to be the representation of a computation which is of some interest with respect to some

purpose. Because representations may be abstract or concrete, a theoretically describable mechanism which effects such a computation is also a form of knowledge. Some examples of knowledge and their interpretation in this framework should illuminate the major concepts. Note that, within this framework, any interesting behavior is prima facie evidence of the existence of knowledge. The correct recognition and naming of a visually presented alphabetic symbol (with appropriate experimental controls) is evidence that the organism knows the symbol and its name; its knowledge is the representation of the symbol, the name, and the computational rules that permit it to encode the stimuous character, compare it to the known symbol, decide the outcome of that comparison, and issue the appropriate naming response. All knowledge is ultimately independent of the organism itself. It could conceivably be written and preserved on paper, programmed in a computer, or otherwise transmitted to another organism. The only apparent dependencies between the knowledge and the organism are due to the possible existence of characteristics of the information representations which are peculiar to the particular organism. Even these apparent dependencies may eventually be eliminated through identification of abstract and functionally equivalent computations. In the case of letter recognition and naming, one such possible abstract knowledge is easily described (see chapter 3).

Another interesting example is knowledge of a lan-

guage. One knows a language when one uses it in speech production or recognition, reading, writing, or gestural production. The knowledge of a language is not the collection of these behaviors but is, rather, the computational system which produces them. Such knowledge consists of computations and memory structures which represent and manipulate words, concepts, syntactic categories, semantic relations, and notions of grammaticality. For the purpose of understanding linguistic knowledge, these behavior producing entities are considered equivalent to a representation of an entirely abstract computing mechanism which performs comparable manipulations on comparable information structures. Such knowledge has already been largely represented on paper and in computers as programs (Chomsky, 1965, 1967, 1968; Woods & Kaplan, 1971; Winograd, 1972; Schank, 1972). What has not yet been codified can surely be completed in theory, if Church's thesis and Turing's thesis are correct (Minsky, 1967). The central problem underlying the difficulty in completing the codification is not that abstract computation is an inadequate representational basis for the knowledge we possess. Rather, the problem is that our present knowledge is not yet adequate for the task of producing an artificial representation of itself <u>in vitro</u>. The complete knowledge of a language is represented by our physical, neural structures, but is not in a form which is currently translatable by us.

Every information processing organism is originally

provided an endowment of computational capabilities. These include its sensors, memory capacity, logical operators, motivational control, and effectors. This endowment determines the representational elements used to encode information and the rules dictating how information may be used (organized, stored, retrieved, integrated). Because the organism's behavior within an environment is controlled by the contents of memory and its memory altering transformational rules, the endowment is a principal determinant of which behaviors may ever be evidenced by the organism. For example, human speech may not be accessible to chimpanzees because they lack an adequate vocal apparatus, even though they can apparently acquire functionally equivalent linguistic knowledge (Gardner & Gardner, 1969; Premack, 1971). These potentialities and limitations are directly determined by endowment. The skills which are actually acquired are determined by the joint influences of the endowment and environmental training.

The primary reason for introducing the concept of endowment here is not to investigate the interesting questions relating to the relative effects of endowment and environment but to distinguish two critically different levels of endowment in terms of the behaviors they make possible. A universal computation endowment is one which makes possible the representation of any well defined computational procedure or, equivalently, the possession of any available knowledge. Such an endowment is evident in any system capable of simulating the behavior of an arbi-

trary computer program or Turing machine. It is also evident in organisms which can read and perform well defined instructions in a book. However, it is not possible to conclude that organisms which lack these particular capabilities have a sub-universal computation endowment. These systems may actually be capable, in well designed training environments, of exhibiting behaviors of equivalent complexity which are homomorphic to the original test behaviors. Although the concept of a universal computation endowment is analytically useful, the only apparently adequate method of determining the existence of such a capacity in an organism is to demonstrate it directly and affirmatively.

This suggested method is further complicated by the related necessity to communicate and explain any task to the organism and, in so doing, to transfer from the environment to the organism whatever knowledge is required to perform the task successfully. A failure to perform successfully could be the result of a sub-universal endowment, inadequate task explanation or communication, or inadequate acquisitive capabilities. The successful performance of an arbitrary communicated task, on the other hand, is evidence that appropriate representations and transformations have been formed, acquired, retained and exploited. The formation of such new knowledge is referred to as ideogenesis. It is the first of two types of learning considered in this paper. In brief, ideogenesis is the

construction of knowledge, the formation of representations or structures for the performance of computations of interest.

The second type of learning is adaptation, the increased probabilistic selection of reliably more highly reinforced behavioral alternatives within a set of mutually exclusive response possibilities. Examples of adaptation can be seen in each of the following: the eventual choice of a reliably higher paying gamble; the development of reliably better (more successful) strokes in tennis; and the measured improvement of an animal's behavior produced by continuous shaping (a program of successively increased reward criteria in response-contingent reinforcement). The notion here is that choices must frequently be made between competitive (mutually exclusive) response alternatives on the basis of reinforcement or feedback provided by the environment on the desirability of any particular chosen response. It is unusual that a single response trial and its attendant reinforcement provide adequate information to know which of the alternatives ought to be selected in the future. It is usually the case however--this is a restatement of the induction principle--that improvements can be made by guiding future response selections by statistical inference based upon previous feedback information. An organism is adaptive to the extent that it exhibits over time an increasing probabilistic selection of more reinforced alternatives.

In terms of the above constructs, a definition of intelligence is straightforward. Intelligence is defined to be a set of three necessarily co-occurring properties: a universal computation endowment; a realizable capacity for ideogenesis of any arbitrary, well defined, and physically possible procedure in at least one appropriately structured training environment; and adaptation. In other words, an intelligent information processing system is one which can be taught to do something of any computational complexity provided only that needed ancillary materials are provided and that the training procedure is appropriately structured from the viewpoint of the learner's two types of learning mechanisms. It is important to read this definition precisely; nothing is claimed about the necessity for an organism to be able to learn from a training procedure which the trainer concludes ought to be adequate. Such questions are within the framework of educational psychology rather than one addressed to fundamental mechanisms of intelligence. The important point for the purposes of this paper is that intelligence itself is understandable and operationalizable in terms of clearly defined mechanisms and concepts, including: information, computation, memory, logic, motivation, reinforcement feedback, ideogenesis, and adaptation. Having neatly characterized what intelligence means in terms of gross trainer-learner interactions, it is now possible to proceed to identify and describe, in precise operational terms, what are considered the fundamental

mechanisms of intelligent behavior. Several of these will be analyzed in greater detail in subsequent chapters.

1.4 The Proposed Approach to Modeling Intelligent Behavior

Intelligent behavior is anything that could be observed about the successive physical conditions of an intelligent organism, excluding things out of its control. The generality of this statement reflects the descriptive weakness of the term intelligent behavior. Nevertheless, it is possible to identify some fundamental mechanisms of behavior, like those already suggested in considering the elements of an adaptive information processing system, and to relate these behavioral control mechanisms subsequently to potential mechanisms of learning, the latter being of paramount interest here. This is, in point of fact, the principle about which the entire paper is organized. In the current section, however, the later developments are foreshadowed as the focal points of the investigation are illuminated.

The fundamental mechanisms of intelligent behavior are representation and computation. Representation refers to the conception and properties of structures corresponding to varying types of sensory information, ideal or known patterns, contents of memory, and rules governing alterations of the contents of memory. Computation refers to the temporal control of the contents of memory based on

the storage, retrieval, generation and comparison of information structures in that memory. Any further abstract discussion of these mechanisms would provide little insight into the interesting aspects of intelligent behavior. Instead, the actual framework used through the paper is now briefly described so that a more tangible context is provided.

The major components of the theory are easily described. The organism's environment is encoded through the sensors and represented internally in terms of events, sets of predicate relations among objects. Rules of behavior are stored in memory in terms of transformations comprising contingency-response pairs. All behavior is computed in the same simple way. Those rules whose contingencies are matched by the current stimulus information or the current internal computational conditions are invoked and their response components performed. A different kind of matching is also important for the production of abstractions of classes of related patterns. The production of abstractions is the central problem in the acquisition of new knowledge. Rules which expand existing knowledge can be produced in a straightforward manner by abstracting the commonalities of diverse events associated with similar responses and employing these commonalities as the contingency in appropriate, novel, generalized transformations.

In the proposed representation scheme, events are

described in terms of primitive logical predicates which interrelate objects in spatial, logical, temporal, activity or computational configurations. Knowledge is represented as a set of related events and transformations between events. Such a representation system is a natural basis for the description of patterns and computational rules of behavior. The chief difficulty in the proposed system results from the necessary introduction of parameters, arbitrarily chosen, meaningless symbols used to denote distinct objects which must be consistently referenced in the several relations in which they participate.

The proposed representation system is like the lambda-notation developed by Church (1941) and prevalent in LISP (McCarthy, et al., 1962; Weissman, 1967). The primary differences between the current theory and those of typical LISP-based implementations of artificial intelligence systems (Winston, 1970; Woods & Kaplan, 1971; Winograd, 1972; Rulifson, et al., 1972) are the explicit focuses in the current paper upon: the structure of integrated pattern representations; the logical matching of complicated stimulus and ideal representations for pattern recognition and general behavior control; and the definition and use of mechanisms of abstraction for the identification of structured subrepresentations common to members of a class of related events and for ideogenesis, the formation of new behavior rules. In addition, the current theory provides a natural framework for the representation and

understanding of principles of knowledge organization, including the following notable issues: the competive selection of decision alternatives; the cooperation of several rules implying the same response; and the overall homogenizing unification of knowledge in terms of events (facts) and transformations sharing a common memory and a common representation system.

1.5 Examples of Intelligent Behavior Modeled in this Paper

Three types of behavior are investigated in this paper. Categorical behavior is characterized by a uniform, constant response to a class of stimulus situations which all exhibit the common property defined by the contingency of the related transformation rule. Illustrative problems which are considered in some detail here are the immediate recognition of a structured pattern like an alphabetic symbol or the acquisition and recognition of a seemingly more complex abstract concept. These problems and their solutions are the topics of Chapters 3 and 4. Chapter 5 presents some novel results bearing on the design of immediate recognition systems in which structured stimulus patterns make direct memory contact with corresponding structured templates.

Substitution behavior refers to rules in which some components are extracted from one information structure and (logically) inserted into another. This sort of behavior is most commonly encountered in the theory of trans-

formational grammar. The first (stimulus) structure in that case is a sentence which evidences certain grammatical characteristics. Portions of that structure (e.g. a noun group or a subject) are transfered to specific positions within a novel (response) structure which represents a grammatical alternative to the original sentence. The representation, acquisition and use of such substitution behavior rules is described in Chapter 6.

More complex behaviors like those of language understanding and production are widely considered to be programmatic, involving hierarchical computational organization. In Chapter 7 the representation and organization of language knowledge is discussed as an example. The ideogenesis of complex rules of behavior functionally comparable to those involved in language understanding is then discussed. It is one of the principal results of this work (App. IV) that the same mechanisms of abstraction shown to be adequate for the acquisition of rules of the categorical and substitutional types are equally adequate for the development of language understanding or any other well defined procedure. The expansion of knowledge through mechanisms of category formation and generalization are the principal problems considered in Chapter 8.

1.6 Objectives

Several goals motivate this work. The paper is intended to provide a framework and terminology for the analysis

of the aforesaid concepts relating to knowledge and its acquisition. The framework is adequate for the representation of all programmatic knowledges and concept or pattern ideals. A functional explanation of structured pattern recognition is provided. A design for the construction of memories is described which would permit direct memory contact for the (associative) recognition of any structured pattern. A definition of abstraction is given in terms of easily understandable and widely applicable algorithms. Theoretical mechanisms for rule learning are proposed. An expected gain maximizing rule for adaptation within the proposed structural representation framework is developed. Finally, the adoption of stimulus-driven (bottom-up) computation, as developed in this paper, as a general model of intelligent behavior is advocated because of its preferability over more prevalent procedural (top-down) representations of knowledge.

There are numerous important problems directly related to the modeling of natural intelligence which are not addressed in this paper. These include: sensory coding and codes; physiological and neuronal bases for computation (save for several speculations in Chapter 10); realization of adaptation procedures; the capacities, relative merits, physical attributes, consolidation and other structural changes of natural memory stores or communication among such stores; pragmatic approaches to the construction of intelligent systems using existing computational machinery; the wealth of knowledge relating to non-digital control

processes. The major unsolved problems and deficiencies of the proposed theory are enumerated in Chapter 9.

The central objective of this paper is, in brief, to provide a formal system for the analysis of the problems of representation, computation and learning, without necessarily providing a testable scientific theory for human intelligence. In order to construct a powerful computing system of the type considered in this paper, one would need vast amounts of computing power. Thus, it is not intended that such systems be immediately constructable and testable. Rather, it is intended that several important analytical questions concerning the nature of intelligence can be profitably studied (formally) in the framework developed herein.

In the next chapter, the foundation for the proposed theory is laid by reviewing the principal previous contributions to the theory of the fundamental mechanisms of intelligent behavior under consideration: representation, abstraction, and recognition of structured patterns and the transformation of such patterns to represent behavior.

Chapter 2: PERCEPTION AND COGNITION: A REVIEW OF RELEVANT LITERATURE

2.1 Chapter Overview

In this chapter two central problems are identified and previous approaches to their solution are briefly reviewed. The perception problem concerns the representation, acquisition, and recognition of structured patterns. The advantages and disadvantages of four major approaches to the perception problem are discussed. These are associated with template, spatial, feature, and generative models of patterns and recognition processes. The cognition problem concerns the more general control of computation or behavior within a particular representational framework. The primitive analytical view of computation in terms of a machine with a state set and transition function is first considered. It is suggested that such a view ignores the principal issues in the realization of actual computing systems: the determination of a suitable representation of state, the design of a decision procedure to effect the transitions from state to state, and the extension of the state set and transition function to reflect new computing power (knowledge). A number of alternative general computing systems are then analyzed with respect to these major design issues. The capabilities and deficiencies noted in this chapter provide a sound basis for the evaluation of the structural representation computing system proposed and employed in subsequent chapters

as a basis for the description of mechanisms of intelligent behavior.

2.2 The Perception Problem: Pattern Representation, Learning, and Recognition

The objective of this section is to formulate a model of what an organism must do in order to classify a stimulus as an exemplar of a previously learned pattern. Thus it is necessary to describe how stimuli are coded into pattern representations, how the representation of a stimulus is compared to a known pattern ideal or prototype representation, and how such ideas are originally abstracted from the organism's experience with the environment. These questions constitute the problem of perception.

It is helpful in evaluating alternative approaches to the perception problem to introduce the concept of a knowledge model, a theory about the relation between data and a process which generates them (Hayes-Roth, 1974b). Although the various approaches to the perception problem which are reviewed in this section are usually classified on the basis of the mathematical procedures they employ for pattern recognition, an analysis based on differences among their associated knowledge models is preferable. It is less useful for a detailed discrimination of specific procedures, but such an analysis provides additional insights into the relative merits of whole classes of theoretical models of perceptual skills. The four categories reviewed are associated with

four distinct knowledge models: the template model, spatial model, feature model, and generative model. Each of these is considered in turn.

A template model is any theory which proposes that pattern recognition occurs because a stimulus representation is directly compared to a completely homologous, stored memory pattern prototype representation and that the results of such a comparison are immediately computed by very simple, immediate, and memory-less devices like threshold gates (see for examples, Fischer, et al., 1962; Rabinow, 1968; Lindsay & Norman, 1972). Graphic template models are the best known exemplars of this class. They propose that stimuli (e.g. examples of the capital letter "A") are generated by some stencil, photoreproductive, or copying process and are recognized because the internal visual representations which they produce in the organism are very similar to previously memorized templates (e.g. a typical letter "A") to which they are compared by a simple rule: the light must be on in the stimulus where the stencil is opaque and off where it is transparent (or vice versa).

The primary advantages of such a recognition procedure are three. First, the comparison between stimulus and template patterns depends on all of the criterial structure and attributes of the pattern. That is, a stimulus is classified as an "A" only if it exhibits the same component lines and vertices in the same configuration as in the template. Second, a stimulus could conceivably be compared to all

memory templates simultaneously because there supposedly are a limited number of templates which can be stored and evaluated completely independently on one another. Third, the recognition procedure is conceived as one of direct memory contact. The stimulus representation is considered to be congruent with and somehow activate the memory template it matches. Thus, recognition is virtually immediate.

The disadvantages of template matching procedures are chiefly associated with the template's unsuitability as a basis for generalization in pattern recognition. Even minor deviations between a stimulus and a template (e.g. differences in proportion, size, orientation, font) may completely inhibit proper recognition. Moreover, template models have proved nearly intractable to learning procedures. Galton's (1907) composite photograph theory, proposing that many exemplars of the same pattern class merge together somehow to produce a composite pattern in which commonalities are retained and individual dissimilarities are eliminated, is among the most fully operationalized proposed learning schemes for this sort of recognition model. Beyond this, Uhr (1973) presents a program for template learning in the simplified context of character string stimuli and a device which can memorize whole stimuli or pieces of them which are thereafter used as recognition templates.

Theories of the second type, spatial models, propose that a stimulus be represented as a point in an N-dimensional space, where each dimension corresponds to a measurable

attribute (feature, unary predicate, property) of the stimulus and the corresponding attribute value is a measure of its magnitude. Known patterns are represented either by an ideal point (a vector of N coordinates) or a volume of points in the space defining all possible stimuli which would be considered examples of that pattern. In general, the likelihood that a previously unclassified stimulus is an exemplar of any particular pattern class is inversely related to the distance between the vector representation of the stimulus (a point in N-dimensional space) and the locus of the corresponding pattern ideal or previously classified exemplars of that class. The underlying analogical conception seems to be that a stimulus is ballistically produced, as if a projectile aimed toward an ideal (target) point in N-dimensional space. Differences between the realized coordinates of the stimulus and the ideal are viewed as errors in its trajectory. In fact, most spatial models consider the errors in each dimension to be independent of the others, although pairwise covariances are sometime considered.

These models are by far the most widely employed and analytically developed. They underlie scaling theory approaches (Torgerson, 1958), signal detection theory (Swets, 1964; Green & Swets, 1966), factor analysis (Thurstone, 1947; Harman, 1967), correlational methods (Prewitt, 1970; Uttal, 1973), discriminant analysis (Patrick, 1972; Meisel, 1972), most likelihood ratio, probabilistic and decision theoretic approaches (Ferguson, 1967; Smith, 1962; Fu, 1968, 1971;

Reed, 1973), and perceptron theory (Rosenblatt, 1958; Minsky & Papert, 1969; Block, 1970; Nilsson, 1965). These models have the simplicity of representation and matching operations as their outstanding quality. Each stimulus is represented as a vector of N measurements and then compared to pattern ideals by computing an arbitrary distance function or by locating the stimulus representation within a corresponding pattern volume according to its coordinates.

The primary disadvantages of spatial models are a corollary of this simplicity. They are generally incapable of representing criterial value dependencies among a subset of stimulus attributes. They are also unable to reflect that, although a particular set of feature values is relevant to one pattern class, those features may be completely irrelevant to the definition of other classes. One way in which this can be seen is by considering the pattern volume corresponding to each pattern class. In spatial models, the procedure of classifying all stimuli close to one prototype as members of the corresponding class defines a non-zero volume of points around each prototype which represents the pattern class. If particular feature values were irrelevant to some class, as "amount of blue paint" is irrelevant to the pattern class "Volvo," a decision procedure should ignore "amount of blue paint" in considering the possibility of classifying objects as "Volvo." The appropriate pattern volume in such a case would then be a zero-volume hyperarea

of some dimensionality less than that of the entire pattern space (Hayes-Roth, 1973b, 1974b); Michalski, 1973; Watanabe, 1973). Furthermore, these models have no means of expressing structural relations among pattern components, a deficiency that may be very serious in situations where unary predicates (attribute values) may fail even to approximate a reasonable pattern description (e.g. chess board descriptions or cancer detection problems).

Acquisition procedures capable of learning new patterns in a spatial framework are well developed, but cannot overcome the endemic representational deficiencies noted above. Learning procedures operate by determining pattern volumes in a given N-dimensional space which are empirically effective bases for classification decisions. The learning algorithms which are used typically identify pattern volumes either by interposing linear hyperplanes between more or less overlapping proximate concentrations of exemplars of two distinct classes or by sweeping out equal hyperspheres of increasing radii from a centroid or otherwise central point among exemplars of a single class.

The third class of models, feature models, includes theories which maintain that a stimulus is represented by a list of its properties and their frequency of occurrence. A pattern prototype in memory is coded as a possibly much shorter list of criterial features and their necessary frequencies. For example, a capital "A" might be represented in memory as the list (2 oblique lines, 1 horizontal line);

all other features present or absent in a stimulus would then be considered irrelevant to this pattern. When a stimulus is to be matched to a set of possibly related ideal representations in memory, some arbitrary measure of the degree of similarity (fit) or dissimilarity (distance) is used. Thus, feature models are actually special cases of spatial models. Nevertheless, the emphasis of these models upon a necessary set of defining features eliminates one of the major problems of ordinary spatial models, their inability to reflect feature dependencies and irrelevancies specific to each pattern class. Feature models express the view that pattern exemplars belong to the same class because they share a common property, the class characteristic, which is expressed as a set of co-related feature values and frequencies. As a result, the dissimilarity between an exemplar and a pattern prototype is computed by a distance function specific to that particular class.

The most notable example of a feature model is Selfridge's Pandemonium (Selfridge, 1959; Selfridge & Neisser, 1960, 1963) which has been successfully used as a basis for character recognition. In its operation, it reflects all of the major strengths and weaknesses of feature models in general. It is capable of recognizing various patterns by distinct conjunctions of features specific to each class but is easily led to erroneous classifications by improper structural configurations of the criterial feature components. Another successful example of a feature model is found in

Uhr and Vossler's recognition program which has been used in a number of different problem environments (Uhr & Vossler, 1961, 1963; Uhr, Vossler, & Uleman, 1962). Their program differs from simpler versions of feature models both by allowing the generation of new features by extracting and "smoothing" small (5 x 5) subsets of retinal (20 x 20) grid patterns from visual exemplars and by evaluating the frequency of occurrence and fit of such local features at every possible location in the entire stimulus retinal display. Nevertheless, these apparent elaborations (spatial averaging) do not overcome the total dependency of such programs on unstructured feature sets as descriptions of complex patterns.

Attempts to operationalize bases for learning of features have been only mildly successful. The "template" feature extraction process of Uhr and Vossler is the best operationalized and tested (see also Uhr, 1973). In this process, the underlying notion is that exemplars of one pattern class share substructures or local features in common. For example, a letter "A" should have three different kinds of line intersections and approximate representations of these intersections may successfully be abstracted from pattern exemplars by extracting at random small subsets of their retinal displays (where each element in the complete 20 x 20 display has one of two values, lighted or unlighted). As in most computer oriented approaches to learning, hypothetical candidates for useful

local feature descriptions are first generated by some *ad hoc* heuristic rule and are subsequently evaluated for their usefulness. The success of this hypothesize-and-test technique is directly linked to the simplicity of the problem or, correspondingly, the probability that hypotheses generated by blind guessing will be satisfactory (Minsky, 1963; Newell, 1973).

In experimental and theoretical psychology where there is no equivalent emphasis on computing mechanisms, the absence of operationalized theories for feature learning is more evident (e.g. the development of distinctive features in Gibson, 1963 and 1969).

Aside from the problem of learning new features, much learning in feature models concerns the identification of *schemata*, sets of features, which are possible bases for classification in that they are identical to or closely approximate the criterial set of features constituting the characteristic of a pattern class. In fact, this problem is frequently posed in the form of deciding which features are not relevant to the classification task among an *a priori* set of possibilities. Here, most computer and statistical techniques employ independent correlational evaluations of each feature's reliability as an indicator of class membership and, typically, discard low performing features and compute a weighted sum of the remaining features for use in classification (Samuel, 1959; Selfridge & Neisser, 1963; Uhr & Vossler, 1963; Fu, 1968; Reed, 1973; Uhr, 1973). The

apparently preferable strategy of focussing (Bruner, Goodnow & Austin, 1956; Hunt, 1962; Kochen, 1961a, 1961b; Hunt, Marin & Stone, 1966), which proposes that schemata be identified by a parallel evaluation of feature values present in several exemplars, although frequently observed in human concept identification experiments, has proved difficult to implement in applied settings. Substantial progress on the development of suitable algorithms for this strategy has recently been made however (Michalski, 1973; Hayes-Roth, 1974b). These recent approaches are characterized by an attempt to evaluate directly the predictive reliability of sets of co-related features, as the feature model suggests one ought to do, rather than estimating it by some arbitrary averaging of the individual predictive reliabilities of the constituent features.

Finally, generative models propose a grammar for the intensional representation of patterns which is supposedly used to produce exemplar stimuli by a series of grammatical productions or derivations from an arbitrary starting representation. Each exemplar is associated with a grammatical structure which relates it to the starting representation by links expressing how it is grammatically derived. The theory supposes that because stimuli are produced by a series of systematic transformations on some basic unit (the starting representation), they may best be recognized by determining what those transformations were.

Notable examples of the generative approach include

syntactic pattern recognition (Uhr, 1971; Joshi, 1973; Fu, 1972; Thomason, 1973; Evans, 1969a, 1969b; Narasimhan, 1964, 1969; Franks & Bransford, 1971), analysis-by-synthesis (MacKay, 1956; Eden & Halle, 1961, 1962; Halle & Stevens, 1962; Neisser, 1967; Norman, 1969) and top-down language and speech understanding systems (Woods, 1970; Woods & Kaplan, 1971; Winograd, 1973; Walker, 1973; Paxton & Robinson, 1973). These systems employ syntactic and semantic rules to describe the kinds of stimuli that may be generated naturally and encountered in the environment. The perception of a stimulus pattern proceeds in two steps. First, the stimulus is synthesized or "forged"; the rules of the grammar are employed in some arbitrary manner until a particular sequence of transitions is found which leads to the production of an acceptably close replica of the stimulus. Second, when the transitions can be given an interpretation which is plausible in the problem domain, that interpretation is employed in some way to effect classification or perception.

One advantage of this type of approach is particularly noteworthy. Given a sufficiently powerful grammar, the generative technique is capable of computing all definable recognition functions (Minsky, 1967; Hopcroft & Ullman, 1969). Specifically, any structural dependencies or feature irrelevancies can be appropriately incorporated into the rules of grammar. On the other hand, the power of these approaches is intimately connected to their principal disadvantages. Among these is their usually enumerative,

recursive, or non-deterministic searches of the space of possibilities (all sentences or exemplars which can be produced by the grammar). As a result of such search procedures, these programs do not provide plausible models of the apparently simple and immediate recognition that is apparently involved in the perception of a well known pattern like the letter "A" (LaBerge & Samuels, 1974). Furthermore, the performance of these systems is seriously degraded by any expansions of the possible set of rules applicable at each step.

From one particular viewpoint, generative models appear related to natural mechanisms of perception in the same way that Turing machines are supposedly related to human cognition. In both cases the strength of the suggested analogy rests solely on the ability of the machine to effect input-output transformations of a complexity equivalent to that of the natural system. Beyond that, there is little to support the notion that the machine provides a believable or otherwise promising model of the natural computations of interest (Dreyfus, 1972; Lindsay, 1972).

Most general attempts at induction of grammars for useful behaviors (Friedberg, 1958; Friedberg, et al., 1959; Fogel, et al., 1966), the problem of ideogenesis, have been plagued by an inadequate comprehension of the relationship between the structure and function of a computational system (see, for discussions of these problems, Minsky, 1963;

Lindsay, 1968; Newell, 1973; Sussman, 1973). In the absence of an adequate basis of knowledge to allow the creation of general induction procedures for perceptual learning, most successful research projects have employed highly educated and skilled computer programmers to write clever programs that can perform effectively in the problem domain of interest. In some cases, these programs can learn by accepting both declarative and imperative information (Winograd, 1973), inducing structural descriptions from exemplars of limited sorts of pattern concepts (Sauvain & Uhr, 1969; Winston, 1970; Siklossy, 1971), and acquiring new skills by reconfiguring previously learned ones (Sussman, 1973). In all of these cases, however, the amount of knowledge acquired is virtually negligible in comparison to the amount incorporated into the performing and learning program, and extensions of these learning systems to new problem domains are always inconvenient and, in some cases, unfeasible. For example, extending Winston's pattern learning theory (1970) outside the world of toy block constructions would require completely new procedures; see, for a precise discussion of this point Banerji (1974).

Because perceptual systems based on generative models actually involve procedural representations of knowledge, the problem of learning to recognize structured patterns involves most of the questions that are central to the problem of learning how to behave generally. It is therefore desirable to proceed immediately to consider the more general

cognition problem, the development and use of effective rules of behavior.

2.3 The Cognition Problem: Rule Representation, Acquisition and Use in the Control of Behavior

The starting point for an analysis of the cognition problem is the simple and clear automata theoretic view of behavior (most points discussed here are treated thoroughly in Minsky, 1967 and Hopcroft & Ullman, 1969). An automaton is a machine which can be at any moment in exactly one of some number of possible states. The change over time of the state of the machine is governed by a simple rule, the transition function. Transition functions are usually expressed as mappings from the joint-space of the current state and the exogenously supplied current input into the space of possible next states. The power of automata theory rests on the ability of such simple devices to mimic any effectively described computational procedure (Turing, 1936; Davis, 1958; Minsky, 1967; Hopcroft & Ullman, 1969). Memory of preceding states and inputs is necessary for most computations and is provided in two ways. First, each distinct path of preceding states occupied by the machine may lead the machine to a unique new state. Second, the machine may be allowed to store information (symbols) in an external memory device like a tape or stack.

The central point of this automata theoretic viewpoint, it seems to me, is that behavior, the temporal succession of machine states and associated outputs or observable res-

ponses, is representable as a serial decision making process. At each point in time, the inputs to and the state of the machine are somehow distinctively characterized, and a decision function is evaluated to determine the next state, which subsequently becomes the current state. Such a view is utterly neutral with respect to practical problems. No stand is taken a priori on what are good ways to represent distinct state possibilities, to compute the decision function, or to extend the state set and transition function in adaptive response to new knowledge and feedback.

On the other hand, these issues are central to the problem of designing useful machines such as nervous systems and computers. These three functions--state representation, realization of the transition function, and acquisition and expansion of transition functions from training information --interact in numerous ways. Each distinct conceptualization of techniques to compute the decision function underlying the state-to-state transitions is realizable with widely varying degrees of difficulty in alternative state (memory) representation frameworks. In the remainder of this section, a variety of prominent alternative computational representations are considered in order to illuminate the ways in which these functions interact. The three main classes of devices considered in turn are finite automata, recursive function or stored program computers, and stimulus-driven computation machines.

The basic ideas in the finite automaton conception have already been mentioned: a homogeneous set of states which constitute memory, an input alphabet, the machine's occupancy of exactly one state at any moment, and a transition function. Non-deterministic automata (Hopcroft & Ullman, 1969) are formalisms which allow a mathematical description of machines which can occupy several states at one time, but do not explain how such a machine could be practically realized within the original state set-plus-transition function analytical framework. Finite automata are primarily useful for analyzing the complexity of languages, usually in terms of what is needed to produce or recognize arbitrary sequences of symbols. It is awkward to use these machines to describe multiply active interacting processes, because each one is the embodiment of a single independent computational function. Additional homogeneous states are introduced whenever it is found that more distinctions of past histories of input symbol sequences are required for some purpose. No easy method for identifying how this should be done, however, is known. On the other hand, within the framework of finite automata the determination of next state is extremely simple: someone exogenously classifies the input as one of N possible alphabetic symbols and an arc (a table entry) is followed to the next state.

Regardless of their apparent elegant simplicity--or, more precisely, because of it--finite automata suitable for

a specific task are extremely difficult to construct *in vitro* (in comparison, say, to functionally equivalent programs for digital computers). The reason, I suggest, is that the notions of memory of data (input sequences) and representation of logic are completely confounded between the state set and transition function. Because of this confounding, a structural analysis relating computational function to state connections appears virtually impossible. As a result, attempts to design mechanisms which can induce structural connections to perform useful functions are always based on extremely general strategies that are practically never successful (Friedberg, 1958; Friedberg, *et al.*, 1959; Minsky, 1963; Fogel, *et al.*, 1966; Lindsay, 1968) although demonstrably possible in theory (Suppes, 1969a, 1969b; Arbib, 1969).

Both from these failures at mechanized acquisition of computational knowledge and from theoretical analyses of the capabilities of various computing devices, an appreciation of the relationship between structure of memory and function has emerged. Specifically, a variety of functional decompositions of the homogeneous state set and additional memory devices have been found necessary or desirable in certain problems. For example, the recognition of context free languages is most easily achieved by a pushdown automaton whose memory comprises, in addition to a finite set of homogeneous states, an infinite bank (stack) of positionally ordered storage registers with which information

is stored, shifted, and retrieved to remember yet unencountered components of previously introduced phrases. More powerful storage systems, like the infinitely long extended tape memory used by Turing machines are necessary to compute more complex functions.

It seems generally accepted, however, that even these elaborations of the earlier undifferentiated concept of state are still inadequate for the practical problems of representing, computing, and easily extending complex cognitive functions. By far the greatest successes in cognitive simulation have been achieved using recursive functions (Kleene, 1936, 1952; McCarthy, 1960; McCarthy, et al., 1962; Weissman, 1967) or their equivalents, subroutines, which are organized into complex hierarchical computer programs (Miller, et al., 1960; Woods & Kaplan, 1971; Winograd, 1972). As ordinarily implemented, both subroutines and recursive functions exploit similar aspects of memory and computational control of modern digital computers.

The capacity for communication which is critical to the organization of computation in hierarchical systems is provided by the ability to bind or instantiate parameters or variables, which are referenced symbolically within a subordinate (called) subroutine or function, to constants or values designated by the superior (calling) subroutine or function program. Control resides in the one program which is currently being executed and may be transferred from superior to subordinate routines when the computation

of a previously differentiated function is required. Control is temporarily passed to the recursive function or program subroutine, which accepts a set of values as arguments for its symbolic parameters and executes a set of operations one-at-a-time, and is then regained by the calling program which receives result values for some of its parameters from the subroutine in a similar manner. Ordinarily, the arguments and result values are transported from the locus of one occasion of their use to another by placing them in a particular storage bin and supplying the addresses of the corresponding bins to all functions which employ them in computation. Loosely speaking, the structure of the programs may remain fixed indefinitely while the contents of memory bins are appropriately altered.

Such a framework, when the possible operations include store, retrieve, increment, decrement, compare-and-jump, and jump or their equivalents is a universal computation environment. More important, however, such a representation scheme has proved extremely powerful in the ad hoc simulation of important cognitive functions, because it is relatively easy to use when a problem and solution are well defined. This is a result of three outstanding features of modern computer programming languages: (1) they allow a precise separation of data and logic; (2) they permit hierarchical construction and control of procedures, because subfunctions can be represented in isolation and

invoked *in toto* with proper argument values whenever needed; and (3) the capacity exists to operate upon stored programs themselves like data, thereby allowing a program to expand or modify itself in systematic ways.

With respect to the problem of ideogenesis and adaptation, few successes have been had with general computer programs in spite of the capacities just cited. One possible approach to this problem is to connect functions, more or less at random, in the hope of discovering an ordered set which accomplishes a desired result. Both because of the similarity of this approach to the unstructured approaches used with finite automata and because general computer programs are so much more complex than finite automata, this approach seems very undesirable. A second approach, related to the first in that it relies on existing subfunctions to construct new ones, differs from the first in its effort to prove symbolically that a particular set of functions perform for all inputs X a desired input-output transformation of X into Y such that some proposition P(X,Y) is always true (Simon & Kotovsky, 1963; Hewitt, 1972; Kotovsky & Simon, 1973; Sussman, 1973; Rulifson, *et al.*, 1972; Balzer, 1973; Boyer & Moore, 1973; Darlington, 1973; Katz & Zohar, 1973). The successes in this approach have been very limited primarily, it seems, because of the extreme difficulty of generating an appropriate problem statement--defining exactly what P(X,Y) is--and proving its validity in any but very well understood contexts. In very

limited problems where the most a priori heuristic knowledge is provided (Rulifson, et al., 1972; Sussman, 1973), performance is best. Even in these cases, however, the extensibility of the previously successful strategies seems extremely doubtful unless a potential learning problem is guaranteed to be severely constrained in complexity by large amount of pre-analysis by the user of the automatic programming (learning) machine. In short, no general, effective, and realistic mechanisms for program induction have been identified and the overall trend continues to turn away from the general learning problems toward limited, knowledge-intensive, and specific ones (Rulifson, et al., 1972; Sussman, 1973; Winston, 1970).

The last group of computing systems to be considered is that of stimulus-driven computation (SDC) machines. These machines represent a return to the automata-inspired notion that a state and input lead directly to the next state, but incorporate the elaborations of state and computational power of the recursive function machines. Specifically, an SDC machine is defined by a possibly unlimited set of memory storage bins, a set of rules describing possible data structures, and a set of rules describing how the contents of bins are to be altered whenever the current contents satisfy some predefined condition. No control effort is expended in SDC systems on memory of where one is within the current program in contradistinction to both automata (finite and infinite) and programmed computers. Rather,

the contents of memory and the currently available set of memory altering rules jointly and directly determine what operations are to be performed at each instant.

The best known examples of computation universal SDC systems are grammatical production systems, especially transformational grammars (Chomsky, 1957, 1965; Langendoen, 1969; Friedman, 1971), type zero formal grammars (Hopcroft & Ullman, 1969), Post productions (Minsky, 1967), and web grammars (Pflatz & Rosenfield, 1969). These systems typically are used to study the production of sentences, theorems, or graphs by a set of independent rules which are applied in an arbitrary order to the previously produced sentence, theorem, or graph until a certain condition is encountered. Each rule is a two-tuple, (contingency, response), which is applicable to the current sentence only if it matches or satisfies the conditions represented by the contingency. One of the rules applicable at each instant is arbitrarily chosen and invoked; that is, the sentence or graph is transformed to satisfy the conditions specified by the response component of the rule. When needed, such rules will be explicitly referred to as contingency-response transformations; otherwise, the term rules will suffice.

In the use of production systems, where the set of transformations is fixed, the state of the machine is defined by the current sentence or graph being manipulated. If the transformation set could expand itself, the state of the machine would necessarily additionally include an account of

the current set of transformations. As practically implemented, these systems are effected by a serial attempt to match the current sentence to the contingency of each transformation and invocation of the first one which is satisfied (Fikes & Nilsson, 1971; Rulifson, et al., 1972; Newell, 1972).

Two major difficulties arise with the use of these representations for complex cognitive behaviors. First, it is difficult to conceive of general cognitive activity in the context of a single sentence which contains all relevant structured information. Thus, several sentences or memory bins are examined and manipulated simultaneously (Newell & Simon, 1972; Newell, 1972). Second, the logical complexity of the contingencies for action often exceeds the description and matching capabilities of the available systems. As a result, ad hoc representations are used for each sort of problem and complex contingencies are realized either by approximation with a series of transformations each of greatly reduced contingency complexity or by a non-deterministic matching algorithm (see, for example, Rulifson, et al., 1972).

Reports of attempts to learn transformations in vitro are practically non-existent (Hayes-Roth, 1973a), despite their being the supposed basis of nearly all natural verbal behavior (Skinner, 1957; Chomsky, 1965, 1968, 1969; Lenneberg, 1967). Holland (1974) proposes a broadcast automaton in which the transformations are between simple symbol strings which are conceived as radio messages stimulating all receivers

whose contingencies (substrings) they match. He suggests that learning can be performed in this context by applying genetic operators such as crossover and inversion on populations of these strings which reproduce in proportion to their usefulness, but is unable to specify exactly what a good relationship between structure and function would be. In a similar vein, workers in the field of cell assembly theory have suggested that simple synaptic change rules might account for all transformational rule learning in both natural and artificial nervous networks. Despite provocative successes in limited areas, they have been unable to demonstrate these strong claims either empirically or mathematically at the level of the nervous net or at a more abstract functional level (Hebb, 1949; Rochester, et al., 1956; Milner, 1957; Good, 1965; Finley, 1967; Plum, 1972; Cavanagh, 1972). Thus, the potential for ideogenesis and adaptation in the context of SDC seems at this point like that previously noted in respect to other representational frameworks: a general capability is known to exist in theory or is strongly suspected, but an effective, structural approach is yet to be discovered. In the next chapters, one such approach is presented, the effectiveness of which is proved in Appendix IV.

Chapter 3: THE STRUCTURAL REPRESENTATION (SR) STIMULUS-DRIVEN COMPUTATION SYSTEM

3.1 Chapter Overview

Structured events are configurations of objects in various relations with one another. A parameterized structural representation for such events is described. Subsequently, the role of such representations in the description of stimuli and pattern ideals and in pattern recognition is explained. Contingency-response transformations between structured events are the basis of computation in the proposed structural representation (SR) system. Transformations are conceived as independent, deterministic, distributed, and associative operators that are computed in parallel, synchronously or asynchronously. Essential principles of organization in the use of the proposed system as a representational basis for modeling cognitive and perceptual functions are discussed. Important practical and theoretical questions of memory control not considered in this paper are acknowledged. The relation of the SR system to previous work is also discussed; that the SR system is computation universal is proved in App. IV.

3.2 Parameterized Structural Representations (PSRs) of Events

A structured event is a configuration of abstract or concrete objects in logical, physical, spatial, temporal, or activity relations. In brief, an event is a set of co-related or conjunctive relations. A relation is defined as

a set comprising a predicate (proposition, property, attribute, assertion) and any number of associated predicate objects whose relationships to the predicate and other objects may be distinctive or identical (symmetric).

A minimum amount of notation is now introduced to permit the exposition of the properties of the SR system. A relation is represented as a set {predicate term, object $term_1$, object $term_2$, ..., object $term_n$}, with the following conventions. The predicate term is of the form p:predicate value, where predicate value is replaced by any arbitrary predicate. Each object $term_i$ ($i = 1, 2, \ldots, n; n > 0$) is a structure similar to that of the predicate term. It is any structure of the sort object type:object value, where object type is replaced by a symbol which designates the role played in the relation by the actual object occurring in place of the corresponding object value. These simple notions are clarified by the following illustrative relations representing the assertions "Bill is handsome" and "Bill adores Mary," respectively:

{p:handsome, who:Bill} (3.1)

{p:adores, agent:Bill, object:Mary}. (3.2)

A small number of terms prove useful in discussing such relations and are now introduced. The order of a predicate (or a relation which contains it) is the minimum number of object elements which may occur in any relation of that predicate type. The names unary, binary, etc. are used to denote relations or predicates of orders one, two,

..., respectively. For example, the unary relation (3.1) assigns the property "handsome" to the object "Bill" while the binary relation (3.2) asserts the the "agent" "Bill" "adores" the "object" "Mary."

The order of a predicate and its semantic interpretation are not necessarily self-evident. These are questions which are under the control of the user of the SR system. The SR system itself is a content-free schema for the structural representation of information and computation, and this paper is almost exclusively dedicated to questions concerning the structure rather than the substance of knowledge and procedures for its acquisition. Thus, the actual predicates and object types used here are all arbitrary. For example, the fact that "Mary, Jane and Paula are all the same" might be expressed in the following compact way using the binary symmetric predicate "same":

{p:same, who:Mary, who:Jane, who:Paula}. (3.3)

Alternatively, (3.3) might be represented as an equivalent set of three primitive relations, as follows:

{{p:same, who:Mary, who:Jane}, {p:same, who:Mary, who:Paula}, {p:same, who:Jane, who:Paula}}. (3.4)

The interpretation of a compact relation like (3.3) is always that it is a shorthand representation of the corresponding expanded set of primitive relations like (3.4) in which the number of object terms is exactly equal to the order of the predicate and there is one object of each required type.

A parameterized structural representation (PSR) of an event is, essentially, a description of that event using a set of relations in which various arbitrarily chosen symbols (parameters) are introduced to refer to objects in a consistent manner in one or more relations. Specifically, a PSR is a relation with the following components: a predicate "event"; one object of type "parms" identifying the set of symbols used as parameters in the event description; one object of type "body" defining the set of relations which describe the event; and one object of type "stat" referring to statistics which may be used to specify attributes of the PSR itself, such as the memory structure it resides in, how active it is, when it will decay, and so forth. For the current objectives, the "stat" object is unnecessary and will be ignored, but it is considered later in conjunction with memory control processes (section 3.5).

Three examples of structured events are given in figures 2, 3, and 4. PSRs called A, B, and C, respectively, describing these events are given below. Following each PSR is a brief corresponding verbal interpretation to clarify its exact meaning.

```
A={p:event, parms:{a1,a2,a3,a4,a5,a6,a7,a8,w,x,y,z},
   body:{{p:incidence,node:a1,node:a2,node:a3,line:w},
         {p:incidence,node:a3,node:a4,node:a5,line:y},
         {p:incidence,node:a7,node:a5,line:z},
         {p:incidence,node:a2,node:a4,node:a6,node:a8,
```

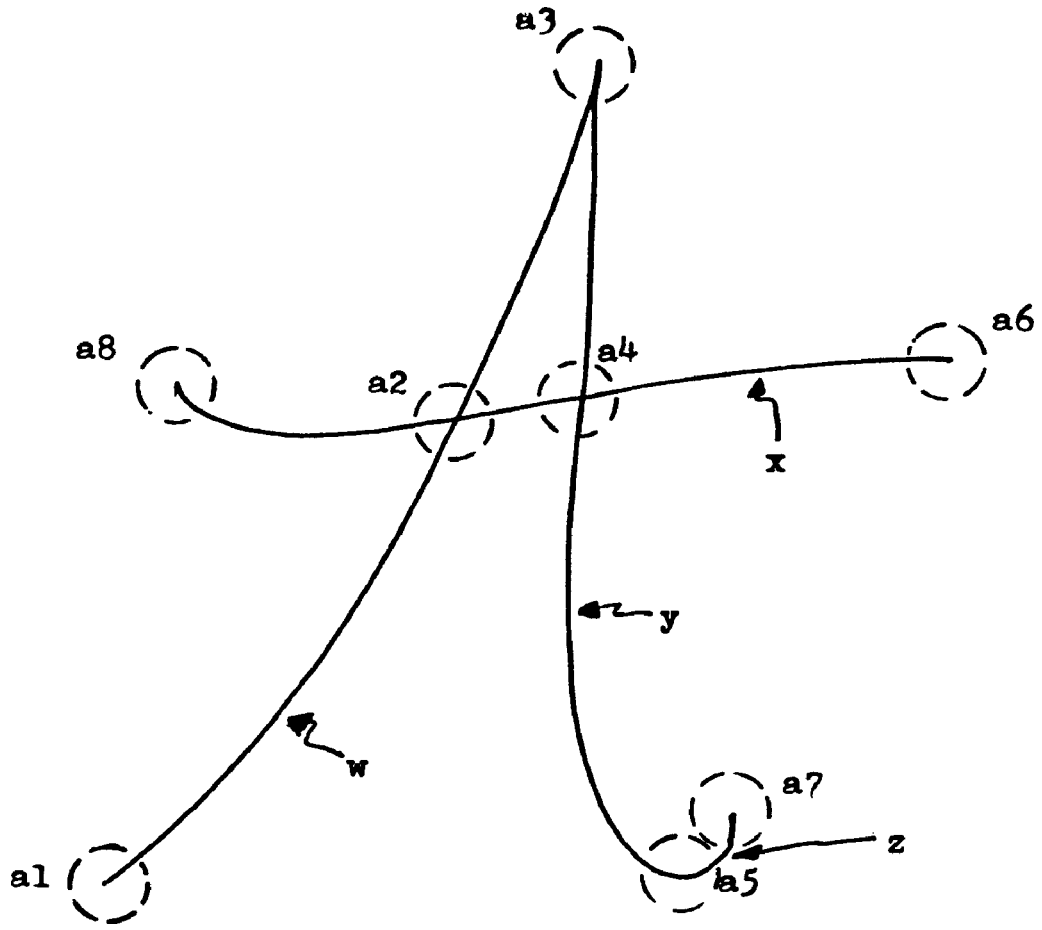


Figure 2. A structured event: a script "A" and the use of parameters to identify nodes of the line drawing (points of intersection or extreme discontinuity) and apparently continuous strokes (w, x, y, z).

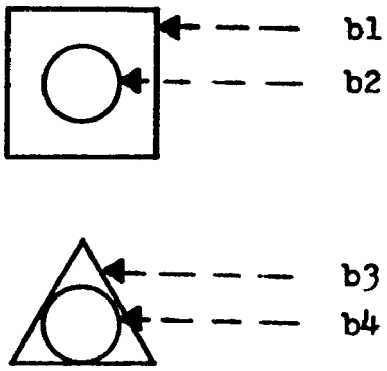


Figure 3. A structured event: an exemplar of a relational concept and the use of parameters to identify geometric forms as objects.

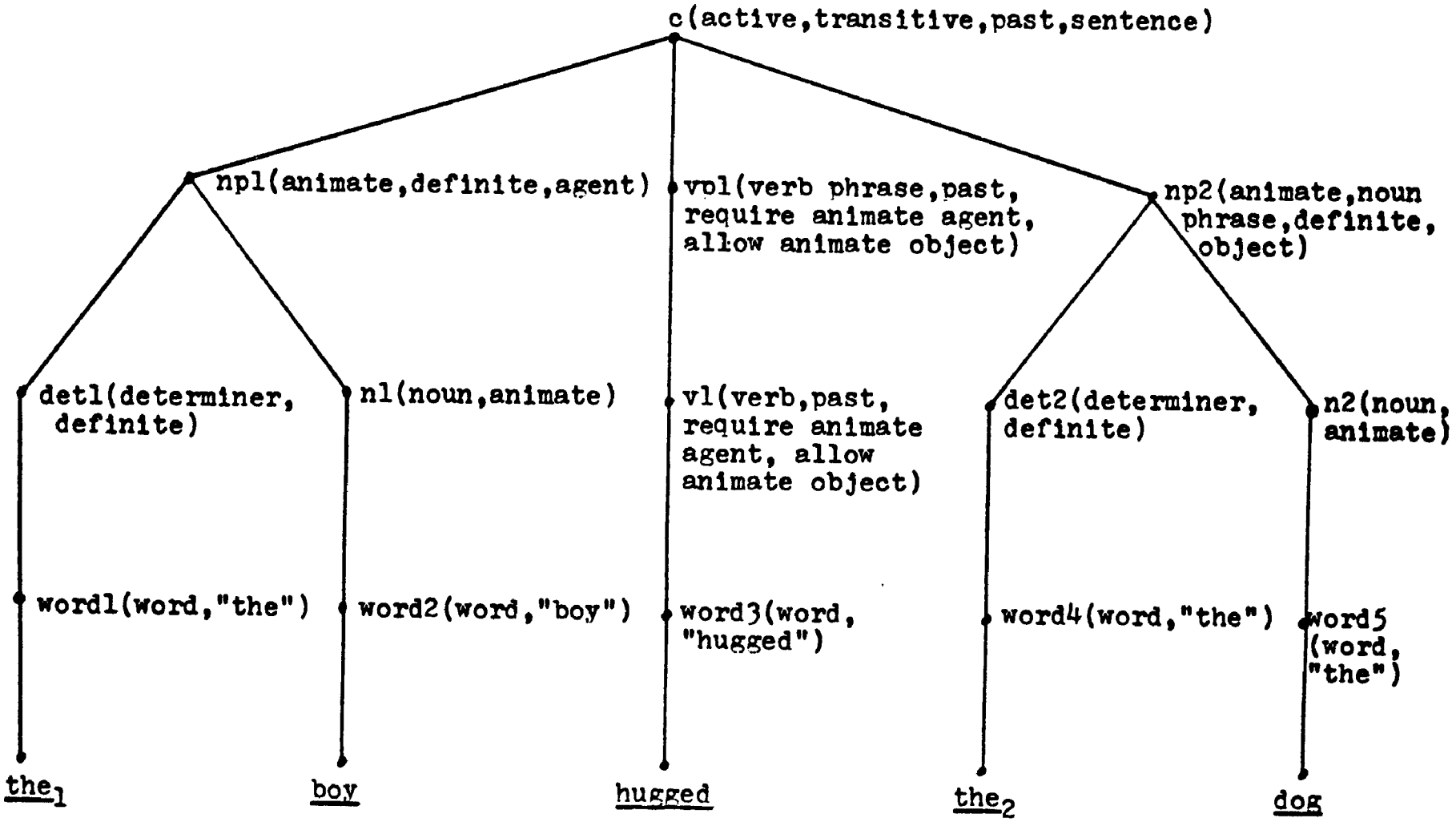


Figure 4. A structured event: a phrase structure tree for the verbal stimulus $\underline{\text{the}}_1$ $\underline{\text{boy}}$ $\underline{\text{hugged}}$ $\underline{\text{the}}_2$ $\underline{\text{dog}}$ meaning "the boy hugged the dog." The tree includes syntactic and semantic markers (e.g. animate, transitive, require animate objects) and parameters (e.g. word1, det1, c) to name nodes.

line:x}]}. (3.5)

The event A is a line drawing with eight nodes (loci of termini of lines, intersections of lines, or extreme curvature). In this event, a seemingly continuous line (w) connects nodes a1, a2, and a3; one (y) connects nodes a3, a4 and a5; one (z) connects a5, a7; one (x) connects a2, a4, a6 and a8. Incidence is a binary relation of a node and a line. Relations with more than one node object represent conjunctions of all distinct such binary relations. Each of these relations will share a common line object, naming the smooth curve on which each of the nodes occurs.

B = {p:event, parms:{b1, b2, b3, b4},
body:{{p:geometric form, type:square, name:b1},
{p:geometric form, type:circle, name:b2,
name:b4},
{p:geometric form, type:triangle, name:
b3},
{p:contain, outer:b1, inner:b2},
{p:contain, outer:b3, inner:b4},
{p:above, upper:b1, upper:b2, lower:b3,
lower:b4},
{p:same shape, name:b2, name:b4}}}. (3.6)

The event B contains four geometric forms named b1, b2, b3, and b4 in two vertically configured nested pairs such that the uppermost square and the lowermost triangle each contain forms which have the same shape and are circles.

C = {p:event,

parms:{c, np1, np2, v1, vp1, det1, det2, n1, n2,
word1, word2, word3, word4, word5},

body:{{p:word, which:"the", subordinate:$\underline{the_1}$,
name:word1},

{p:word, which:"boy", subordinate:boy,
name:word2},

{p:word, which:"hugged", subordinate:
hugged, name:word3},

{p:word, which:"the", subordinate:$\underline{the_2}$,
name:word4},

{p:word, which:"dog", subordinate:dog,
name:word5},

{p:determiner, definite:yes, subordinate:
word1, name:det1},

{p:noun, animate:yes, subordinate:word2,
name:n1},

{p:verb, require animate agent:yes, allow
animate object:yes, subordinate:word3,
name:v1},

{p:determiner, definite:yes, subordinate:
word4, name:det2},

{p:noun, animate:yes, subordinate:word5,
name:n2},

{p:noun phrase, animate:yes, definite:yes,
determiner:det1, noun:n1, name:np1},

{p:verb phrase, tense:past, require animate

subject:yes, allow animate object:yes,
verb:v1, name:vp1},
{p:noun phrase, animate:yes, definite:yes,
determiner:det2, noun:n2, name:np2},
{p:sequence, 1:start, 2;np1, 3:null, 4:
vp1, 5:null, 6:np2, 7:end},
{p:sentence, active:yes, transitive:yes,
tense:past, agent:np1, verb phrase:vp1,
object:np2, name:c}]}. (3.7)

This PSR is interpreted as follows. The event C is the grammatical description of an active, transitive, past tense sentence (c) composed of the agent noun phrase np1, the verb phrase vp1, and the object noun phrase np2. np1 is a noun phrase with the definite determiner word1 ("the") and the animate noun n1 ("boy"), etc. There is one new feature of the SR system illustrated in (3.7) in the relation with predicate "sequence" which needs explanation. Frequently in natural problem contexts, spatial or temporal ordering is important. To simplify the representation of such orders, numeric object types (1, 2, 3, ...) may be used with the following interpretation. Each relation containing numeric object types is interpreted as the set of primitive binary relations with only two object types, 1 and 2, expressing all possible consistent pairwise orderings. The relation in the preceding example represents the fact that the "start" of the sentence directly preceded np1 which directly preceded vp1 ("null" means no intervening words) which

directly preceded np2 which directly preceded the "end" of the sentence.

Several general points about PSRs should be noted. First, every relation may be named to allow it to be referenced as an object by other relations. In these examples this was always done by introducing an object term with object type "name" followed by the parameter symbol chosen to name it. Naming need not be done in exactly this way, but this is a convention followed in this paper. Similarly, any set, say M, which is referenced *in toto* as an object in some relation may be constructed by introducing the additional object term, "in:M," in every relation which is contained by the set M. For example, the term in:C can be embedded in every relation in (3.7) to permit the sentence structure defined by (3.7) to be referred to as one object simply by citing the parameter C. This technique is a convention followed in the current paper and is desirable because it provides for complete representational uniformity in the SR system: every element of memory is a single predicate relation. The choice of parameter symbols, of course, is arbitrary, except that each distinct object is referred to by exactly one unique parameter. PSRs are equivalent to expressions in lambda notation (Church, 1941; Newell, 1963; McCarthy, *et al.*, 1962), Appendix IV relates PSRs to conjunctive forms of the predicate calculus.

3.3 Representation of Ideals (Patterns, Concepts, Prototypes)

The desirability of PSRs as representational bases for knowledge rests to no small degree on their suitability for describing abstract or concrete pattern ideals. Where the previous approaches to representation (chapter 2) have relied on structural simplification or programmatic intensional descriptions (e.g. generative grammars), descriptions in the SR system are non-computational and completely general. For example, the events of figures 2, 3, and 4 may be supposed to have been specific exemplars of particular patterns the recognition of which would necessarily require some representation of the corresponding ideals. In this section, such ideal representations are presented.

Figure 5 represents a Gothic capital "A" which may be supposed to be the graphic template for all such capital letters, including that shown in figure 2. A representation of such an "A" is given here in terms of the PSR called T(A).

T(A)={p:event, parms:{A1,A2,A3,A4,A5,L1,L2,L3}.
body:{{p:incidence,node:A1,node:A2,node:A3,line:L1},
{p:incidence,node:A3,node:A4,node:A4,line:L3},
{p:incidence,node:A2,node:A4,line:L2}}}. (3.8)

Note that any stimulus whose representation reflects all of the relations present in T(A) would necessarily contain as a subpattern the pattern in figure 5. For example, the pattern in figure 2 described by the PSR A (3.5) can easily be seen to contain all relations in (3.8) if the cor-

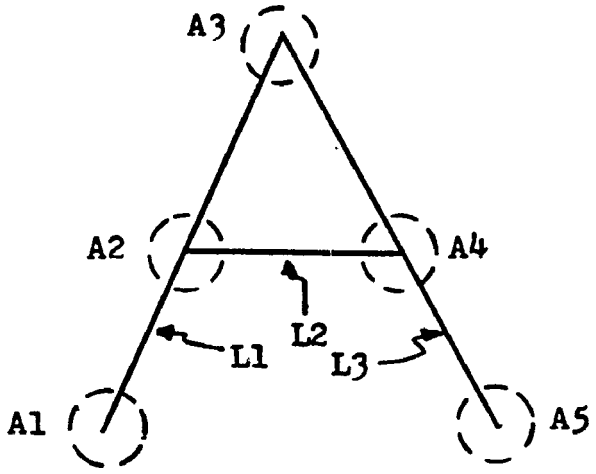


Figure 5. A Gothic capital "A" and eight parameters used to construct a structural template, T(A) (see text).

respondences a1=A1,a2=A2,a3=A3,a4=A4,a5=A5,w=L1,x=L2,y=L3 are made. Thus, T(A) is a structural template for recognition of examples of such forms of capital "A"'s. The recognition problem, deciding when a stimulus does in fact manifest all of the relations present in the PSR template, is analyzed and solved in the subsequent chapters. For the present, the important point to note is that the PSR T(A) is a structural template for capital "A"'s which is (intentionally) insensitive to deviations of size, orientation, font, etc.

The underlying concept of which the geometric form configuration of figure 3 is actually an example is shown in figure 6. The precise concept to be represented is the following: "the pattern consists of two pairs of nested geometric forms in a vertical configuration such that the interior figures of both pairs are the same shape." A PSR representing this idea, a structural template for the recognition of occurrences of this concept, is T(B):

```
T(B) = {p:event, parms:{B1, B2, B3, B4},
        body:{{p:geometric form, name:B1, name:b2,
               name:B3, name:B4},
              {p:contain, outer:B1, inner:B2},
              {p:contain, outer:B3, inner:B4},
              {p:above, upper:B1, upper:B2, lower:B3,
               lower:B4},
              {p:same shape, name:B2, name:B4}}}.   (3.9)
```

It is clear that the previously described geometric

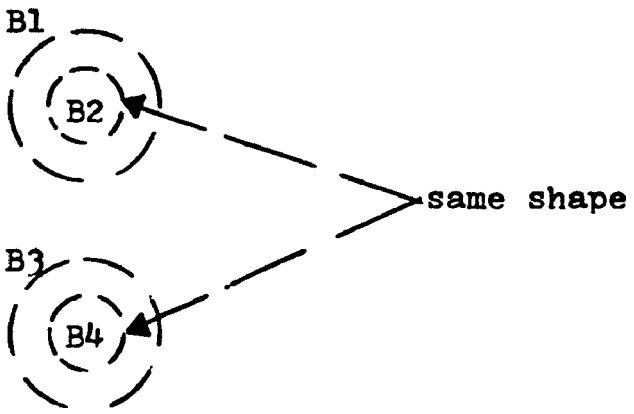


Figure 6. An abstract concept for a class of patterns and the use of parameters to identify objects in a structural template, T(B) (see text).

configuration from figure 3 is an example of one pattern that satisfies the conditions expressed by the body of T(B). If the correspondences b1=B1, b2=B2, b3=B3, and b4=B4 are made, every relation in T(B) is also present in B (3.6). Thus B matches T(B) and is an exemplar of the pattern whose ideal is T(B).

The last example comes from a stated task requiring the recognition of all active, transitive, past tense sentences with one object for the purpose of converting them to equivalent passive forms. For this purpose, a template of the form shown in figure 7 is needed. The corresponding PSR, T(C), is given below:

```
T(C) = {p:event,
        parms:{ct, nplt, vplt, np2t},
        body:{{p:noun phrase, name:nplt},
              {p:verb phrase, tense:past, name:vplt},
              {p:noun phrase, name:np2t},
              {p:sequence, 1:start, 2:nplt, 3:null,
               4:vplt, 5:null, 6:np2t, 7:end},
              {p:sentence, name:ct, agent:nplt, verb
               phrase:vplt, object:np2t, tense:past,
               active:yes}}}.                          (3.10)
```

As in the preceding examples, the ability of T(C) to serve as a template for matching events is illustrated by noting that every relation in (3.10) is also present in (3.6) when the parameters npl, np2, vpl and c in C are replaced by their correspondents in T(C), nplt, np2t, vplt

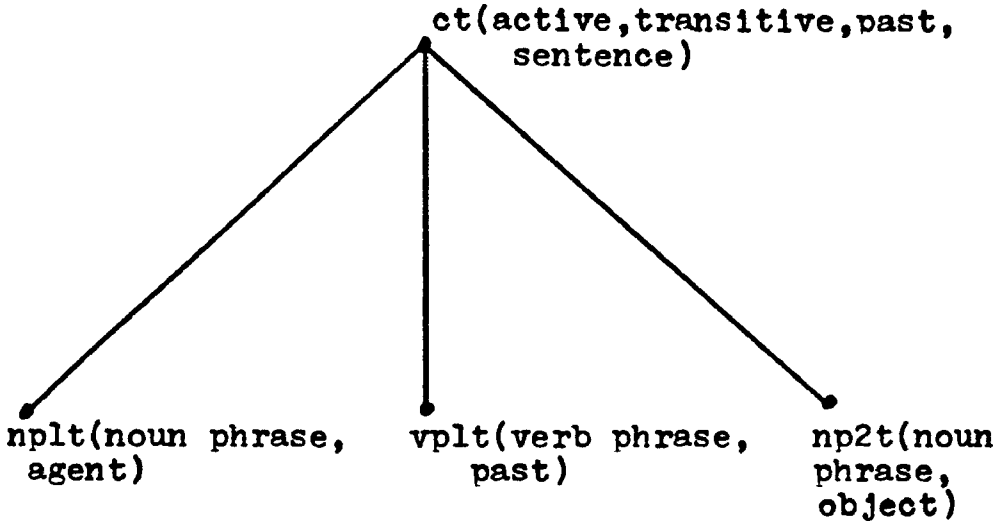


Figure 7. A phrase tree fragment used as the structural template for a rule of transformational grammar.

and ct, respectively. In later chapters the abstraction of such templates from experiential training information is discussed and the use of such templates in recognition and transformational grammar is explained.

At this point it is desirable to credit others whose approaches to pattern description have been similar to that reflected in the proposed PSRs of the SR system. Barrow, Ambler, and Burstall (1972) considered directed, connected graphs as suitable bases for descriptions of visual stimuli and templates, in a manner quite similar to that discussed here. Their work and my own developed independently. In any case, their analysis lacks the generality which has led to the identification of the parameter set and n-ary relations as the preferred representational basis. The difficulties in their proposed methods are common to most connected graph approaches to pattern representation: by restricting relations to binary connections (arcs) among nodes (parameters), higher-order relationships are difficult to represent correctly and, if represented in the manner usually suggested, are simply and incorrectly translated into sets of supposedly equivalent binary relations. For example, the relation {p:give, agent:Mary, recipient:Jane, object:ball} would in the typical manner be converted into the set {{p:give, agent:Mary, recipient:Jane}, {p:give, agent:Mary, object:ball}, {p:give, recipient:Jane, object:ball}}. Although this set is implied by the original relation, the converse

is not generally true (unless no interacting relations with similar predicate and object types are also present). In order to retain all content in an n-ary relation, it is further necessary to indicate which binary relations belong together. Such weaknesses are overcome in the framework of binary relations either by a hierarchical representation in which essentially every subset of objects is successively constructed by pairing two lower-order subsets (see, for examples which lack parameters, Anderson & Bower, 1973) or by some additional relations connecting binary relations arising from the same original n-ary relation (see the discussion of uniform PSRs in section 4.7).

Several other researchers have proposed graphical representations like those just discussed (Clowes, 1969, 1971; Narasimhan, 1964; Winston, 1970; Shaw, 1970, 1972). In general, the capabilities of other proposed representations are subsumed by the PSRs described in this chapter. On the other hand, PSRs are capable of representing all sorts of structured events in a precise and easily interpreted manner, including those outside the field of graph-representable visual patterns and, therefore, seem to be a desirable generalization of these previous research efforts (see App. IV for advantages of PSRs over predicate calculus and graphic representations).

3.4 Representing Behavior Rules: Contingency-Response Transformations (TXs)

The form of contingency-response transformation (TX)

used to represent computing rules in the SR system is that of a simple relation, {p:TX, parms:parameter set, C:contingency body, R:response body, stat:statistics}, where the contingency body is a set of relations describing an event whose recognized occurrence will cause those actions described by the set of relations in the response body of the transformation to be effected. The statistics represent optional information about the strength, fatigue, activity state, etc. of the transformation which may be important for simulating natural computing systems. On the other hand, these unit statistics may be completely omitted from consideration, which is the course adopted here. (For a discussion of some uses of unit statistics, see Hayes-Roth, 1973a and section 3.5 of the current paper.)

Some examples of transformations should indicate their essential properties. The three ideals, T(A), T(B), and T(C) in (3.8-10), of figures 5-7 will be used as contingencies for three behaviors of obvious relevance to perception and cognition. These will include labeling examples of the letter "A" as "letters, which are A's," categorically perceiving exemplars of the pattern concept in figure 6 and categorizing them with "yes" responses, and transforming active past tense sentences matching the template in figure 7 into equivalent passive forms. The transformations needed to do these things are expressed simply in terms of the bodies of T(A), T(B), and T(C) which

will be denoted body(T(A)), body(T(B)), and body(T(C)), respectively. Specifically, the transformations label(A), categorize(B), and transform(C) in (3.11-13) perform the desired labeling, categorization, and transformation functions:

```
label(A) = {p:TX, parms:{A1, A2, A3, A4, A5},
     C:body(T(A)),
     R:{{p:letter, which:"A", name:$(A1, A2, A3,
         A4, A5)}}}                                    (3.11)
categorize(B) = {p:TX, parms:{B1, B2, B3, B4},
     C:body(T(B)),
     R:{{p:speak, what:"yes"}}}                        (3.12)
transform(C) = {p:TX,
     parms:{ct, nplt, np2t, vplt, vlt, passive-ct,
          word3t, v, w, x, y, z},
     C:body(C(T)) ∪ {{p:verb phrase, name:vplt,
          verb:vlt},
          {p:verb, name:vlt, word:word3t},
          {p:word, which:x, name:word3t},
          {p:past participle, past tense:x,
           participle:z}},
     R:{{p:sentence, agent:nplt, tense:past,
         transitive:yes, passive:yes, object:np2t,
         verb phrase:y, name:passive-ct},
        {p:verb phrase, name:y, verb:w, tense:past,
         aux:"was"},
        {p:verb, name:w, word:v},
```

{p:word, which:z, name:v},
{p:sequence, 1:start, 2:np2t, 3:null, 4:y,
5:null, 6:"by", 7:null, 8:nplt, 9:end}]}.

(3.13)

These transformations are easily interpreted. The contingency values are parameterized patterns which are compared against the current patterns in memory to determine if any set of correspondences between the parameters of a memory pattern and the contingency pattern exist which entail the conclusion that the two patterns match. If so, the corresponding memory pattern parameters are substituted into the response pattern issued by the transformation. (Alternatively, the variables in the response pattern are instantiated and bound to the corresponding arguments in the matching stimulus pattern.) Thus, the presence of an "A" in memory will match label(A)'s contingency pattern and cause a new relation with predicate "letter" to be stored in memory. The name of that relation, $(A1, A2, A3, A4, A5), represents a newly generated unique symbol which is determined as a function of the stimulus pattern parameter names corresponding to template parameters A1-A5. This transformational operation is functionally identical to the pattern directed matches and instantiations of PLANNER (Hewitt, 1972), QA4 (Rulifson, et al., 1972) and QLISP (Reboh & Sacerdoti, 1973). Here however, pattern matching is presumed to be automatic (bottom-up, immediate) and associative, and the instantiation of response

patterns is the basis of all computation behavior (this is elaborated in sections 3.5 and 3.6).

The contingency of categorize(B) is the previously described body of T(B). If a stimulus is present whose internal representation matches the concept ideal of figure 6, a categorizing "yes" message is spoken.

Finally, the contingency of transform(C) is matched by any structure which represents a sentence of the sort in figure 7 if, in addition, the verb word (x) is in past tense form and the corresponding past participle (z) is also presently stored in memory. If this is so, a response is issued whose effect is to: reverse the order of the noun phrase constituents; insert in the new verb phrase (y) the auxiliary "was" before the verb (w) component containing the past participle (z) which replaces the original verb word (x); and insert the passive word "by" before the agent nplt. This rule is illustrated in figure 8. If the sentence "the boy hugged the dog" (figure 4 and (3.7)) were in memory and matched T(C) causing transform(C) to be invoked, the resulting sentence would have the surface structure "the dog was hugged by the boy."

3.5 Principles of Computing in the SR System

The PSRs and TXs described earlier in this chapter are the essential components of any computational system modeled in the SR framework. They are of theoretical interest because of their suitability as a basis for the analysis of

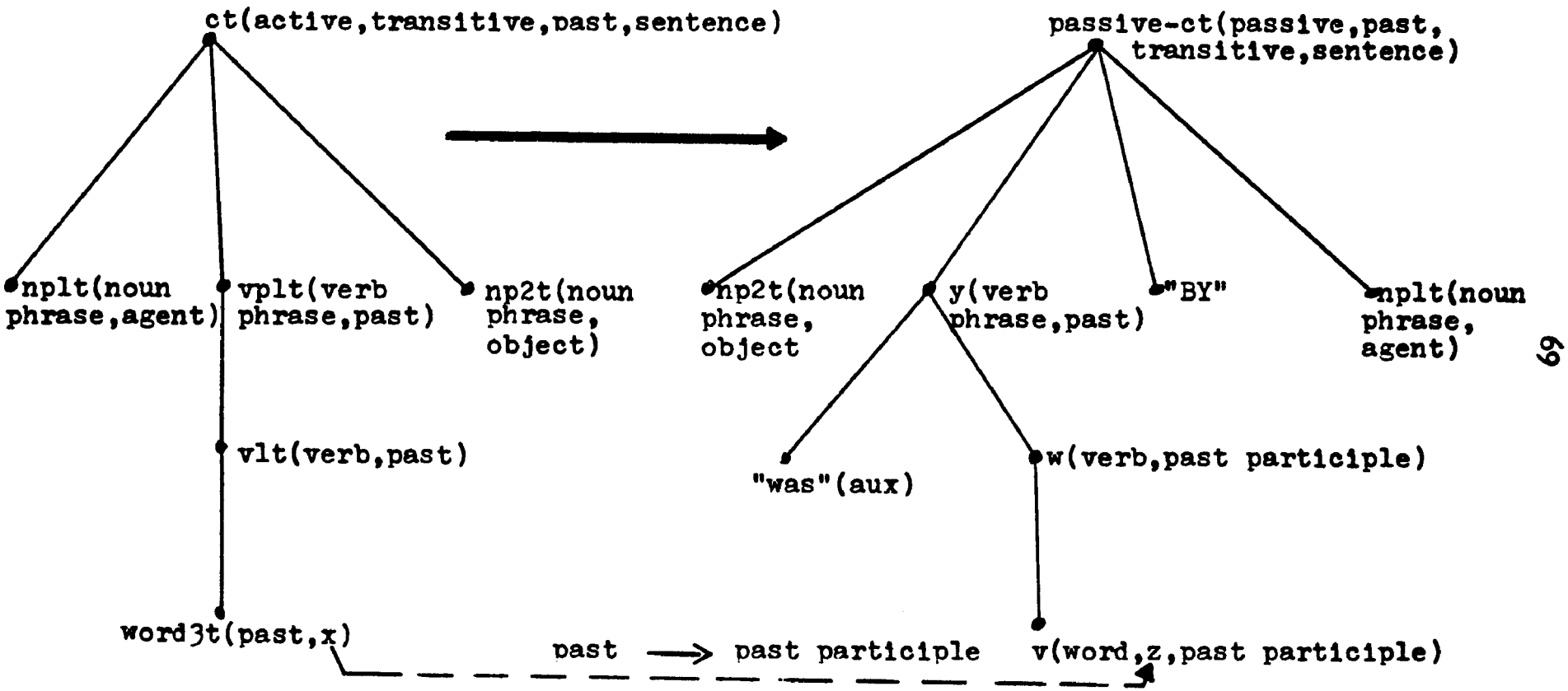


Figure 8. A diagram of the transformation, transform(C), which is matched by any active sentence of the sort illustrated by ct and which produces a corresponding passive form, passive-ct.

the representation, organization, acquisition and use of structured knowledge. They are important not because they constitute an immediately practical basis for the implementation of large scale computer systems but, rather, because they represent a formal theoretical alternative to current computational models of advanced information processing systems. Much will be said about practical issues later in this paper. The objective of this section, however, is to relate the proposed SR framework to notions of organization and function typically held to be important in the analysis of complex perceptual and cognitive behavior. The material is not required for an appreciation of the subsequent results and may be skipped if desired.

The specific issues to be considered include: the encoding of sensory stimulation; the functional specialization of memory components and inter-component communication; the realization of choice functions through competitive response classes; the realization of disjunction through cooperative response classes; the representation and realization of hierarchical or goal-oriented knowledge and computational control structures; the representation of sequential programmatic behaviors using transformational chains; the view of memory as a single, integrated knowledge or structure transforming unit; and the relation of the SR system to the view of natural behavior as an immediate and associative process. In addition, several important memory functions not considered in this paper are mentioned. Each of these major topics is now discussed in

turn.

Encoding

The question of encoding concerns the manner in which descriptions of stimuli and internal conditions come to exist in memory. Both as a question of experimental psychology and one of theory (theoretical psychology, artificial intelligence, pattern recognition) the specification and computation of encoding functions are of central importance because of their role in determining the primitive predicates which are the representational and substantive bases for all acquired knowledge (see, for a variety of views on what such primitives might be and how they are connected by learning: Melton & Martin, 1972; Hubel & Wiesel, 1965; Hebb, 1949; Winograd, 1972; Kolers & Eden, 1968; Winston, 1970; Uhr, 1973). No claim is made here about how such encodings are naturally performed, although chapter 10 presents some related speculations. The current problem is simply to discuss how the results of such functions are introduced into the SR representation of behavior in ways which are compatible with the stimulus-driven, transformational, and predicate relational framework.

Briefly, encoding refers to the collection of behaviors that organize and store information about energy impinging on receptors. This information is presumed representable in terms of parameterized, conjunctive predicate relations of PSRs. Of course, natural or artificial systems may have limited attentional or processing capacities, and

the encoding responses of the organism will clearly be affected by such factors (LaBerge & Samuels, 1974; Neisser, 1967; Norman, 1969). In any case, an encoding response is modeled in the SR system by the automatic production of PSRs and their storage in various sorts of memories which may be distinguished by their storage capacities, time constants, and communication channels.

Differentiated memories, communication and reconstruction

It is not necessary for the representation of general computation that memory be differentiated and specialized. In fact, most computer application and simulation programs make no such distinctions. When large-scale computer systems are designed, however, questions about the time constants (read, write, decay, and refresh times) and communicability among necessarily differentiated storage devices are frequently crucial. In view of the apparent functional complexity and physical differentiation of natural nervous systems and human memories (Wooldridge, 1963; Norman, 1970; Arbib, 1973), one might reasonably assume that this is *a fortiori* true of them.

Because the SR framework is basically a content-free, structural framework for knowledge representation, the differentiation of memory stores and the specification of which stores may influence others are not issues which it addresses directly. However, some simple conventions based on the differentiation of two memories, an active *short term memory* (STM) and an inactive *long term memory* (LTM) are suggested

(having no necessary relation to STM and LTM in humans). Specifically, it will be assumed that relations occurring in contingencies of transformations which do not somehow specify a particular memory location refer to patterns that are present in STM or, alternatively, are active. STM is supposedly characterized by short (0-60 second) time constants meaning that items placed in STM disappear autonomously relatively quickly. LTM is supposedly a permanent repository of facts (PSRs) and transformations. TXs are automatically invoked when STM patterns match their contingencies. Facts may be retrieved from LTM and placed into STM by reconstruction. This is a generic term referring to any process which examines LTM for a pattern matching a query pattern and places the former in STM when a match is found. The reconstruction of any pattern (PSR or TX) which matches a template T might be explicitly modeled by assuming it to be a direct consequence of placing the following query in STM,

{p:activate, template:T}, (3.14)

or by assuming that every subpattern A' of any pattern A in LTM is associated with a reconstructive transformation like

{p:TX, parms:parameter set of A, C:A', R:A} (3.15)

or by any other equivalent mechanism. In fact, the convention actually opted for in this paper is to allow implicit mechanisms of reconstruction, functionally equivalent to those in (3.14) and (3.15), to operate (see section 7.5 for details).

The preceding discussion illustrates how two memories may be arbitrarily distinguished and communication channels modeled in the SR system. Generalizing this procedure to any number of memory stores, if desired, is straightforward. Any further consideration of such issues is simply beyond the scope of the current paper. The preceding discussion has been presented primarily to indicate the suitability of the SR framework as a basis for modeling typical, multicomponent memory systems.

Competitive response classes: Mechanisms of choice

Much of the activity of a complex information processing system seems organized around problems requiring the selection of the one response among a set of alternatives which is most appropriate in a given stimulus or problem context. As an example, Milner (1957) introduced the notion of mutually inhibitory connections among alternative cell assemblies as a model of choice in the framework of cell assembly theory. Similar mutually inhibitory structure appears to be a general rule of organization in natural nervous systems (Ratliff, 1965; Bekesy, 1967; Cornsweet, 1970; Dodwell, 1970). In all cases, mutual inhibition among members of a category of possible responses seems to serve one principal function: it sharpens the contrast among the alternatives and acts to insure that only the strongest candidate for response to a particular stimulus situation is chosen. In fact, Plum (1972) has suggested that the basic mechanism of choice or decisionmaking is

the competitive response class, a set of detectors of which at most one may be simultaneously active.

Regardless of its notational and conceptual simplicity as a model of choice behavior, the competitive response class may in fact be no more than an abstract construct for a simple, underlying phenomenon. This simple phenomenon, following Milner (1957), would be the independent learning of excitatory and inhibitory associations between varying contextual preconditions and response alternatives--a product of basic learning functions under the control of response-contingent reinforcement. Specifically, actions which follow a stimulus and are temporally proximate to rewarding circumstances are strengthened and modeled in the SR system by a contingency-response TX with an excitatory (positive) response to that stimulus or its components (see Chapter 4). Conversely, actions seemingly leading to undesirable consequences are suppressed and are modeled by an inhibitory (negative) response in a corresponding TX (Hayes-Roth, 1973a). In the SR system, the type and strength of a transformation may be reflected by an appropriate entry in the unit statistics associated with that TX.

There would appear to be little reason for anyone to disagree with these suggestions. Their principal impact is upon the issue of whether competitive response classes are a primitive concept in the organization and representation of behavior. Because individual contingency-response TXs appropriately detailed to express strength of association

and to distinguish excitation from inhibition are sufficient for the representation and functional modeling of competition classes, it seems unnecessary at this point to demand that the classes be given a distinct representation or a primitive organizational role. The alternative, representing competitive classes implicitly in terms of exclusive response alternatives under the control of several corresponding excitatory and inhibitory transformations of variable strengths, is adopted here.

Cooperative response classes: the realization of disjunction

It is often the case that a single response is appropriate for numerous stimuli which are in no obvious way similar. For example, the adjective "dirty" is applicable to a soiled face, cloudy water, excessively radioactive nuclear fallout, an obscene joke, and dishonorable conduct. This is an example of a disjunctive contingency, a set of conditions each of which is adequate to evoke a common response. It is typical to attempt to model such behavior in terms of a single rule with one contingency expressed in part as a logical disjunction of appropriate terms (Bruner, et al., 1956; Hunt, 1962). The artificial acquisition of such rules is frequently quite difficult and the procedures used convey the impression that no simple general mechanism would suffice. Thus, given the apparent frequency with which such rules appear in natural behavior, the conclusion would seem to be that these behaviors will necessarily be

difficult to model.

The construct which suggests itself in view of the preceding discussion of competitive response classes is that of a cooperative response class comprising a set of conditions any of whose occurrence is sufficient for the production of the common response. This construct is unnecessary, however, since the same technique used to represent a set of competing responses, production of separate, independent and appropriate transformational rules, is equally applicable to the current problem. Here, however, a set of cooperative TXs are used, each TX possessing the same response component but reflecting distinct contingencies. Moreover, the cooperative transformations may differ in their excitatory strengths, reflecting varying amounts of certainty and reinforcement, and may be supposed to induce their common response either totally independently or by some process of summation (see, for examples, Luce, 1959; Good, 1965; Selfridge, 1959; Selfridge & Neisser, 1963) of the strengths of all transformations whose contingencies are simultaneously satisfied. Experimental evidence consistent with such summation of independent evidential components may be found in Reitman & Bower (1973), Underwood & Zimmerman (1973) and Hayes-Roth & Hayes-Roth (1974). Nevertheless, no position is taken here on which of these two alternative strength algorithms should be generally employed within the SR system. While it is clear that totally independent rules are statistically preferable to (perform better than) summation

procedures (Hayes-Roth, 1973b, 1974b; see chapter 4), it is still plausible that summation processes are a more typical basis for natural disjunctive behavior.

In short, physically independent transformational rules possessing a common response component may be assumed as s representation of any disjunctive rule. Whether the strength (certainty, intensity, duration, etc.) of the response is best computed by maximization or summation over the cooperating TXs which are concurrently satisfied is a question whose answer is completely dependent upon the objective of the computational system (e.g. simulation vs. pattern recognition).

Hierarchies and goal-oriented control structures

It is frequently argued that complex behavior is best characterized by a hierarchical organization, each lower level representing subordinate operations or subgoals of the operation or objective at the immediately superior level and that computation in hierarchies is controlled by top-down methods (Miller, et al., 1960; Ernst & Newell, 1969; Nilsson, 1971; Arbib, 1969; Neisser, 1967; Winograd, 1972). Top-down methods are defined by the temporary transfer of computational activity to each subordinate from its superior and the recovery of that activity by the superior upon the completion of the subordinate's task. One speaks, in this context, of the superior function calling the subordinate and the latter returning (upon its termination) control to the caller.

Adherence to top-down philosophies of modeling natural behavior is apparently based largely upon the beliefs that: (1) the development of frequently used, general subfunctions which are later applied to a wide variety of taks is adaptive; and (2) that the application of subfunctions to new tasks requires top-cown control structures. Although I do not disagree with the first argument, it is clear that the second is patently incorrect. For example, Plum (1972), extending the work of Friedman (1967), incorporated the evidential aspects of hierarchies (Neisser, 1967) by priming lower-order cell assembly units without actually invoking them. The primed units then responded with greater strength to signals from their own subordinate units.

Although Plum considered his technique an example of the execution of top-down control by a set of distributed processors, it is in actuality an example of bottom-up or stimulus-driven computation (SDC). This follows from the fact that control is not passed and returned between calling and called programs. In a similar manner, any hierarchical organization may be realized in an SDC framework. That this is true can be seen by considering an arbitrary top-down, hierarchical system in which a call to a subordinate function B occurs when program A is in location (state, condition, etc.) X, denoted A(X), and B returns upon completion to the next location in A, say A(X+1). Diagrammatically, this sequence is A(X)→B→A(X+1). This

sequence could equally well be performed by an SDC using two independent TXs of the following forms, A(X)→B and (A(X), B)→A(X+1), where the left-hand side of each rule represents the contingency for its activation, the right-hand side represents the associated response, and the pattern (A(X), B) denotes the sequence of events A(X) succeeded by B.

Although it is thus possible to represent hierarchical activity in the framework of SDC, it is not self-evident that for all purposes this ought to be done. Rather, such an approach is suggested because it is consistent with the belief that natural computation is essentially a stimulus-response process, with the understanding that "stimulus" and "response" both refer to the structured, internal state of the organism. Furthermore, it can be shown that simple, general functions exist which enable the acquisition of transformational rules capable of effecting behavior of any complexity (Appendix IV). Thus, the ability of the proposed transformational framework to represent hierarchical organizations in a simple, natural way is only one of several capacities which motivate its investigation as a basis for modeling behavior.

Chains or programs

Behavioral chains, in which a set of responses are sequentially organized, and serial computer programs, in which a set of logical operations are sequentially organized, if only for their apparent ubiquity, support the

idea that sequencing is a fundamental principle of computational organization of comparable significance to that of hierarchies. On the other hand, it is obvious how chains are realized in the SR framework. Each chain is the manifestation of a series of single transformational responses to appropriate contingencies.

Suppose a chain of interest is A before B before C ... before Y before Z. It is clear that a set of independent TXs which exhibit the same sequence of responses is, using the sequence notation introduced above: A→B, (A, B) →C, ..., (A, B, ..., Y)→Z. Here, each rule is a simple contingency-response pair. The "sequences" on the left-hand side of these rules (e.g. (A, B, ..., Y)) represent temporal patterns that are to be <u>recognized in toto</u>, not <u>produced</u>. To the extent that a capacity to exhibit chains is fundamental to complex behavior and that the SR system is an accurate model of behavior, it is thus axiomatic that complex organisms are necessarily capable of representing, discriminating, and recognizing such temporal patterns.

Some readers may question the desirability of a representational framework in which a simple notion like the serial execution of a list of operators (a program) is necessarily expanded into a set of rules perhaps containing contingency patterns of apparently great complexity. Such an expansion would clearly be undesirable for program implementation on current computers. On the other hand, the

goal here is to represent and understand natural computation. There is no reason whatsoever to believe that single transformational rules with contingencies of any complexity are less "primitive" or "fundamental" in natural systems than are behaviors which have simple contingencies and serial response configurations. Again, the SR system is suggested for consideration not because it provides a verbally trivial representation of typical computer programs, but because of the totality of functions which may be understood in terms of its elements. Only if it were impossible to represent some aspect of behavior organization in a straightforward way, would the value of the proposed system be questioned.

The integrality of memory and the transformation of structure

In the SR system, viewed as a model of natural computation, there are four principal components: (1) event representations (PSRs); transformations (TXs); (3) unit statistics; and (4) learning functions. PSRs and TXs are the information structures stored within the physical memory structure, unit statistics regulate the accessibility of information structures during processing activities, and learning functions are conceived as continuous activity which somehow modifies the contents of memory and the unit statistics in an intelligent manner. As an example of a system manifesting four corresponding constructs one might hypothesize that the human nervous system: (1) employs

nervous firing patterns as a representation of events; (2) continually transforms some patterns into others on the basis of previously learned or innate processing rules which are realized by distributed, variable, synaptic transmission coefficients; (3) regulates the availability of information according to individual neuron state variables (e.g. fatigue, habituation, priming) and interneuronal associative strengths which are a function of prior sequential activities, reinforcement, attention, etc.; and (4) learns by continually modifying synapses in some adaptive way.

If the hypotheses in the preceding discussion of human information processing are true (or if others which interact in comparably complex ways are), the construction of an accurate brain model from entirely separable units is probably impossible. This follows from the inability to isolate physical equivalents for each of the principal memory constructs: contents, activity, structure, and learning. On the other hand, it would still be possible to simulate these constructs and thereby achieve a useful model of the natural systems of interest. That is the purpose of the SR system.

The advantages of the proposed system for such modeling are several, and they depend on a number of weaker assumptions than those above. At this point, I will briefly identify these advantages and the assumptions from which they seemingly derive.

(1) It is assumed that natural behavior is a stimulus-driven (transformational) process, the effective stimulus for each TX being the internal state of the organism which, for many purposes, may simply be associated with the various activity levels of the contents of memory. An advantage of the SR system is that it provides a natural basis for the representation of such behavior.

(2) It is assumed that much knowledge is simply the logical organization of representations which are internal correspondents of environmental objects (e.g. semantic memory, Tulving & Donaldson, 1972) or the representation of the organization of environmental objects (e.g. pattern description). An advantage of the SR system is its ability to represent such organizations in the same simple way in terms of PSRs which may comprise whatever predicate relations required.

(3) It is assumed that knowledge is decomposable into functions that, in natural systems, are computed asynchronously, in parallel, and independently by distributed, immediate, and associative processes. An advantage of the SR system as a model of such systems is its corresponding computational organization and behavior.

Untreated aspects of memory control

There are many aspects of memory control which, although clearly important for a complete model of natural behavior, are beyond the scope of the current paper. For example, no effort is expended here on establishing care-

ful models or operational definitions of storage, rehearsal or retrieval. Similarly, issues concerning consolidation, conditioning, and differential saliencies of representational components are ignored. No attempt is made to decide the philosophical question of determinism or to represent behavioral non-determinacy. Finally, although spatial and holographic representations of information seem strongly implicated in natural memory processes, they are not considered in this paper.

3.6 Relationship Between the SR System and Other Computing Frameworks

The PSRs of the SR system are almost identical to the lambda expressions of the programming language LISP (Church, 1941; McCarthy, 1962; Weissman, 1967). One principal difference is the definition of the parameter set and body of a PSR as sets rather than lists. Another important difference is that of semantic interpretation. PSRs are conceived as descriptions rather than expressions or computational forms. Thus, one goal of the SR system is to permit an analysis of the means by which a structured pattern can be described and how the related description can be directly and immediately compared (without intermediate computation) to templates in memory which are contingencies of transformational rules.

The previous work which is most similar to the current work is that of Evans (1969a, 1969b), Fikes & Nilsson (1971),

and Rulifson, Derksen, & Waldinger (1972). Each of these projects has used parameterized sets of relations to describe structured events. Evans, in particular, attempted to construct pattern descriptions from lower level functions which represented subordinate, component relations and to recognize patterns by a top-down computation of the components specified by relational templates. (Although restricted to the domain of binary connectives in graphical context, Barrow, et al. (1972) also proposed heuristic methods for matching structured stimuli and templates.) Fikes & Nilsson discuss methods of matching complex contingencies and the representation of problem solving procedures in terms of transformational rules. Rulifson, et al., describe the organization of essentially top-down problem solving procedures about a mechanism for associatively accessing functions whose invocation will satisfy some goal; this is determined by matching patterns describing the caller's goal and the subordinate function's purpose. It would appear that the SR system is a synthesis of these three ideas plus that of SDC, which was developed in Selfridge's Pandemonium (1959), Rosenblatt's perceptron (1958), and Hebb's cell assembly theory (1949).

This chapter should be concluded with a statement of what may alternatively be called the recognition, decision, or matching problem. This is deciding when the current contents of memory satisfy the set of conditions constituting the contingency for an associated response. This is truly

the essential problem in the realization of SDC systems. When contingencies are allowed to be as complex as PSRs (equivalent in descriptive power to labeled, directed graphs), the recognition problem is equivalent to the general graph covering or monomorphism problem: decide whether one graph contains another as a subgraph.

Practical solutions to that problem thus become important for two reasons. First, they will be required by any effective application of these concepts. Second, they will be important evidence in support of SDC as a plausible model of complex natural behavior. While the current paper is the first to my knowledge to address the latter issue, much effort has already been spent on the former problem in the context of decision tables (see, for example, Humby, 1973) in computer and management sciences. There the basic problem has been identifying good methods for converting any set of contingency-response rules into an efficient, equivalent, serial computer program. That problem is clearly important, both theoretically and pragmatically, given the nearly complete dependency of current computing tasks (including SDC simulations) upon serial digital computers. On the other hand, decision table programs are, in my opinion, poor candidates for models of natural SDC systems (notwithstanding the seminal, successful work of Feigenbaum (1963) on EPAM nets) and, as a result, the search for plausible candidates remains a principal objective (see Chapter 5).

Chapter 4: PERCEPTION AND PATTERN LEARNING

4.1 Chapter Overview

In this chapter the recognition problem, deciding whether or not a stimulus matches the contingency of a transformation, and the pattern learning problem, determining the contingencies for pattern recognition transformations, are operationally defined. The proposed approach to the solution of the learning problem, in terms of alternative hypothetical TXs, is discussed. Subsequently, the details of the solution are provided in terms of an interference matching (IM) algorithm. The IM algorithm is shown (Appendix IV) to provide a complete solution to the recognition problem. Because the application of the IM algorithm to PSRs is at the core of all of the learning procedures presented in this paper, the explicit detail required of PSRs for the most general use of the IM algorithm is also discussed. Finally, the definition of perception in terms of four component operations and the chief contributions and deficiencies of the current approach vis-a-vis such a definition are considered.

4.2 The Recognition Problem

The recognition problem is, simply stated, to decide when the representation of a stimulus matches a template, the contingency for a related perceptual or recognition response. Consider, for example, the previously suggested

problem of recognizing the "A" in figure 2, whose PSR A is given in (3.5), as an exemplar of the Gothic "A" template in figure 5, whose PSR T(A) is given in (3.8). Because the actual predicates and objects used in these PSRs are arbitrary, a general procedure which solves this exemplary problem will be equally capable of solving all structured event recognition problems.

In order to decide whether the PSR A satisfies the contingency specified by T(A), denoted A (*) T(A), it is necessary to show that every relation in T(A) is also present in A. This is called the all-or-none matching problem. In order to solve the all-or-none matching problem, however, it is first necessary to identify a correspondent in A for each parameter in T(A) such that the body of A contains as a subset the body of T(A) when the alphabetic differences between corresponding parameters are ignored. The binding problem is to determine how the identification of correspondents should be performed to enable a solution to the matching problem. Unfortunately, a solution to the binding problem necessarily entails a solution to the matching problem, and vice versa. As a result, developing a solution to the recognition problem in terms of solutions to both of its principal subproblems, binding and all-or-none matching, is a nontrivial task. In fact, one of the best solutions for the recognition problem, in which both subproblems are solved simultaneously, utilizes a procedure that is naturally associated with another, more general problem, that of pattern

learning. Thus, further discussion of the recognition problem will be delayed until the pattern learning problem and its solution have been presented.

4.3 The Learning Problem Conceptualized

A context in which the learning task may be clearly viewed is the "concept learning" or "abstraction" problem illustrated in figure 9. The four patterns X1, X2, X3, and X4 on the left are exemplars of the category X and those on the right, Y1, Y2, Y3, Y4, are exemplars of the category Y. The learner's task is to infer rules from the training exemplars which can be successfully used to classify them (post hoc) and, in addition, any novel test items presented in the future. The reader may have already noticed that X1 is the same pattern as in figure 3 (with PSR B of (3.6)) and that the underlying concept described by figure 6 (with PSR T(B) of (3.9))is actually the rule used to generate the other exemplars of X. That is, each element of class X comprises two vertically configured pairs of nested geometric forms such that the interior forms have the same shape. Class Y is simply conceived to be the set of all patterns that are not members of X.

The pattern learning problem may be formally defined in a straightforward way in the context of this illustrative abstraction task. In general, there are four elements in a learning problem: training information, response alternatives, contingency alternatives, and performance criteria. Each of

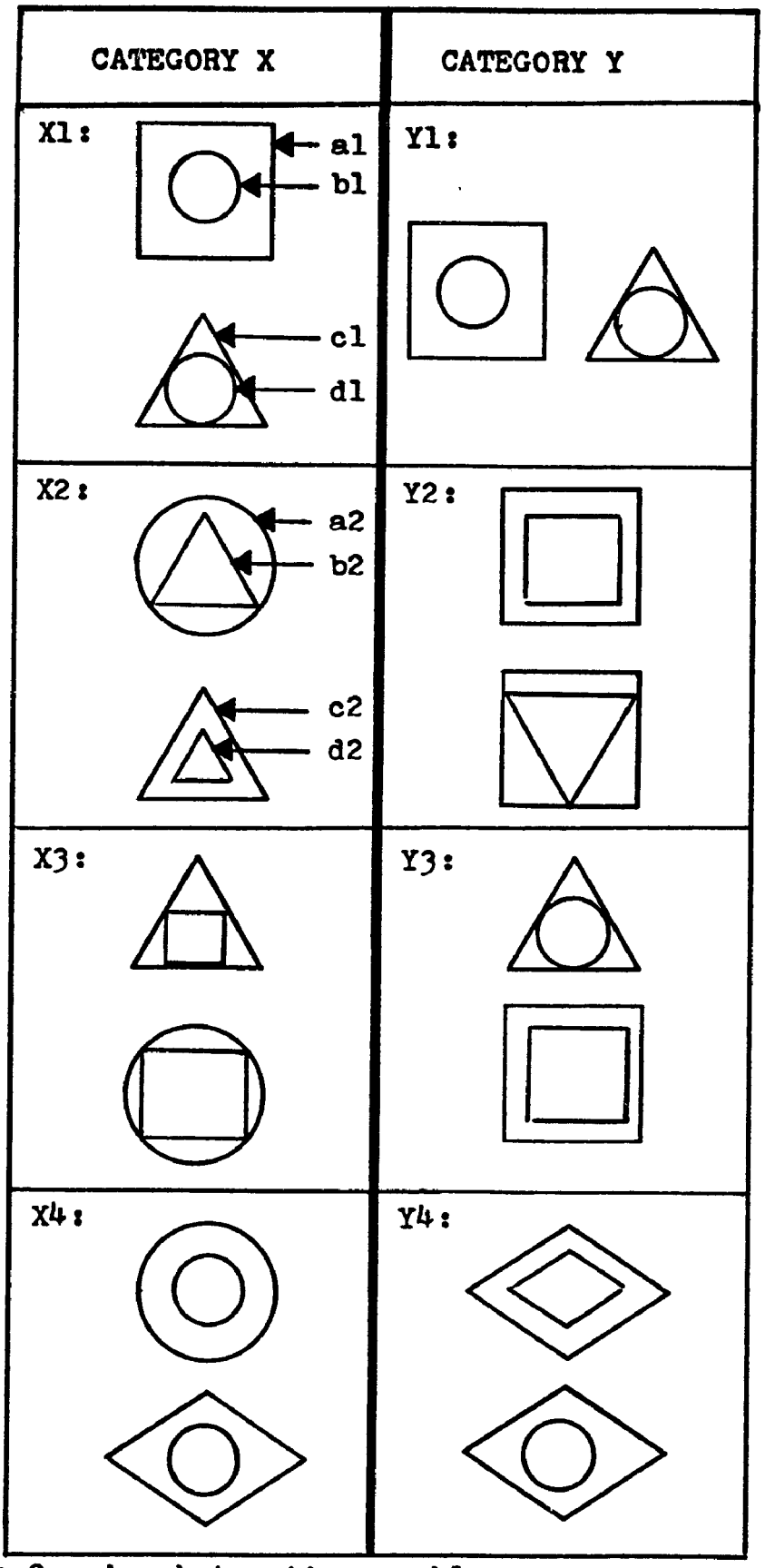


Figure 9. An abstraction problem.

these is considered in turn.

The training information for a learning problem consists of a set of paired stimulus and response descriptions (PSRs). For example, the PSR of any exemplar Z (Z = X1, X2, ..., Y4) in figure 9 could be simply denoted Z and the PSR of the associated response might simply be denoted X or Y. In that case, the training information I for the current concept learning problem would simply be the set of ordered pairs (two-tuples):

$$I = \{(X1, X), (X2, X), (X3, X), (X4, X), (Y1, Y), (Y2, Y), (Y3, Y), (Y4, Y)\}. \qquad (4.1)$$

The response alternatives of the problem are determined by the kinds of responses that might be abstracted (selected or inferred) from the training examples. In the current problem, the response alternatives are simply X or Y, corresponding to the two possible ways items are classified.

The contingency alternatives are, in a similar way, the kinds of preconditions for any response which might be abstracted from the stimulus descriptions in the training information. Given the type of PSR used for X1 (B) in (3.6) and those that would probably be used for the other concept examples, the contingency alternatives are a large set of conjunctive relational descriptions about the stimulus patterns, e.g. "contains one square and one triangle" or "contains four figures of any shape such that two of them have the same shape." Specifically, any set of predicate relations which is true of all pattern exemplars is a possible contingency

for a corresponding conjunctive classificatory rule, and any which is true of at least one exemplar is a possible contingency for a corresponding disjunctive rule.

Finally, performance criteria determine how to weight the relative effects of correct and incorrect behaviors (responses to specific stimuli) and how much significance to attribute to the information in the training samples. In the current example, one might arbitrarily specify that an erroneous classification behavior should be weighted ten times as "undesirable" as a single correct classification is evaluated "desirable." Further, it might be independently specified or estimated that the true proportion of X test items in the future will be 0.5, that there is no a priori information about the relative likelihood that any particular contingency-response rule is preferable to any other, and that the system's goal is to maximize the expected net overall weighted difference of correct less incorrect classification responses. These statements simply illustrate the sorts of details that are necessary in the formal statement of performance criteria for a solution to a learning problem. Later, some specific performance criteria are recommended as a desirable basis for artificial learning and a potentially useful model of corresponding natural behavior.

4.4 An Overview of the Learning Problem Solution

The proposed solution to the learning problem depends

centrally on the concept of an abstraction of two or more representations in a reference set. Briefly stated, an abstraction is a common subrepresentation of each of the reference representations or, equivalently, a PSR which is matched by each of the reference PSRs. Thus, an abstraction of several events (patterns) is any subevent (subpattern) of each of them. In order to infer rules which relate responses to contingencies in ways which are common to several experiential exemplars, training examples with similar responses are first compared to produce abstractions which are matched by several of the associated stimulus events. Second, each of these abstractions is considered a structural template representing an hypothesis about the characteristic which is the criterion for membership in the related pattern (response) class. The performance of each such template as a predictor of class membership is then evaluated. TXs may then be produced either by employing only the most highly performing templates as contingencies for the corresponding categorical response or by allowing all possible such transformations to compete and cooperate in some reasonable manner, with strengths directly related to their performance values (see section 3.5). In the next section, these operations are described precisely.

4.5 Solving the Pattern Learning Problem Using Interference Matching, a Mechanism of Abstraction

Categorical behavior

The pattern learning problem requires the generation of categorical transformations, ones whose response components represent some fixed behavior like "X" or "Y" for each stimulus pattern which matches their contingencies. Furthermore, since the solution proposed employs the training exemplars from only one response category at a time, the learning problem may be reduced to two simple subproblems. The first is to produce abstractions of several exemplars of the same class. The second is to evaluate the performance of these abstractions as indicators that stimuli which match them actually belong to the class with which they are associated.

Models

One principal conceptual device in the development of a solution to the learning problem is the model, a structure relating several representations to a common abstraction. Specifically, a model of the reference set E of PSRs E1, E2, ..., En is the triple (P, A, R) where P is a set of parameter correspondences among the parameters of elements of E, A is an abstraction of each PSR in E, and R = (R(E1), R(E2), ..., R(En)) is a set of residuals, the residual R(Ei) being the set of all relations in the body of Ei (i=1,2,...,n) which are not represented in the abstraction A. In other words, the abstraction contains those relations which have been identified as common to each event, under the assumption that certain parameters in each event are structural correspondents, and the residuals identify the remaining

relations which are not "explained" by the common abstraction. Models are produced by noting the occurrence of similar predicate relations in each compared representation and forcing those parameters which occur as corresponding objects into an hypothesized correspondence or binding relation. The process of generating models of a reference set is called partial-matching.

Partial-matching using the IM procedure

In order to provide a concrete example for the development of the technique of IM to accomplish partial-matching, the following explicit (non-compact, all objects are parameters) PSRs for the first two exemplars, X1 and X2, of category X (figure 9) are provided:

```
X1 = {p:event, parms:{a1, b1, c1, d1},
      body:{R1:{p:geometric form, name:a1},
            R2:{p:geometric form, name:b1},
            R3:{p:geometric form, name:c1},
            R4:{p:geometric form, name:d1},
            R5:{p:square, name:a1},
            R6:{p:circle, name:b1},
            R7:{p:triangle, name:c1},
            R8:{circle, name:d1},
            R9:{p:contain, outer:a1, inner:b1},
            R10:{p:contain, outer:c1, inner:d1},
            R11:{p:above, upper:a1, lower:c1},
            R12:{p:above, upper:a1, lower:d1},
            R13:{p:above, upper:b1, lower:c1},
```

R14:{p:above, upper:b1, lower:d1},

R15:{p:same shape, name:b1, name:d1}}} (4.2)

X2 = {p:event, parms:{a2, b2, c2, d2},

body:{S1:{p:geometric form, name:a2},

S2:{p:geometric form, name:b2},

S3:{p:geometric form, name:c2},

S4:{p:geometric form, name:d2},

S5:{p:circle, name:a2},

S6:{p:triangle, name:b2},

S7:{p:triangle, name:c2},

S8:{p:triangle, name:d2},

S9:{p:contain, outer:a2, inner:b2},

S10:{p:contain, outer:c2, inner:d2},

S11:{p:above, upper:a2, lower:c2},

S12:{p:above, upper:a2, lower:d2},

S13:{p:above, upper:b2, lower:c2},

S14:{p:above, upper:b2, lower:d2},

S15:{p:same shape, name:b2, name:d2},

S16:{p:same shape, name:b2, name:c2},

S17:{p:same shape, name:c2, name:d2}}}.(4.3)

Notice that each relation in X1 is named R$\underline{i}$ ($\underline{i}$ = 1, 2, ..., 15) and each in X2 is named S$\underline{j}$ ($\underline{j}$ = 1, 2, ..., 17) in order to facilitate the description of models of {X1, X2} in the subsequent paragraphs.

Although a variety of techniques for partial matching exist (Hayes-Roth, 1973b), the one considered in this paper is the combination serial-parallel two-step <u>interference</u>

matching (IM) procedure described by the flow diagram in figure 10. The operations of the IM procedure will be explained as they are illustrated by partial-matching X1 and X2.

The first step in partial-matching is to identify those relations in X1 which are possible matching correspondents for relations in X2 because they contain identical predicates. The potential correspondence of matched relations reflects the possibility that, under appropriate parameter bindings, one relation represents exactly the same feature in one event that its matched correspondent represents in its own event. For example, it might be supposed that object b1, the uppermost circle in X1, corresponds to a2, the uppermost circle in X2 because both occur in a relation with predicate "circle." Alternatively, b1 might be supposed to correspond to b2 either because these parameters both occur in potentially matched "contain" relations in the object term of type "upper" or because they both occur in relations with predicate "same shape." In short, each pair of matchable relations in X1 and X2 gives rise to a plausible hypothesis about correspondences between parameters in the two event representations. Conversely, hypothesized parameter correspondences reflect the fact that there are objects in both events which are similar at least to the extent that they play identical functional roles in one relation.

Table 1 lists the complete set of models produced in

Step 1:

In parallel, in as many ways as possible, a set of relations with identical predicates is formed by choosing one from each partial-matched PSR (E1, E2, ..., En), and a model corresponding to this set is then generated, as follows. Each parameter in one relation is asserted to be the correspondent of one parameter in each of the matched relations which is associated with an identical object type. An abstraction is formed consisting of a single relation of the same type as that of the matched relations; the parameter objects are renamed as the set of corresponding parameters related to that object term. Finally, the residual of each element Ei in this model is the original body of Ei less the one relation which has been matched in this process.



Step 2:

In parallel, in as many ways as possible, sets of consistent and mergeable models are merged to form more informative, more complete partial models. Models are consistent if each parameter is assigned at most one correspondent in each PSR. Models are mergeable if the abstraction of each model is matched by each residual of the other models or, equivalently, if their abstractions arise from complementary sets of relations in the matched PSRs.

Figure 10. A flow diagram of the interference matching (IM) procedure in the production of models of a reference set of exemplars E = {E1, E2, ..., En}.

TABLE 1

MODEL	P	A	R(X1)	R(X2)	MATCHED RELATIONS
M1	{aa:{X1.a1,X2.a2}}	{GF(1,1):{p:geometric form, name:aa}}	X1-{R1}	X2-{S1}	R1,S1
M2	{ab:{X1.a1,X2.b2}}	{GF(1,2):{p:geometric form, name:ab}}	X1-{R1}	X2-{S2}	R1,S2
M3	{ac:{X1.a1,X2.c2}}	{GF(1,3):{p:geometric form, name:ac}}	X1-{R1}	X2-{S3}	R1,S3
M4	{ad:{X1.a1,X2.d2}}	{GF(1,4):{p:geometric form, name:ad}}	X1-{R1}	X2-{S4}	R1,S4
M5	{ba:{X1.b1,X2.a2}}	{GF(2,1):{p:geometric form, name:ba}}	X1-{R2}	X2-{S1}	R2,S1
M6	{bb}	{GF(2,2)}	X1-{R2}	X2-{S2}	R2,S2
M7	{bc}	{GF(2,3)}	X1-{R2}	X2-{S3}	R2,S3
M8	{bd}	{GF(2,4)}	X1-{R2}	X2-{S4}	R2,S4
M9	{ca}	{GF(3,1)}	X1-{R3}	X2-{S1}	R3,S1
M10	{cb}	{GF(3,2)}	X1-{R3}	X2-{S2}	R3,S2
M11	{cc}	{GF(3,3)}	X1-{R3}	X2-{S3}	R3,S3
M12	{cd}	{GF(3,4)}	X1-{R3}	X2-{S4}	R3,S4
M13	{da}	{GF(4,1)}	X1-{R4}	X2-{S1}	R4,S1
M14	{db}	{GF(4,2)}	X1-{R4}	X2-{S2}	R4,S2
M15	{dc}	{GF(4,3)}	X1-{R4}	X2-{S3}	R4,S3
M16	{dd}	{GF(4,4)}	X1-{R4}	X2-{S4}	R4,S4
M17	{ba}	{CIR(6,5):{p:circle, name:ba}}	X1-{R6}	X2-{S5}	R6,S5

TABLE 1 (continued)

M18	{da}	{CIR(8,5):{p:circle, name:da}}	X1-{R8}	X2-{S5}	R8,S5
M19	{cb}	{T(7,6):{p:triangle, name:cb}}	X1-{R7}	X2-{S6}	R7,S6
M20	{cc}	{T(7,7):{p:triangle, name:cc}}	X1-{R7}	X2-{S7}	R7,S7
M21	{cd}	{T(7,8):{p:triangle, name:cd}}	X1-{R7}	X2-{S8}	R7,S8
M22	{aa,bb}	{CON(9,9):{p:contain, outer:aa, inner:bb}}	X1-{R9}	X2-{S9}	R9,S9
M23	{ac,bd}	{CON(9,10):{p:contain, outer:ac, inner:bd}}	X1-{R9}	X2-{S10}	R9,S10
M24	{ca,db}	{CON(10,9):{p:contain, outer:ca, inner:db}}	X1-{R10}	X2-{S9}	R10,S9
M25	{cc,dd}	{CON(10,10):{p:contain, outer:cc, inner:dd}}	X1-{R10}	X2-{S10}	R10,S10
M26	{aa,cc}	{A(11,11):{p:above, upper:aa, lower:cc}}	X1-{R11}	X2-{S11}	R11,S11
M27	{aa,cd}	{A(11,12):{p:above, upper:aa, lower:cd}}	X1-{R11}	X2-{S12}	R11,S12
M28	{ab,cc}	{A(11,13):{p:above, upper:ab, lower:cc}}	X1-{R11}	X2-{S13}	R11,S13
M29	{ab,cd}	{A(11,14):{p:above, upper:ab, lower:cd}}	X1-{R11}	X2-{S14}	R11,S14
M30	{aa,dc}	{A(12,11):{p:above, upper:aa, lower:dc}}	X1-{R12}	X2-{S11}	R12,S11

TABLE 1 (continued)

M31	{aa,dd}	{A(12,12)}	X1-{R12} X2-{S12} R12,S12
M32	{ab,dc}	{A(12,13)}	X1-{R12} X2-{S13} R12,S13
M33	{ab,dd}	{A(12,14)}	X1-{R12} X2-{S14} R12,S14
M34	{ba,cc}	{A(13,11)}	X1-{R13} X2-{S11} R13,S11
M35	{ba,cd}	{A(13,12)}	X1-{R13} X2-{S12} R13,S12
M36	{bb,cc}	{A(13,13)}	X1-{R13} X2-{S13} R13,S13
M37	{bb,cd}	{A(13,14)}	X1-{R13} X2-{S14} R13,S14
M38	{ba,db}	{A(14,11)}	X1-{R14} X2-{S11} R14,S11
M39	{ba,dc}	{A(14,12)}	X1-{R14} X2-{S12} R14,S12
M40	{bb,dc}	{A(14,13)}	X1-{R14} X2-{S13} R14,S13
M41	{bb,dd}	{A(14,14)}	X1-{R14} X2-{S14} R14,S14
M42	{bb,dd}	{SS(15,15,1):{p:same shape, name:bb, name:dd}}	X1-{R15} X2-{S15} R15,S15
M43	{bd,db}	{SS(15,15,2):{p:same shape, name:bd, name:db}}	X1-{R15} X2-{S15} R15,R15
M44	{bb,dc}	{SS(15,16,1):{p:same shape, name:bb, name:dc}}	X1-{R15} X2-{S16} R15,S16
M45	{bc,db}	{SS(15,16,2):{p:same shape, name:bc, name:db}}	X1-{R15} X2-{S16} R15,S16

TABLE 1 (continued)

M46	{bc,dd}	{SS(15,17,1):{p:same shape, name:bc, X1-{R15} X2-{S17} R15,S17 name:dd}}
M47	{bd,dc}	{SS(15,17,2):{p:same shape, name:bd, X1-{R15} X2-{S17} R15,S17 name:dc}}

Table 1. First step IM models of X1 (4.2) and X2 (4.3) for exemplars of category X in figure 9. Each model M = (P, A, R), where P is a set of parameter correspondences, A is a set of relations constituting the abstraction of {X1, X2}, and R comprises R(X1) and R(X2), the residuals of X1 and X2, sets of relations in X1 and X2, respectively, which are not represented by A. Each parameter correspondence set (a pair of parameters, one each from X1 and X2) is assigned a name yz, where y and z are parameters in X1 and X2 respectively, as a notational shorthand. Similarly, the abstracted relation from matched relations R_i in X1 and S_j in X2 are named code(i,j[,k]), where code identifies the type of relation and k = 1 or 2 is optional and distinguishes names of two abstracted relations derived from binary symmetric relations.

the first step of the interference matching procedure applied to the PSRs X1 (4.2) and X2 (4.3), where the following notational conventions are used. Each model is named M*i*, *i* = 1, 2, ..., 47. The uniqueness of parameter symbols arising in distinct PSRs is insured by concatenating the event name symbol to them (e.g. parameters in X1 are referred to X1.a1, X1.b1, etc.). A correspondence binding of two parameters, say X1.a1 and X2.b2 as in model M2, is the set {X1.a1, X2.b2} and, for notational simplicity, such a binding is denoted ab. Similarly, each relation in an abstraction is named by a symbol denoting the type of relations and actual relation names which were matched to produce it. For example, the abstraction of model M2 which represents the matching of relations R1 and S2 binds X1.a1 and X2.b2 and contains exactly one relation, denoted GF(1, 2), which is {p:geometric form, name:ab}. This model reflects the fact that a possible commonality between X1 and X2 is to be found by matching R1 and S2: each event contains at least one geometric form (GF), namely a1 in X1 and b2 in X2. Finally, the residuals R(X1) and R(X2) are represented as the set difference between the original PSR body (X1 and X2 for short) and the set of the one relation now represented by the model's abstraction, e.g. R(X1) = X1 - {R1} and R(X2) = X2 - {S2} in M2.

The first sixteen models in table 1 arise from the matching of "geometric form" predicate relations, M17 and M18 from "circle" relations, M19 - M21 from "triangle" relations, M22 -

M25 from "contain" relations, M26 - M41 from "above" relations, and M42 - M47 from "same shape" relations. Note that when binary relations are matched two pairs of parameter correspondence bindings are simultaneously generated. For example, M22 reflects the fact that X1.a1 and X2.a2 correspond in that they both "contain" corresponding objects labeled by the parameters X1.b1 and X2.b2. The parameter binding set P of M22 is thus {aa, bb} = {{X1.a1, X2.a2}, {X1.b1, X2.b2}}. Furthermore, when symmetric binary relations, like those with predicate "same shape," are matched, both permutations of correspondences between the two parameters in X1 and the two in X2 are possible. Thus, each of the models M42 - M47 has a partner which arises from the alternative correspondences. For example, M42 which binds X1.b1 to X2.b2 and X1.d1 to X2.d2 in order to abstract SS(15,15,1) which is {p:same shape, name:bb, name:dd} from R15 and S15 has as such a partner model M43 which binds X1.b1 to X2.d2 and X1.d1 to X2.b2 in order to abstract SS(15, 15,2), the second possible "same shape" (SS) relation abstracted from R15 and S15, which is {p:same shape, name:bd, name:db}.

The second step of the IM procedure combines all sets of consistent, mergeable step one models. Models are consistent if each parameter in one representation occurs in at most one parameter correspondence binding, i.e. if parameter correspondences are at most one-to-one. On the other hand, two models are mergeable if each abstraction is contained by the

other's residuals when the alphabetic differences between corresponding parameters are ignored. Equivalently, two models are mergeable if the matched relations giving rise to one's abstraction are present in the other's residuals.

The process of combining models continues until no additional merger of consistent and mergeable models is possible; then (see App. IV) two distinct classes of models will be around. The important one for most purposes is a minimal complete set of maximal abstractions. An abstraction is maximal in a set if it is not contained (matched) by any other abstraction in the same set. Submaximal abstractions include all others. Because every submaximal abstraction that could be generated by successively deleting relations from a maximal abstraction will in fact be produced by the IM procedure, the set of maximal abstractions is minimal complete in the sense that it is the smallest set of representations that directly exhibits all information about the commonality of the matched PSRs.

The cardinality of the set of maximal abstractions is bounded by the number of possible unique one-to-one parameter correspondence binding functions. If a set of N PSRs is to be partial-matched with parameter sets of cardinalities $n_1, n_2, \ldots, n_N$ where $n_1 \leq n_2 \leq \ldots \leq n_N$, then the maximum number of distinct maximal abstractions which may be produced is

$$K = \binom{n_2}{n_1}\binom{n_3}{n_1} \cdots \binom{n_N}{n_1}(n_1!)^{N-1} . \qquad (4.4)$$

In the current example, the partial-matching of X1 and X2 with four parameters each can produce at most K = 4! = 24 distinct maximal abstractions. This is a very small number of course compared to the potential values of K (4.4).

One principal advantage of the IM procedure over an enumerative algorithm, which would produce all K maximal abstractions by successively computing all possible correspondence binding functions and intersecting the original PSR bodies while treating correspondent parameters as identical constants (Hayes-Roth, 1973b), is that it is a mechanism which may be easily controlled to prune (limit) the number of binding rules considered. If this is done on the basis of the relatively greater success of some sets of bindings to resolve larger or more significant sets of relations than others, such pruning is very desirable. One excellent way of implementing such a pruning mechanism is presented in Appendix I.

Another major advantage of the IM procedure is that it performs the minimum possible number of manipulations in the production of all maximal abstractions. This is a consequence of the fact that two models are merged only if they are guaranteed to yield a consistent, more informative abstraction. Conversely, when no abstractions exist which are more informative than those in the current set, the IM procedure immediately halts.

In the current example, there are 24 possible maximal abstractions. One of these, produced by the merger of models

M1, M6, M11, M16, M20, M22, M25, M26, M31, M36, M41 and M42 arising from the pairwise matching of relations R_i and S_i (i = 1, 2, 3, 4, 7, 9, 10, 11, 12, 13, 14, 15) involves the parameter correspondences {aa, bb, cc, dd} = {{X1.a1, X2.a2}, {X1.b1, X2.b2}, {X1.c1, X2.c2}, {X1.d1, X2.d2}} and is represented by the following compact body of relations:

{{p:geometric form, name:aa, name:bb, name:cc, name:dd},
{p:triangle, name:cc},
{p:contain, outer:aa, inner:bb},
{p:contain, outer:cc, inner:dd},
{p:above, upper:aa, upper:bb, lower:cc, lower:dd},
{p:same shape, name:bb, name:dd}}. (4.5)

The residuals of R(X1) and R(X2) of this model are composed of all relations not accounted for by these mergers. Specifically, R(X1) = {R5, R6, R9} and R(X2) = {S5, S6, S9, S16, S17}. These residuals are interpreted as follows. R(X1) represents the information about which particular shapes a1, b1 and d1 were in X1 since they do not conform to the shapes of their correspondents in X2. Similarly, R(X2) contains the information about the distinctive shapes of a2, b2, and d2 and, in addition, about two pairwise shape equivalences, between b2 and c2 and between c2 and d2, which do not have any correspondents in X1.

The abstraction body (4.5) is interpreted exactly as that of T(B) (3.9) corresponding to figure 6 which was previously claimed to be the concept underlying all exemplars of category X except for one thing. This abstraction states that

the outer figure of the lower pair (cc) is a triangle, something that is true of X1 and X2 but which is not criterial for the concept X. The close equivalence of (4.5) with the body of T(B) can be noted by forcing the parameter correspondences aa=B1, bb=B2, cc=B3, and dd=B4. Thus, the application of the IM procedure to two exemplars of category X has produced, among others, one maximal abstraction of which one large subrepresentation corresponds to X's class characteristic. If this abstraction were subsequently matched to PSRs for X3 or X4 or both, the additional condition that "cc" be a "triangle" would be lost, and a new maximal abstraction exactly equivalent to the class characteristic T(B) would be produced. The same would be true if the entire set {X1, X2, X3, X4} were simultaneously partial-matched using the IM procedure. How such activity is to be controlled and how the resultant abstractions are to be evaluated is the subject of the next subsection.

Schematic classification theory

Schematic classification theory (Hayes-Roth, 1973b, 1974b) is a suggested technique for pattern recognition or classification behavior. It is associated with an underlying characteristic model which relates stimuli or data to their environmental generators by the assumption that each exemplar of a pattern class manifests a class characteristic, a set of co-related features or relations. The characteristic model, in contradistinction to most template,

spatial and generative models (Chapter 2), supposes that each pattern exemplar manifests those features which define the class. These latter models typically suppose that such features are, at best, approximated by the exemplars and, at worst, are literally transformed into different ones (see, for example, Franks & Bransford, 1971).

The characteristic model may be transformed into a classification theory in a straightforward way; the result is schematic classification theory. A schema is any set of features or relations which may co-occur in pattern exemplars. Thus every maximal or submaximal abstraction of pattern exemplars is a schema. The core of the theory is that each schema possesses some predictive power for the task of classifying new exemplars which match it. This power derives from the successful empirical performance of that schema as a basis for distinguishing the positive exemplars of a class from the non-exemplars (counterexamples) of that class.

Although the details of the statistical derivation (Appendix II) are somewhat complex, the results are straightforward. Each schema which is matched by some positive exemplars of a pattern is a plausible hypothetical basis for the concept underlying that pattern. The strength of the schema T to indicate that a test pattern S which matches it belongs to a given class C is an increasing function of the positive frequency with which exemplars of C match T and a decreasing function of the negative frequency with which non-

exemplars of C match T. Although a variety of measures are consistent with the above rule, the fundamental principle of predictive power (Hayes-Roth, 1973b), one in particular is recommended because it is very general, widely applicable, intuitively reasonable, and easy to evaluate. This measure is called the performance of T as an indicator of C, denoted $U(T \Rightarrow C)$. It is the Bayesian posterior expected net gain of weighted correct less incorrect decisions resulting from the classification of all items matching T as belonging to class C, under the assumption that the conditional membership of an item in C which matches T is a Bernoulli process with some probability q to which a uniform prior distribution is assigned.

If the gain attributable to each correct classification of an item in class C is denoted G_C, the loss attributable to an incorrect classification of an item in class C is denoted L_C, the proportion of elements which actually are elements of C is denoted p, the number of training exemplars of class C is denoted $|C|$, the positive frequency of training exemplars of class C matching T is denoted $|C(*)T|$, the number of non-exemplars of C is denoted $|\overline{C}|$ and the negative frequency of those matching T is denoted $|\overline{C}(*)T|$, then

$$U(T \Rightarrow C) = |G_C| \, p \, \frac{1 + |C(*)T|}{2 + |C|} - |L_C|(1-p) \, \frac{1 + |\overline{C}(*)T|}{2 + |\overline{C}|} \quad (4.6)$$

Having thus solved the two principal pattern learning subproblems, the production and performance evaluation of abstractions, the pattern recognition (classification) rule is immediate: A test item S which matches a schema T may

be classified as a member of class C if for no other class C' and schema V matched by S is it true that

$$U(V \Rightarrow C') > U(T \Rightarrow C). \qquad (4.7)$$

In short, each schema is conceived as an independent (disjunctive) characteristic possessing classificatory power. No a priori, arbitrary judgment is made regarding which features may be summed in which ways. Instead, the best rule that could be used in light of the available data is selected to classify an object.

Although the vast number of possible schemata may present no significant problem in the adoption of schematic classification theory as a model of natural learning, it is a very definite impediment in the application of the theory to practical problems in artificial systems. Thus, it is necessary to consider how one might, if necessary, reasonably constrain the set of schemata considered as bases for classification. In even simple cases, the number of schemata is enormous, because it is the cardinality of the union of the complete sets of abstractions (maximal and submaximal) of each pattern exemplar. This selection problem is greatly exacerbated when disjunctive concepts are encountered, because class characteristics need not be matched by all exemplars of the class, nor is it possible to know definitely how many disjunctive terms (distinct schemata) are criterial or which exemplars manifest the same unknown criterial schema.

An algorithm has been developed which handles these

difficulties in the same way for both conjunctive and disjunctive concepts. It is called the Space Limited Interference Matching (SLIM) procedure (Hayes-Roth, 1974b) and is part of a fully programmed but untested learning and classification system. SLIM is chiefly constrained by the number of schemata it may maintain in working memory. Within the limitation imposed on memory space, the procedure performs the following actions in sequence: it successively partial-matches exemplars of each class; it extracts maximal abstractions; it evaluates the unconditional predictive performance measure as in (4.6); it dynamically ranks schemata indicating the same class; it appropriately reduces (conditionalizes) the performance values of schemata which are redundant with (are matched by) more informative and higher performing schemata or are obviated by (match) less informative and higher performing schemata; and, finally, it eliminates from overcrowded storage the lowest conditionally performing schemata. As a result, the procedure dynamically optimizes the overall net performance of all schemata in storage. When the production of such maximal abstractions is complete the schemata from all classes are merged and sorted so that test items may be classified according to the highest performing schema which they match.

As previously mentioned, schematic classification theory is a normative behavior rule, which is suggested because it minimizes expected loss. Whether or not it is

an accurate description of natural behavior is an open question. There is evidence that it may be approximately accurate in a wide variety of cases where the limitations of natural memory processes and the encoded features are successfully determined or guessed by the experimenter. For example, Reitman and Bower (1973) use summed presentation frequencies of letter n-grams (schemata) to produce post hoc explanations of confidence judgments of humans on later recognition of test sets of letters containing n-grams of various levels of prior exposure. Hayes-Roth and Hayes-Roth (1974), using subsets of juxtaposed characters (one or two out of four possible in 2 x 2 character matrices) were able to predict successfully a large number of recognition judgments and, in addition, account for the data of Franks and Bransford (1971) with which the latter had previously rejected simpler "feature" models. Although such "frequency induction" mechanisms as those of Reitman and Bower and Hayes-Roth and Hayes-Roth (see also Underwood & Zimmerman, 1973) are simple approximations of the concept of performance in schematic classification theory, there are at least two reasons why they may prove acceptable for experimental modeling. First, the main criterion for acceptability of a model's prediction is some degree of adequacy to account for data. It is clear that the predictions of frequency induction and schematic performance models are highly correlated and, therefore, very difficult to distinguish. Second, because of the dif-

ficulty of obtaining accurate experimental control over feature encodings and their conjoint frequencies, no precise model will be readily testable without large pragmatic sacrifices.

4.6 Recognition Using Interference Matching

One advantage of the IM procedure not yet mentioned is that it provides a complete solution (see App. IV) to the recognition problem, deciding whether a stimulus pattern S matches a template T in the all-or-none sense, denoted S(*)T. The way this is done is to partial-match S and T using the IM algorithm to produce models of {S,T}, each of which includes a set of parameter correspondence bindings, an abstraction, and residuals R(S) of S and R(T) of T. Each distinct model of {S,T} in which R(T) is empty (null) describes one way in which the pattern S contains the subpattern T or, alternatively, that the pattern S can be perceived to match the template T. Because every occurrence of T in S is identified by one model, the IM procedure provides a complete solution to the recognition problem.

Although the preceding technique is an efficient means of performing all the computation logically required by the complexity of the structured event recognition problem, it does not manifest the apparent immediacy of natural perception. This failure makes the computationally complex IM procedure, in my opinion, untenable as a model of natural

processes of recognition for well known patterns. It simply requires too much computation, with too many delays, to be plausible. In the next chapter, however, the functions of IM which are related to recognition are accomplished by a computationally simple, immediate, homogeneous threshold network memory which obviates the correspondence problems entailed in the current framework of PSRs by the presence of arbitrary parameters. While such a network is of great theoretical interest as a possible model of natural matching procedures, it would seem that the prospects for immediate productive use of the PSR based IM procedure both for natural behavior simulations of learning and recognition and for similar artificial intelligence applications are much more favorable than any which may depend on unusual computational hardware.

4.7 More Detail on PSRs for Matching: Uniform Representations

The purpose of this section is to establish criteria for a uniform PSR which makes available all the content in representations for the generation of abstractions by the IM procedure described previously. The chief weakness of of the current non-uniform PSRs is that they are unsuitable as representational bases for learning problems in which it is necessary to abstract (incorporate into a model) something less than one complete relation at a time. For example, in the preceding example of partial-matching X1 and X2, the maximal abstraction (4.5) failed to reflect that, in both

X1 and X2, one of the four figures was a "circle" (b1 and d1 in X1; a2 in X2). If, additionally, containing a "circle" was criterial to the category X concept, the essential information would be lost by the IM procedure. This occurs because neither X1.b1 or X1.d1 was the correspondent of X2.a2; thus, the equivalence of their predicates was lost because none of the associated relations could be matched in a way which was consistent with the chosen parameter bindings (associating each figure in X1 with the one in a corrsponding position in X2).

In order to remedy this problem, it is necessary to introduce a uniform representation which, unfortunately, is somewhat more abstract than the previously described non-uniform PSRs. In the case of uniform PSRs, there are only three types of relations: symmetric binary same parameter (SP) relations; symmetric binary same set (SS) relations; and unary content relations.

Each relation in a non-uniform PSR may be converted to a set of equivalent uniform relations in the following way. First, it must be insured that every object of every predicate is a parameter. If this is not the case (e.g. "square" is not a parameter in {p:geometric form, type: square, name:b1} as in (3.6)), a corresponding set of explicit relations in which every object is a parameter must be used (e.g. {{p:geometric form, type:s, name:b1}, {p:square, name:s}}). Then, each object type symbol is altered if it is ambiguous to reflect, in addition to the functional role of its object, the predicate of that

relation. For example, the predicate symbol could be concatenated to the object type symbol. At this point, each object term is used to produce a corresponding unary relation, where the predicate is the unambiguous object type symbol and the first and only object is a new, unique parameter symbol which is substituted for the original object parameter. For example, the relation {p:geometric form, name:a} might be translated into the uniform relation {p:geometric form, 1:newa}.

To complete the uniform PSR, SP and SS relations must be generated which relate the new unary relations to one another. SP relations are generated for each pair of new, unique parameter symbols arising from a single parameter in the original PSR. If, for example, the parameters newa and newa2 both arose from the parameter a in two distinct object terms, the relation {p:SP, 1:newa, 1:newa2} would be generated. Similarly, SS relations are generated whenever two new parameters represent different object terms in the same original set (relation). For example, the original PSR relations {{p:geometric form type:s, name:a}, {p:red, name:a}, {p:square, name:s}} might be represented in the following uniform way: {{p:geometric form type, 1:news1}, {p:geometric form name, 1:newa1}, {p:red, 1:newa2}, {p:square, 1:news2}, {p:SP, 1:newa1, 1:newa2}, {p:SP, 1:news1, 1:news2}, {p:SS, 1:news1, 1:newa2}}.

The purpose of such uniform representations is to require each relation to contain exactly one unit of

structured information: that a certain property is true of some object (unary content relations), that two properties modify the same object (SP relations), or that two objects occur in a relational set (SS relations). It is shown in Appendix IV that uniform PSRs contain all the information necessary to insure that the IM procedure as previously defined is sufficient to produce all possible abstractions of a set of event representations. Furthermore, they clarify exactly what may be meant by a unit of information in the context of abstraction problems. Whenever the production of abstractions is henceforth required, it will be presumed that this is achieved by partial-matching the appropriate uniform representations.

On the other hand, additional alternatives exist for all-or-none matching and, in the next chapter, a completely different framework is considered. Interestingly, the three basic uniform representational units considered essential for partial-matching are seen to have conspicuous correspondents in the proposed network matching framework.

4.8 Feature Coding + Pattern Learning + Pattern Recognition + Consciousness = Perception

This chapter has addressed the nature and solution of the pattern learning and recognition problems. It is suggested here that these are only two of four essential elements constituting what is conventionally intended by the

word perception. The other elements are feature coding (discussed in section 3.5) and consciousness, the impression that one is aware of her or his self and environment. It has been my purpose to explicate those operations of perception, namely pattern recognition and learning, which are most general, best understood, and most exploitable in the solution of practical problems. Whether consciousness is necessary for effective, complex behavior is dubious and, in any case, there is nothing in the SR system which models it. (Although, if operationalized, the SR system could model it.) Thus, the principal deficiencies in the current paper as a functional treatment of perception, in my opinion, are the lack of operational analysis of feature coding and the all-or-none nature of the recognition process. Each of these is now briefly considered.

Feature coding per se is an extremely interesting problem, especially in light of the apparent necessity to produce structured representations such as those modeled here by PSRs. Unary content predicates, such as {p:red, what:x}, might be easily computed in artificial and natural systems by simple specialized receptors sensitive to specific energies. On the other hand, the detection of a line composed of a series of variably spaced collinear dots is not achievable by any such single, simple, local detector. Rather, this and numerous other trivial problems are extraordinarily difficult for conventional artificial devices (see, for discussion, Minsky & Papert, 1969).

While each special feature coding problem may be best accomplished by a "clever" ad hoc solution (e.g. the line analysis may be easily performed by an autocorrelation function if somehow the correct spacing parameters are known: see, for example, Uttal, 1973), it is not clear that the feature coding process will ultimately be seen to manifest a few, simple, general operations which may be abstracted from one or two areas and applied successfully to a variety of new problems, as is the case, for example, with the representation, learning, all-or-none, and partial matching operations which are also intrinsic to perception. The feature coding problem is further complicated by the apparent necessity for organisms to orient their receptors in particular ways and to perform other behaviors to "extract" particular features of interest. Explanations of these behaviors are beyond the scope of this paper.

The primary disadvantage of an all-or-none relational matching mechanism as the underlying basis for modeling natural perception is its inability to utilize in a straightforward way noisy or continuous distributions in the pattern recognition and learning processes. While it is true that any continuous variable or process may be modeled suitably by parameterized functions of some sort, it is both inconvenient and unsatisfying to do so. Rather, some additional techniques should be discovered to merge continuous and relational data in direct, desirable way. Some ideas have been suggested (see, for example, Zadeh, 1965; Fu,

1972; Hayes-Roth, 1973b; Thomason, 1973) which relate examples of noisy (fuzzy) patterns to a structured prototype, but all of the suggested mechanisms tend to expand greatly the number of alternatives which must be considered. For example, an IM procedure for less-than-total recognition based on partial-matches of constitutent relations would probably involve computing more features and comparing more relations than the corresponding all-or-none procedure. In addition, a less-than-total match would necessitate some determination of which aspects of similarity or difference were most significant, which least, etc. This is an important theoretical problem which should be investigated further in the future. Some speculations on natural bases for matching, which are relevant here, are considered in the last chapter. For the moment, however, it is suggested that all-or-none matching is a clearly important and useful subproblem, which provides an apparently adequate model of many complex perceptual and cognitive behaviors.

Chapter 5: DIRECT MEMORY CONTACT MATCHING FOR IMMEDIATE BEHAVIOR

5.1 Chapter Overview

The purpose of this chapter is to investigate, analytically, the plausibility of stimulus-response or stimulus-driven computation (SDC) models. Such models, although considered by some to be insufficient bases for general cognition, are easy to use and describe in the SR system and provide a computation universal (see Appendix IV) and stimulus-response basis for the representation of behavior. The central formal problem in any effort to realize such systems is deciding when a stimulus representation matches a contingency pattern or template. Although all-or-none matching can be effectively accomplished using the interference matching (IM) procedure (Appendix IV), a major unsolved problem impeding the adoption of SDC as a general behavior model is to show the feasibility of performing structured representation matching by some direct memory contact (associative, content addressed) process which obviates the intervening delays, serial computations, and vast amounts of temporary storage for intermediate results entailed by the IM procedure or other graph-matching techniques (Unger, 1964; Barrow, et. al., 1972; Corneil & Gottlieb, 1970; Rastall, 1969; Sussenguth, 1965).

A necessary and sufficient condition for the design of a structured associative (content addressable) memory

is presented which capacitates such a direct memory contact comparison of stimulus and template representations. The implications of this crtierion lead directly to the specification of the proposed SANER nets. These nets solve the recognition problem in a framework of homogeneous threshold logic elements, but can do so only if some stimulus representation components are permuted in specific ways.

An alternative memory design which enables direct memory contact matching is presented in which all stimulus subpatterns (abstractions) are realized separately as coded representations in canonical forms. Matching may then be performed by associative comparisons of the respective frequencies of any template canonical form and each stimulus subrepresentation canonical form. Finally, the impracticality of all such techniques as bases for the realization of large-scale artificial SDC systems is considered. It is suggested that, if SDC is a valid model of natural behavior, the information processing capacity of natural systems dwarfs that of artificial systems potentially realizable in the near future.

5.2 The Fundamental Stimulus-Response Idea: Stimulus-Driven Computation

The current chapter investigates the central questions which arise in an attempt to provide an SDC model for behavior. Although Hebb (1949) and other workers in cell assembly theory who succeeded him (Milner, 1957; Rochester, et

al., 1956; Good, 1965; Finley, 1967; Plum, 1972) argued that simple synaptic change rules could account for the learning of any arbitrary phase sequence of local firing pattterns and that this would be the basis for all acquired knowledge, there was never any precise attempt to explain in general how a distribution of firing patterns could represent a stimulus or a template and how their matching could be performed. Although the current paper is not a "cell assembly" model, its major questions are the same. Chief among these is the following immediate recognition problem: design a system in which all-or-none matching of a stimulus and template representation can be achieved immediately without the storage and subsequent manipulation of complicated intermediate results.

The reason for carefully investigating the immediate recognition problem is that SDC systems appear to be important models for a wide variety of behaviors which do not seem, introspectively, to be controlled by heuristic, higher-order, conscious, or problem solving procedures. Those behaviors which most would agree are "immediate" in this sense include walking or driving an extremely well known route, reading, understanding speech, recognizing faces or signs, dressing, an so forth. Furthermore, it is plausible (to me, at least) that all behavior is produced by specific responses to prior conditions and that all behavior is best modeled by corresponding contingency-respnse TXs. In any case, it would seem to be an important analyti-

cal task to establish the possibility that simple processing devices such as threshold logic units may be suitably organized to effect the immediate recognition of any structured event both because such devices are like those presumably operative in the corresponding natural recognition processes and because a demonstration of the feasibility of simple but powerful artificial SDC systems is the first step toward their development.

5.3 Recognition as a Problem of Graph Matching

Two problems of matching are paramount in general graph theory. One is to decide whether two graphs (each defined by its own set of nodes and a set of possibly differentially labeled and directed arcs connecting arbitrary pairs of its nodes) are identical. The second is to decide if one contains the other as a subgraph. The first problem is usually referred to as one of isomorphism or equivalence while the latter is one of monomorphism or covering. Every graph may be represented as a PSR in a direct manner: each node is assigned an arbitrary parameter name and each labeled or unlabeled arc is represented by a corresponding relation comprising a predicate, corresspondding to the arc label type, and two parameter objects identifying the nodes connected by that arc in the graph. The converse is also true (see Appendix IV). Thus, the problem of recognition of structured events is tantamount to that of identifying when one graph (corresponding to the

stimulus PSR) covers (contains, matches) another (the template PSR). Thus, any solution to the immediate recognition problem provides a solution for an important class of mathematical problems. The relationship of the canonical form isomorphism test used later in this chapter as a basis for one design for an SDC memory to other extant isomorphism procedures is discussed in Appendix III. The relationship between the IM procedure as a basis for deciding S(*)T and extant graph monomorphism procedures is discussed in Appendix IV.

5.4 Criteria for an SDC Memory

The first criterion which must presumably be met by any memory which can be a basis for solving the immediate recognition problem is that of canonicality: two events which are identical must have identical representations, and the determination of their identicalness should be achievable without complex procedures by a direct comparison of two homologous structures. Although this may sound obvious, the PSRs of the SR system which have heretofore been touted as desirable representational bases for so many purposes do not satisfy this criterion. An event or pattern may be represented in any number of distinct ways simply by choosing new parameter symbols each time. There is no easy way to establish the identicalness of two PSRs other than through the use of the IM procedure or a comparable graph matching procedure: if A(*)B and B(*)A

then A and B are equivalent, are representations of the same event.

In point of fact, it is possible to derive a canonical form for any PSR (or, a fortiori, any graph). The details of this result are presented in Appendix III. In brief, the canonical form is derived by determining a specific ordering of the parameter symbols which is dependent only upon the content and structure of the PSR. Because this ordering is unaffected by the particular parameter symbols used in the original PSR, the canonical PSR may be formed by replacing each original parameter symbol by a literal constant corresponding to its position in the derived ordering. Subsequently, the relations in the body of the PSR, actually strings of alphabetic symbols, are lexicographically ordered and concatenated. As a result, two representations are equivalent (two graphs are isomorphic) if and only if they have identical canonical PSR bodies, each considered as a simple string of alphabetic symbols.

Unfortunately, satisfying the canonicality criterion alone, as these canonical PSRs do, is not enough to make these representations desirable bases for constructing SDC systems. Later in this chapter (section 5.8), canonical PSRs are briefly reconsidered as bases for direct memory contact matching. For the present, however, they need not be considered because of their inability to satisfy the second and, in fact, principal proposed criterion.

The second condition which must be satisfied by any

candidate for an SDC memory is the criterion of subordinate congruence: if the pattern A contains the subpattern B, the representation of A should contain that of B as a literal and homologous substructure. This condition clearly cannot be satisfied by PSRs; the addition of any relations to the body of one canonical PSR will, in many cases, cause an alteration of the parameter orderings and, as a result, preclude literal string comparisons (or any other trivial operation) as a basis for determining whether two patterns match. If PSRs did satisfy the congruence condition, determination of A(*)B would be extraordinarily simple: whenever each component of B's representation is present in the stimulus representation A, B is matched and recognized.

It should be pointed out that the property of subordinate congruence logically implies (hence obviates) canonicality. If A and B are equivalent event representations, A(*)B and B(*)A, the representation of A contains that of B and vice versa. This could only be true if the two representations were identical. Thus, subordinate congruence is a very strong design criterion. It is for this reason that a solution to the canonicality problem for PSRs (Appendix III), although a useful mathematical theorem because it solves the equivalent graph matching problem, is useless with respect to the more general graph covering problem which is central to the task of realizing SDC systems.

In short, the design problem for SDC memories is com-

pletely specified by a single criterion: a subrepresentation of any containing representation must correspond to a subpattern of the associated pattern, and it must be a simple computational task (one which can be performed using any zero-delay--i.e., memory free--logic elements in a fixed computational network) to decide whether this relationship obtains for any template and possibly matching stimulus.

5.5 Implications of the Subordinate Congruence Memory Design Criterion

Two implications of the subordinate congruence criterion for the design of an SDC memory are immediate. First, all parameters must be removed from representations if it is to be possible to compare directly two representations, S and T where S(*)T, by simple computations on their components regardless of which arbitrary symbols are used as parameters in S and T.

The second implication is that all equivalent relations (same predicate and object types, regardless of object parameter symbols) must somehow occupy identical loci in the space of memory representational components. If this were not true, it would be necessary to compare (enumeratively) each relation of one type in one representation with each one of a same type in the other representation, as in the IM procedure. Such enumerative comparisons are inadequate for SDC memories because the congruence criterion requires that

comparable representations be matchable using a fixed network of elements. If each relation of S of the same type as one in T were enumeratively compared to it, the greater the number of matchable pairs of relations between the stimulus and template, the greater the number of comparisons (hence comparison elements) required to recognize S(*)T, violating the requirement for a fixed network to recognize T for all S.

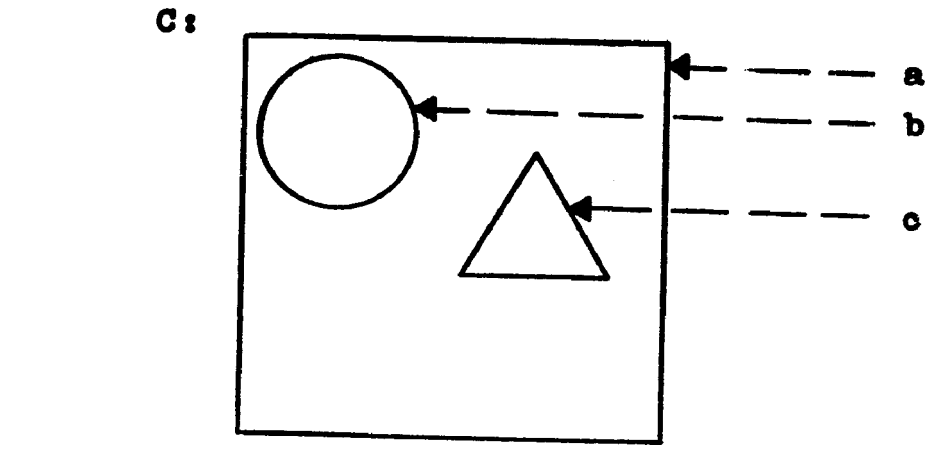
5.6 SANER Nets

One framework which satisfies the preceding criteria (and which is consistent with their implications) is provided by structured associative network event representation (SANER) nets (Hayes-Roth, 1974a). The SANER representation of any template pattern consists of a set of nodes, directed arcs, and associated node and arc component frequencies. Similarly, the SANER representation of any stimulus pattern is a net composed of the same types of elements. Moreover, SANER nets satisfy the subordinate congruence criterion and, as a result, guarantee that a stimulus matches a template in the all-or-none sense only if the SANER net contains the template representation as a subnet.

An explicit PSR (one whose objects are all parameters) is converted into a SANER net as follows. First, the body of the PSR is partitioned into equivalence classes of relations with identical predicates and, within each equivalence class, the constituent relations are arbitrarily indexed 1, 2, ... These indexes are called position indexes. Second, each distinct predicate object type is re-

presented by exactly one unique and correspondingly named node in the SANER net. Equivalently, the set of nodes for SANER nets comprises all distinct (predicate, object type) two-tuples. Third, a directed arc connects any two nodes representing objects occupied by the same parameter in the original PSR. The arc is directed from its tail node to its head node. The tail of the arc is the node whose name is lower in an arbitrary lexicographic ordering than is the name of the head. Fourth, each arc is position indexed by affixing a label of the sort [t,h] indicating that the object parameter giving rise to this arc occurred in the relation in position t in the equivalence class of relations containing the tail object and in the relation in position h in the class containing the head object. Each arc with a distinct position index label is considered a distinct type of arc. Fifth, the frequency with which any node or arc occurs in this net generation process is affixed to it by a frequency label, where a frequency count of k is denoted (k). Finally, every frequency and arc position label is prefixed by one symbol which represents the name of the event whose representation comprises these elements.

These operations are illustrated by the event C, its PSR, and its SANER net labeled Z in figure 11. There are five nodes in the representation of Z, corresponding to each distinct (predicate, object type) locus in which any of the parameters a, b, or c occur in the PSR. The dashed



parms: {a, b, c}

body:{{p:square, name:a}, {p:circle, name:b},

{p:triangle, name:c}, {p:contain, outer:a, inner:b},

{p:contain, outer:a, inner:c}}

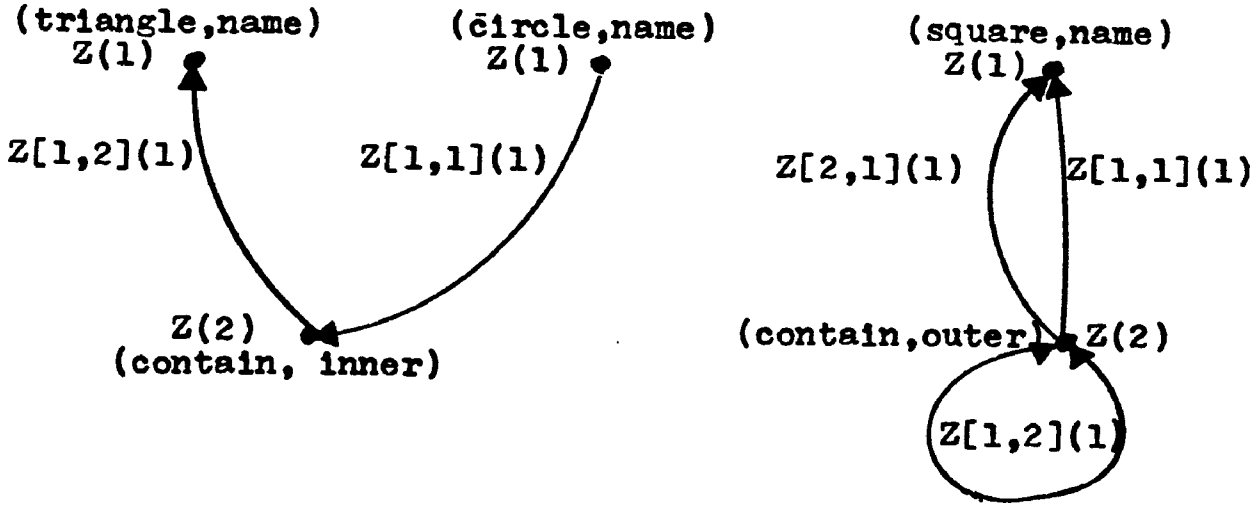


Figure 11. A PSR for the event C and a corresponding SANER net. The dashed lines in the PSR body indicate the structure of the PSR. If the actual parameter symbols are removed, the PSR is still interpretable.

lines in the PSR body, representing multiple occurrences of identical parameters, are translated into the directed arcs of the SANER net. Because these arcs maintain the structure of the PSR, there is no need for parameters in the SANER net. Each arc is directed by lexicographic order and position indexed to distinguish which original relations produced it. Note that lexicographically unorderable head/tail pairs (e.g. (contain, outer) in the first and second "contain" relations) are simply indicated as bidirectional arcs with the understanding that the corresponding [t,h] position index label is symmetric.

As a result of the criteria used for their construction, SANER nets exhibit the following important recognition condition: if the SANER representation (S) of one event contains all of the components (nodes, arcs) of another event representation (T) and the component frequency counts of the former representation are at least as large as those of the corresponding components in the latter representation, it follows that S matches T, S(*)T.

The proof of this statement derives directly from the fact that a PSR R*(R) corresponding to any SANER net N(R) of a PSR R can be deterministically constructed from N(R) by the following procedure. For each relation of predicate type Q of order n, let k(Q) be the frequency of nodes (Q,j) (j=1,...,n). For each i=1,...,k(Q), add the relation $\{p{:}Q,\ j{:}x_{i,j} : j=1,\ldots,n\}$ to the body of the new PSR R*(R). Then for each arc from tail (Q,j) to head (Q',j') with po-

sition index label [t,h], establish the equality $x_{h,j'} = x_{t,j'}$. Then it is clear that if the i-th and m-th relations of predicate type Q and Q' in R are {p:Q, j:$y_{i,j}$: j=1,...,n} and {p:Q', j':$y_{m,j'}$: j'=1,...,n'}, $k(Q) \geq i$, $k(Q') \geq m$, and if the lexicographic order of Q is not greater than that of Q', (for all j, j', i, m) $y_{i,j}=y_{m,j'} \Rightarrow$ there is an arc from (Q,j) to (Q',j') with position index label [i,m]. But this in turn implies that the relations {p:Q, j:$x_{i,j}$: j=1,...,n} and {p:Q, j':$x_{i,j'}$: j'=1, ...,n'} will be in R*(R) and $y_{i,j}=y_{m,j'} \Rightarrow x_{i,j}=x_{m,j'}$. Thus R*(R) = R. Then if the frequency of every arc and node of the SANER net N(T) of T is no greater than that of the corresponding element of N(S), it follows that R*(S) (*) R*(T).

The recognition that S(*)T may then be computed by a conjunction of the comparisons of corresponding S and T component frequencies. Furthermore, both the frequency comparisons and the overall conjunction of comparisons may be performed by simple zero-delay threshold gates: the threshold for a component comparison gate is set equal to the component's frequency in T and its momentary input is the frequency of that component in S; the output of each gate is 0 unless its input exceeds its threshold, in which case it is 1; and the overall conjunction is computed by summing the outputs of each component frequency comparison and comparing the sum (as input) to the maximum possible (as threshold). Thus SANER nets provide a memorial repre-

sentation in which the recognition of any structured event can be performed by a simple, direct memory contact, and immediate (zero-delay) process.

One complication which arises in the use of SANER nets for SDC remains to be mentioned. It was stated that a SANER net for a template was produced in a series of steps, the first of which was the arbitrary ordering and position indexing of relations in each predicate equivalence class. Because of the arbitrariness of this ordering, something must be done to insure that any potential stimulus is ordered in a comparable way to permit the satisfaction of the SANER recognition condition. One method is to generate one SANER subnet for each distinct set of permutations of the relations within each equivalence class of the stimulus PSR. Each such subnet is distinguished by appending an additional unique subnet name to the net name label. As a result of such stimulus coding procedures, the SANER recognition condition obviously becomes biconditional: a stimulus S matches a template T, S(*)T, if and only if one subnet of the SANER representation of S contains, in equal or greater corresponding frequencies, each component of the single subnet representation of T.

The similarity of SANER nets with previously studied SDC bases are many. For example, these nets use the same type of threshold gates as Pandemonium (Selfridge, 1959) and ostensibly serve some of the same functions, notably pattern recognition (see Chapter 4). Because they are ca-

pable of recognizing structured events, however, they are actually more similar to the logical computation nets of McCulloch and Pitts (1943; also, McCulloch, 1965). These nets are composed of homgeneous threshold devices too, but are associated with unit time delays in order to provide memory for the storage of intermediate results. Thus, while McCulloch-Pitts nets are capable of recognizing structured events, they require large time delays to do so. This was one of the major problems which the criteria for an SDC memory were proposed to avoid.

But SANER nets have a real problem which would impede their practical application: the permutation problem. The number of possible distinct orderings of relations within predicate equivalence classes is likely to be vast in real problem contexts. Yet, the encoding of a stimulus in terms of all possible permuted position indexed subnets is necessary if a stimulus and template are to be matched with certainty by direct memory contact and component frequency comparisons. Such a requirement makes the SANER nets very inelegant. One solution to this problem is provided in the next section.

5.7 Canonical Abstraction Powerset Representations

One way in which the permutation problem of SANER nets may be eliminated is to use canonical abstraction powerset representations in which the representation P(R) of any PSR R={p:event, parms:X, body:{R_1,...,R_N}} is defined as

follows. $P(R) = (A(R), F_R)$, where $A(R)$ is the set of all distinct canonical forms (CFs), $CF(R')$ of any subset R' of $\{R_1,\ldots,R_N\}$. F_R is the set of frequencies of each such CF, namely $F_R = \{F_R(r) : r \in A(R), F_R(r) = |\{R': R' \subset \{R_1, \ldots,R_N\} \& CF(R') = r\}| \}$. Using such powerset representations, it is again possible to decide if $S(*)T$ using only frequency comparisons of equivalent representation components. Specifically, $S(*)T \Leftrightarrow (\forall r \in A(T))\ F_T(r) \leq F_S(r)$. This is true because $S(*)T \Leftrightarrow (\exists S'$, a subset of relations in S) $CF(S') = CF(T) \Leftrightarrow$ there is some correspondence between the parameters of R and S' such that every relation in T corresponds identically to one in S' and, hence, to one in $S \Leftrightarrow (\forall T'$, a subset of relations in T) there exists a corresponding subset of relations S" in S such that $CF(T') = CF(S")$.

In fact, it is possible to economize somewhat on the number of elements in P(R) whenever there are relations in R which are not connected. Any set of relations $\{R_1,\ldots, R_N\}$ is connected if $(\forall j=1,\ldots,N-1)$ there is at least one parameter in R_j which also occurs in R_{j+1} (for some ordering of the N relations). If a set of relations is not connected, it may be partitioned into subsets of relations which are connected and for which it is true that if two relations are connected they are in the same subset. Then each connected subset represents a pattern which is expressed as a set of relations over a distinct set of parameters (no relations in two distinct subsets have any

object parameters in common). If each such maximally connected subset $V_1,\ldots,V_k$ of the relations in T is such that $S(*)V_i$ $(i=1,\ldots,k)$, it is true that S(*)T (because every relation in T has a correspondent in S under some parameter correspondences $\{[T.t_j, S.s_j]: j=1,\ldots,n_i\}$ arising from the matching of S and V_i and the union of these correspondences is consistent, each parameter t_j in T is assigned at most one correspondent s_k in S).

These observations mean that the <u>minimal canonical abstraction powerset representation</u> $P^*(R) = (M(R), F_R)$ of R can be defined with $M(R) = \{r: r \in A(R)$ and the relations in r are connected$\}$ and that it is also true that $S(*)T \Leftrightarrow (\forall r \in M(T))\ F_T(r) \leq F_S(r)$. The proof of this claim follows from the fact that for any T, there is a set of maximally connected subsets $V_1,\ldots,V_k$ which constitute a partition of the relations of T and that $(\forall i=1,\ldots,k)$ $F_T(CF(V_i)) \leq F_S(CF(V_i)) \Rightarrow \exists W_{i_1},\ldots,W_{i_{F_S(CF(V_i))}}$, distinct subsets of relations in S) $(\forall j=1,\ldots,F_S(CF(V_i)))$ $CF(V_i) = CF(W_{i_j}) \Rightarrow (\forall i=1,\ldots,k)$ $S(*)V_i$. Thus, S(*)T.

As a result, if S and T are coded in their minimal canonical abstraction powerset representations, S(*)T if and only if the frequency of any subpattern T' in T with CF(T')=r, $F_T(r)$, is less than or equal to the corresponding frequency in S. Thus, S(*)T can be computed by comparing (with a set of threshold elements) the number of times any particular abstraction r occurs in S (as input) to the

the number of times it occurs in T (as threshold) and conjoining the results of all comparisons for every abstraction (subpattern) in T. Because each abstraction corresponds to a fixed address in memory (its CF), S(*)T is a simple computation as desired.

5.8 Conclusions: Powersets and Matching

The number of distinct frequencies apparently required for direct memory contact matching for deciding if S(*)T is of the order of the powerset of (connected) abstractions of T. If T contained only 30 relations, the cardinality of this set would be about 10^{10}, approximately equal to the number of neurons in the human brain. It would seem then that direct memory contact matching of structured events requires an enormous amount of representational complexity (numerous components) to enable simple computations to be effective. Such a capacity is probably out of the reach of potentially available artificial systems.

In short, although this analysis of direct memory contact recognition processes has successfully demonstrated the possibility of their existence, the questions which remain to be answered are no less difficult: How can a natural nervous system accomplish SDC and how can an artificial SDC system be constructed without exceeding all practical physical limits. Unfortunately, these problems are not solved in this paper. Some related speculations

are, however, ventured in the final chapter.

Chapter 6: SUBSTITUTION TRANSFORMATIONAL BEHAVIOR AND ITS ACQUISITION

6.1 Chapter Overview

Substitution transformations are contingency-response rules that correspond to universally quantified implications in first-order predicate calculus. For example, the rule "for all x, if x wears red suspenders assert x is a fireperson" in the presence of the given information "Sally wears red suspenders" is a basis for the substitution of "Sally" for "x" and the assertion, "Sally is a fireperson." The contingency of a corresponding substitution TX in the SR system would simply be: {{p:wear, who:x, what:y}, {p:suspenders, name:y}, {p:red, name:y}}, where x is a parameter. The associated response would be: {{p:fireperson, name:x}}. As previously explained (section 3.4), the presence of the message {{p:wear, who:Sally, what:z}, {p:suspenders, name:z}, {p:red, name:z}} in STM should cause the SR system to place the desired response in STM: {{p:fireperson, name:Sally}}.

In this chapter, the generality of such substitution rule computing is illustrated by drawing examples from transformational grammar. It is shown how readily such problems yield to representation in the SR framework. Then, the problem of inducing such grammatical rules from training information is formalized and solved using a rule learning procedure based entirely on the abstractions produced by

the interference matching (IM) procedure (section 4.5).

6.2 Transformational Grammar as a Typical Problem

There are two basic components in the usual view of transformational grammar (Chomsky, 1965, 1967, 1968; Langendoen, 1969; Friedman, 1971), a phrase tree representation of some "deep structure" which is the conception underlying the eventually produced sentence, and a set of possibly non-deterministic alternative transformational rules which systematically alter the deep structure until a final surface phrase structure tree is produced in which the terminal leaves are the words spoken or written, in their proper serial order. Examples of these concepts are given in figures 12 - 14. Figure 12 illustrates a simple deep structure phrase tree for the idea, "the boy hugged the dog belonging to the boy (himself)". Figure 13 illustrates the general active-to-passive rule by which any sentence manifesting the appropriate active form deep structure can be converted to a corresponding passive form and its use to produce "the dog belonging to the boy was hugged by the boy." Similarly, figure 14 illustrates an obligatory pronominalization rule by which the structure in figure 12 is converted to one which would be spoken as "the boy hugged his dog."

The reason for introducing these rules is to illustrate the principal features of such transformations. First, the contingency for computation is described by some

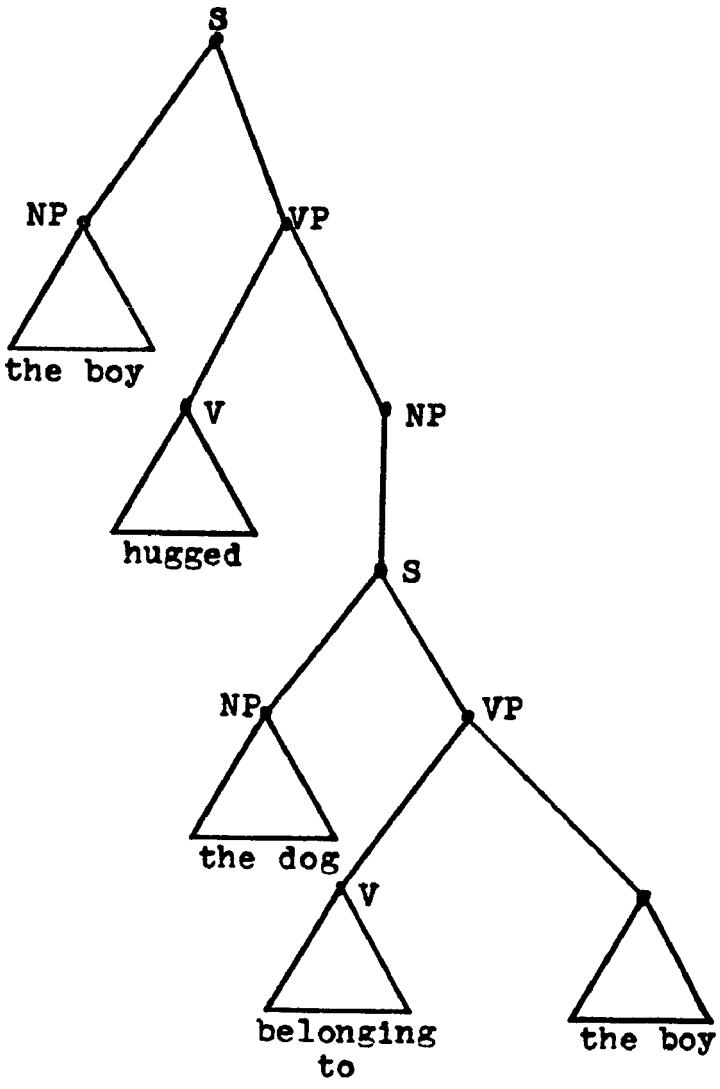


Figure 12. A deep structure for the idea, "the boy hugged the dog belonging to the boy (himself)."

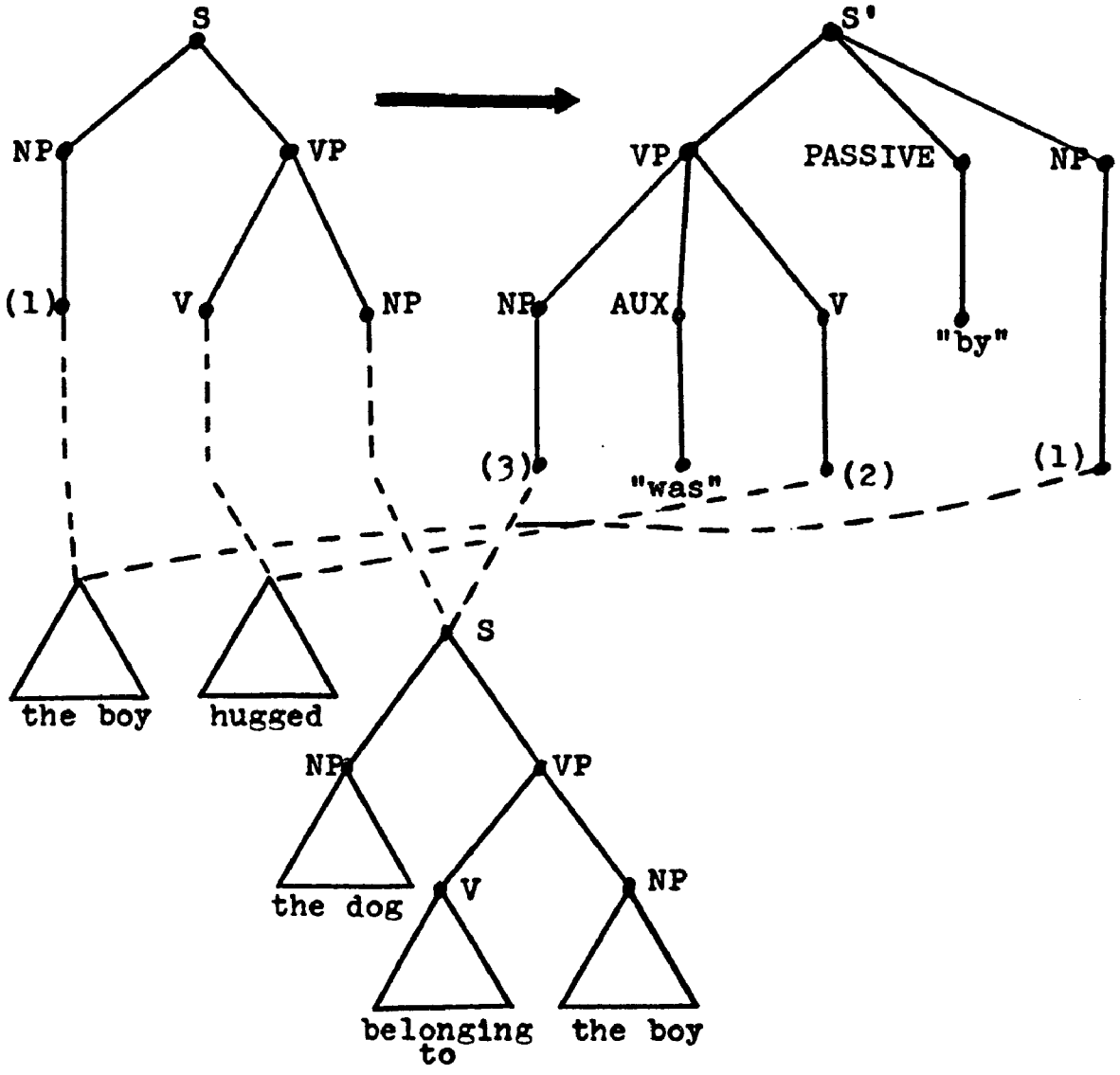


Figure 13. An active-to-passive transformation and its use in the conversion of the sentence, "the boy hugged the dog belonging to the boy," to the sentence, "the dog belonging to the boy was hugged by the boy."

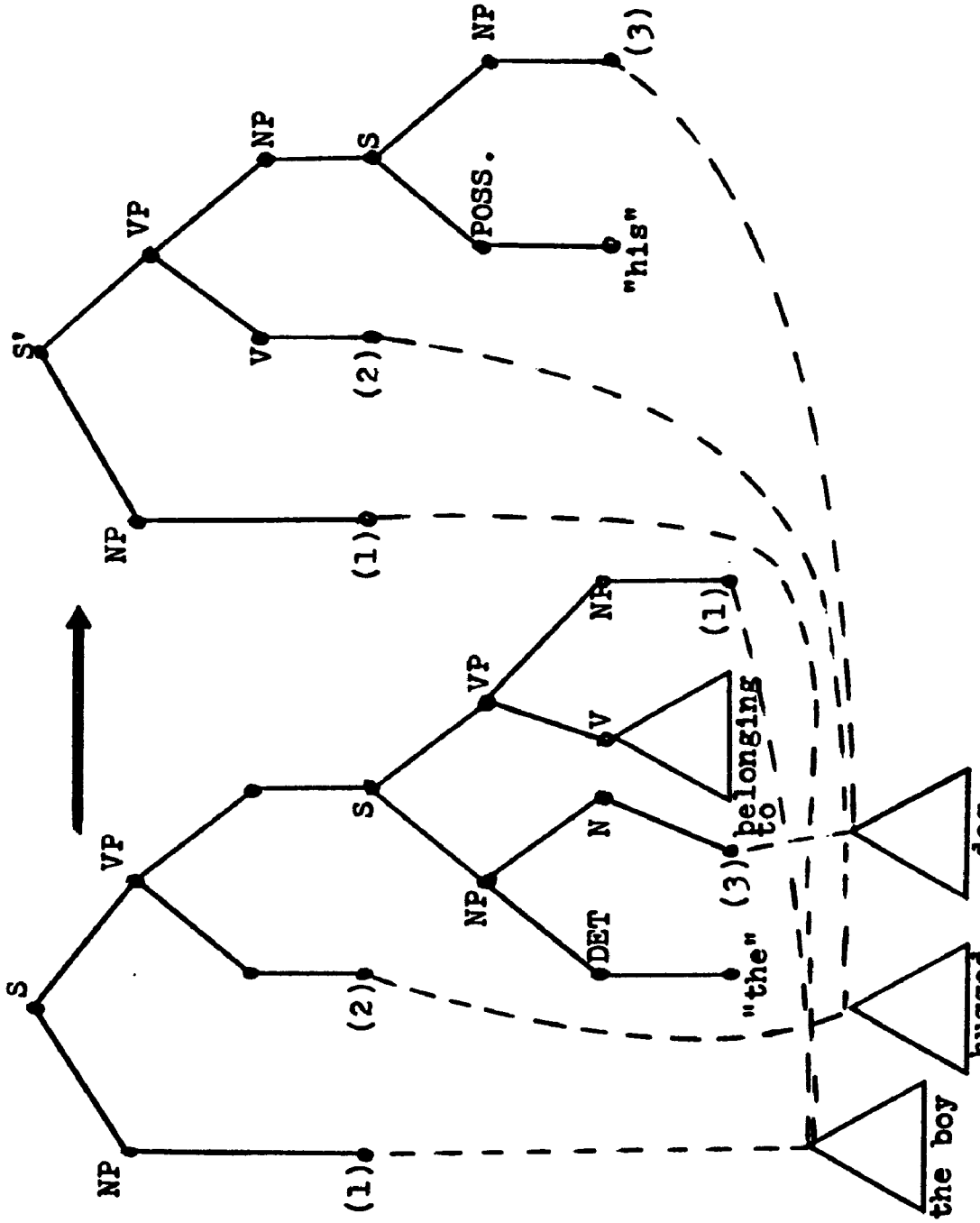


Figure 14. A pronominalization transformation and its use to introduce the possessive pronoun "his" in place of "belonging to [the subject, NP (1)]".

schema, partially in terms of unbound variables, indicated by (1), (2) and (3) in figures 13 and 14, which are supposed to take the specific values arising in the corresponding loci of the stimulus pattern. These values are then substituted into the appropriate positions in the response patterns. There is no outstanding reason why phrase structure trees are used to represent the content of these sentences. Rather any relational structure whatever may be used. The important thing is that the matching of the stimulus and contingency patterns be determined in some way which permits the binding of corresponding stimulus pattern values to variables which are then appropriately substituted into response patterns. Whether or not transformational grammars are valid models of natural linguisitc behavior is unknown, but they typify the sort of pattern matching and variable binding that is widely employed in modern cognitive simulations and artificial intelligence programs (Fikes & Nilsson, 1971; Hewitt, 1972; Rulifson, et al., 1972; Winograd, 1972; Becker, 1973; Reddy, et al., 1973). In the remainder of this chapter, such rules are first given precise SR representations and, second, it is shown how these representations are easily abstracted from appropriate training information.

6.3 Representing Transformational Rules in the SR System

Earlier, in section 3.4, a more complicated version of an active-to-passive transformation was provided than is

needed for the current purposes. Thus, the following simpler contingency-response TX will be used here as a formal description of the rule in figure 13.

```
A-P:{p:TX,
     parms:{s,npl,vpl,v2,np3,vp2,byname,wasname},
     C:{{p:sentence, agent:npl, verb:vpl, name:s},
        {p:verb phrase, verb:v2, object:np3, name:vpl},
        {p:sequence, 1:start, 2:npl, 3:null, 4:v2,
         5:null, 6:np3, 7:end}},
     R:{{p:sentence, agent:npl, verb:vp2, passive:
         byname, name:s'},
        {p:verb phrase, verb:v2, object:np3, aux:
         wasname, name:vp2},
        {p:"was", name:wasname}, {p:"by", name:byname},
        {p:sequence, 1:start, 2:np3, 3:null, 4:wasname,
         5:null, 6:v2, 7:null, 8:byname, 9:null,
         10:npl, 11:end}}}.                          (6.1)
```

The rule A-P is invoked whenever any sentence structure is present in STM which matches its contingency. When this is the case (see chapter 4), one parameter in the stimulus will correspond to each parameter in the contingency template and its name will be substituted into the appropriate response loci. Thus (6.1) is a formal description of the rule illustrated in figure 12. This rule is in explicit form, where every predicate object is a parameter (e.g. wasname is the parameter corresponding to the word

"was"). Because it is clear how other transformational grammar rules would be similarly represented as SR TXs, the problem of learning such rules is considered now without further delay.

6.4 The Learning Problem: Grammatical Inference

The desirability of the SR system as a representational base for the analysis of cognitive behaviors is due in part to the tractability it affords to the problem of grammatical inference. While other researches seem hindered by awkward formalizations of the grammatical inference problem or study heuristic procedures for its solution (Solomonoff, 1959; Biermann & Feldman, 1972; Siklossy, 1972), the previously suggested framework for the pattern learning problem (section 4.5) provides a readily solvable and general statement of the grammatical inference problem. Recalling the essentials of that framework, it is necessary to provide a training sample I of associated stimulus-response two-tuples, (S_1,R_1), (S_2,R_2), ..., (S_n, R_n), and some criteria for evaluating the performance of inferred rules. Little will be said here about performance criteria since the discussion in section 4.5 is still relevant and is directly generalizable from the categorical response alternatives previously considered to active, structured response alternatives when the latter are viewed as independent categorical alternatives. That is, since each possible response body is uniquely determined upto

equivalence (Appendix III), whether or not the response involves substitutions, each distinct response may be considered a categorical alternative and competition may be controlled according to previously described performance criteria (see end of Appendix IV for details).

In the current problem, then, the crucial question is how one might abstract rules from the (S_i,R_i) pairs (i = 1, 2, ..., n) which would, in effect, actually be capable of transforming future stimuli (active transitive sentences) in appropriate ways (converting them to corresponding passive sentences). One simple method is proposed here which, it can be seen, solves this exemplary problem and all others in the framework of transformational grammar. Moreover, the proposed solution is equally suitable for all computational learning problems when they are posed in the aforementioned format (proved in last section of Appendix IV).

There are two successive steps to the rule learning procedure: prediction-matching between corresponding stimulus and response representations and, subsequently, generalization-matching over all (S_i,R_i) elements in the training set I. Each of these steps is understandable in terms of the IM procedure of Chapter 4 and is discussed in turn.

Prediction-matching: the first step in rule learning

Each item in the training set is a pair of related stimulus and response information structures which, by definition, are associated because a common transformation has been applied to the former to produce the latter.

Prediction-matching is the process of partial-matching these two structures (using, for example, the IM procedure) to produce a model which identifies their commonalities (an abstraction) and differences(residuals). Such matching is called prediction matching because it is assumed that some aspects of the stimulus representation are sufficient to predict some aspects of the associated response representation.

Consider any model of the pair $\{S_i, R_i\}$ for a particular i. The related abstraction of $\{S_i, R_i\}$ describes ways in which structured relations in S_i can also be seen to obtain in R_i under appropriate parameter correspondences between the two representations. It is precisely these correspondences which are the basis for inferring variable substitutions. That is, if any active and passive forms of the same deep structure were represented by PSRs, as in the contingency and response bodies of (6.1), the most informative abstraction that could be produced would result from the correspondence binding of the first noun phrase in one sentence with the second noun phrase in the other and vice versa and from the binding of the parameters naming corresponding verb words in the two sentences. Under such bindings, the produced abstraction would reflect all that was common to the two grammatical structures. The residuals, on the other hand, would reflect properties of the sentence which were specific to the alternative voices, active and passive, of the two related sentences. Later, those corre-

spondence bindings identified at this stage which survive the second phase of rule learning will play the exact substitutional role of parameters repeated in both the contingency and response components of a TX.

To complete the first step in rule learning, all that is necessary is to rename the original parameters in both sentences so that bound correspondents in both sentences are represented by the same symbol. If the best model M_i of the pair $\{S_i, R_i\}$ entails the correspondence bindings $C_i = \{\{S_i.p_1, R_i.q_1\}, \{S_i.p_2, R_i.q_2\}, \ldots, \{S_i.p_m, R_i.q_m\}\}$ of the parameter p_j from S_i with the parameter q_j from R_i (for each j), let $S_i(C_i)$ and $R_i(C_i)$ represent the original PSRs S_i and R_i, respectively, modified so that correspondents in C_i are renamed identically in the two PSRs. Thus, prediction matching converts the original training set $I = \{(S_i, R_i) : i = 1, 2, \ldots, n\}$ into a new set $I' = \{(S_i(C_i), R_i(C_i)) : i = 1, 2, \ldots, n\}$. Actually, this process must be generalized to account for alternative rule possibilities by incorporating not just "the best" abstraction matches between each stimulus and response structure, but all of them which are produced by partial-matching. For the current purposes of exposition, however, this is a numerical complication which adds nothing essential to the central developments and is, therefore, ignored.

Generalization-matching: the second step in rule learning

While prediction-matching produces models between sti-

mulus-response pairs in order to relate similar substructures by identifying corresponding parameters, generalization-matching is the process of abstracting common stimulus, response, and stimulus-to-response correspondences from several training exemplars. Specifically, each renamed stimulus-response pair, $(S_1(C_1), R_1(C_1))$, is compared to every other one in I' in order to produce a common abstraction of the entire set of two-tuples. Using the IM procedure, this is straightforward. Each renamed two-tuple is matched with the others, insuring that stimuli are matched to stimuli and, simultaneously, the corresponding response structures are matched to one another. The derived maximal abstraction will exhibit the following properties: (1) parameters representing substitution variables in the underlying TX will be associated in the abstraction with simultaneous occurrences in both the stimulus and response components; (2) the complete structural representation of the contingency component of the desired TX will be manifested by the "generalized" abstraction of the renamed stimulus components; (3) similarly, the abstraction of the renamed response components will constituted the desired TX response; and (4) the residuals produced by generalization-matching, representing specific word differences in the case where exemplars are different sentence structures for the same grammar rule, describe differences among the multiple training examples which are irrelevant to the desired rule.

For example, if "the boy hugged the dog belonging to the boy" and "a girl spanked a cat" were two stimulus components in the training set for the active-to-passive TX induction problem, the residuals would include the following pairs of unmatchable phrase structures named by corresponding parameters: "the boy" and "a girl"; "hugged" and "spanked"; "the dog belonging to the boy" and "a cat." In Chapter 8 the important role of these associated residuals in the acquisition of syntactic and semantic categories is discussed. For the present, however, these residuals may be completely ignored; the desired transformational rule has been identified by abstraction.

Schematically, if the set of abstractions of any set E of PSRs E_1, E_2, ..., E_m is denoted $E^* = E_1 * E_2 * \cdots * E_m$, the training set for a rule induction problem is $I = \{(S_i, R_i) : i = 1, 2, \ldots, n\}$, and the set of renamed exemplars is denoted $I' = \{(S_i(C_i), R_i(C_i)): i = 1, 2, \ldots, n\}$, the desired TX is identified by partial-matching I' to produce, simultaneously, abstractions $U = S_1(C_1) * S_2(C_2) * \cdots * S_n(C_n)$ and $V = R_1(C_1) * R_2(C_2) * \cdots * R_n(C_n)$ under the overall parameter correspondence bindings B and, in specific, the TX is defined simply as: {p:TX, parms:B, C:U, R:V}.

6.5 Generality of the Prediction-Plus-Generalization-Matching Procedure

The two-step prediction-matching plus generalization-

matching technique is a completely general rule learning procedure. It requires only that a desired TX be representable in terms of some aspects of relations that are evident within the training information. This reflects an important difference in approach between the current paper and previous efforts at grammatical inference (see, for example, Biermann & Feldman, 1972) in which the only information provided to the learner are the sequences of terminal nodes (words) of the deep or surface structure tree. In such a case, the best one can do is essentially guess production rule grammars which hopefully approximate the sequential statistical constraints among the words.

On the other hand, the approach suggested here practically "eliminates" the problem by showing that if, in addition to the serial order of the words, the deep structure of each sentence is provided, a remarkably straightforward two-step matching procedure is adequate for rule learning. Nevertheless, I would argue that this approach is the only reasonable one. First, it is an approach which will work even in very complex grammatical problems where the alternative surely will not. Second, knowledge of the necessary structuring relations must, logically, precede their introduction into more complex organizing rules like those being considered here. Thus, it would be ill-advised for either natural or artificial learners to fail to exploit such knowledge. Finally, the automatic introduction or activation of such structuring relations upon the recogni-

tion of various stimulus words and their syntactic and semantic roles is assumed to be a principal feature of SDC in intelligent systems. Given the view of learning expressed at the outset of this paper, then, categorical classifications, semantic relationships, and grammatical structure which are part of the momentary state of the organism are grist for the learning mill. Any arbitrary distinction between what is observable and what is not, when both are essential components of the same computational behavior, is both unnecessary and obstructive in the analysis of computational development.

At this point, it is my intention only to clarify the assumptions that underlie the proposed approach to learning. Later, in Chapters 7 and 8, the "structuring information" which has heretofore been supposed as exogenously supplied to the learning procedures, will be considered directly as <u>supplementary information</u> which is both automatically retrieved during the course of related cognitive processing and automatically stored during generalization-matching and rule learning. It should be noted here, however, that while the proposed two-step rule learning algorithm will identify any rule which is manifested by appropriate training information (proved in Appendix IV), a very large amount of computation may be required. Moreover, if the learner itself must construct the training sets for each possible rule, a large number of alternative sets may have to be analyzed before finding the right one. These represent serious

impediments to the immediate and unlimited use of the proposed rule learning procedures in unstructured problem contexts.

Chapter 7: PROGRAMMATIC BEHAVIOR AND ACQUISITON: LANGUAGE LEARNING

7.1 Chapter Overview

Previous approaches to modelling natural language abilities have been successful when top-down grammatical parsers, sufficiently augmented by semantic analysis and problem solving capabilities, have been used in very restricted domains of discourse. The two most successful top-down methods (Woods & Kaplan, 1971; Winograd, 1972) are briefly reviewed and compared to the hybrid (top-down and bottom-up) approach to language understanding proposed by Schank (1972) and Reddy, et al.(1973). It is suggested that the major advantages of each of these approaches may also be obtained within a completely stimulus-driven (bottom-up) language understanding system which, in addition, affords advantages not readily derivable in other frameworks, notably learnability, extensibility, and error tolerability. The suitability of the two-step rule learning procedure described in the previous chapter for general language learning problems is also discussed. Because the SR system is computation universal (App. IV), the results in this chapter can be generalized to any programmable (effective) behavior.

7.2 Outstanding Top-Down and Hybrid Language Understanders

Surely the success of Winograd (1972) and Woods & Kaplan (1971) at the construction of effective computational models of natural language processes is one of the major

accomplishments of behavioral science to date. Even if these computer programs ultimately fail to predict useful human behavior, they provide the only completely operationalized theories for one of the highest forms of cognitive activity, linguistic and thinking behavior. It is very unlikely, however, that either program is a valid model in its details, if only because each is seriously dependent on the data storage and manipulation conventions of the computer languages in which they are constructed. On the other hand, the functions which these programs perform provide a useful benchmark for alternative theories of language understanding.

Both systems considered here operate in remarkably similar ways. In each case, a recursive grammar is used to guide the left-to-right, top-down parsing of the input sentence. The ultimate test of whether or not the sentence has been "understood" comprises two conditions: (1) have the rules of grammar been followed in reproducing the sentence and (2) is the semantic interpretation of the sentence associated with the chosen grammar rules "consistent" with what is known about the objects being discussed. The major differences between the Winograd and Woods & Kaplan system are ones primarily of style: how to represent the grammar, how to construct and represent the semantic interpretation, how to verify an assertion or answer an interrogative, and how to interleave the computation of related syntactic and semantic processes.

The chief characteristics of such top-down systems are

their complete dependence upon grammaticality, their enumerative approach to generating plausible underlying syntactic structures and, hence, their serious degradation under any substantial expansions of the grammar or possible semantic interpretations. For example, a sentence like "pick please red ball up," being ungrammatical, could only be properly parsed if rules of grammar were added which specifically interpreted such types of grammatical irregularity. Moreover, while any particular syntactic or semantic possibility may be handled by some sort of programming modification (a "demon" or "glitch"), as the number of such irregularities increases, the workability of the total system declines exponentially.

It is for these and other reasons that other researchers have pursued the development of hybrid language understanders. These systems restrict the use of top-down analysis methods to guiding local searches for words or phrases many of whose features have been "intelligently guessed" on the basis of previously recognized properties of other words, notably principal verbs and nouns. For example, Schank (1972) describes a system of conceptual dependencies which predict the types of agents, instruments, objects, recipients, etc. that are likely candidates to occur in the same sentence as a verb of a particular type. In fact, his proposed method for language understanding requires a preliminary identification of the main verb and a determination of its potential conceptual dependencies (a

bottom-up process) followed by a search for correspondents in the sentence for each of the conceptual slots remaining to be filled (a top-down process). As an added benefit, such an analysis provides a basis for predicting and understanding related discourse because of its underlying systematic conceptual basis.

The HEARSAY speech understanding system (Reddy, et al., 1973) reflects many of these same ideas but operates in a problem context, natural human speech, which is much less structured and uniform than that of typewritten input. As a result of the vast uncertainties and "errors" in speech (e.g. omitted words or phonemes, phonetic liasons, stuttering, non-grammatical utterances), this system has been designed to separate each potential source of linguistic knowledge and to allow each of these to participate in whatever way possible in the generation of "intelligent guesses" about other components of the related discourse. For example, a spectral analysis of one time segment of speech may lead to numerous hypotheses about the possible words that the speaker intended; each of these words may, via "conceptual dependencies," entail hypotheses about the adjacent words; and so forth.

In both of these hybrid systems the central "hypothesize-and-test" paradigm is paramount: given some information about the nature of the input, guess other features of the same sentence based on learned statistical regularities, try to affirm these by direct analytical computation,

and reiterate. Although it is possible that such an organization may be optimally effective in the design of a digital computer program, this is a strong and undemonstrated claim. Moreover, there is one principally awkward aspect of such system designs. At any point in time the information available would support vast numbers of reasonably specific conjectures about the surface structure of a sentence. Many of these are probably equivalent paraphrases (Reddy, et al., 1973). Thus, the hypothesize-and-test paradigm inevitably leads to the generation and separate evaluation of numerous but essentially redundant alternatives. One radical approach toward the elimination of such difficulties is to eliminate completely the top-down ("-and-test") aspect of such hybrid systems. If this is done, what remains is an SDC system with automatic invocation of TXs whose responses augment the current information base by supplying additional features of the speech which are, more or less, logically indicated by the current information. Such a bottom-up system for understanding is sketched in section 7.4 after the more typical top-down analysis is described in the context of an illustrative example.

7.3 Top-Down Understanding: "Pick up a big red block"

As the central example of language understanding, I consider the well-worn, "Pick up a big red block," from Winograd (1972, section 5.3.1). By serially checking each of the possible grammatical structures which might produce

this surface structure, the parse tree shown in figure 15 is ultimately generated. Although this simple example fails to suggest the breadth and power of such artificial understanding systems, it is quite satisfactory to illustrate how they work. Each component of the parse tree represents some information about the sentence which is added by one particular subroutine which succeeds in finding the type of word group it is looking for at that position in the tree.

The parser begins by looking for a clause at the leftmost work ("pick") of the input. Although clauses may have a variety of alternative structures, one possibility is that it will manifest the particular sequential structure: a verb group consisting of a transitive main verb which takes a particle (e.g. "up"), followed by a particle, followed by exactly one noun group representing the object of the verb. Because this is one of the possibilities and each possibility is considered until an inconsistency arises, each of these hypothesized clause component structures is sought in turn.

Before anything specific is known, a verb group is sought at the start of the sentence. Because some verb groups begin with an infinitival form of the main verb, this possibility is checked. "Pick" happens to be such a verb and, as a result, a syntactic structure is generated which reflects the possibility that "pick" is the infinitive verb word of a verb group which is the first constitu-

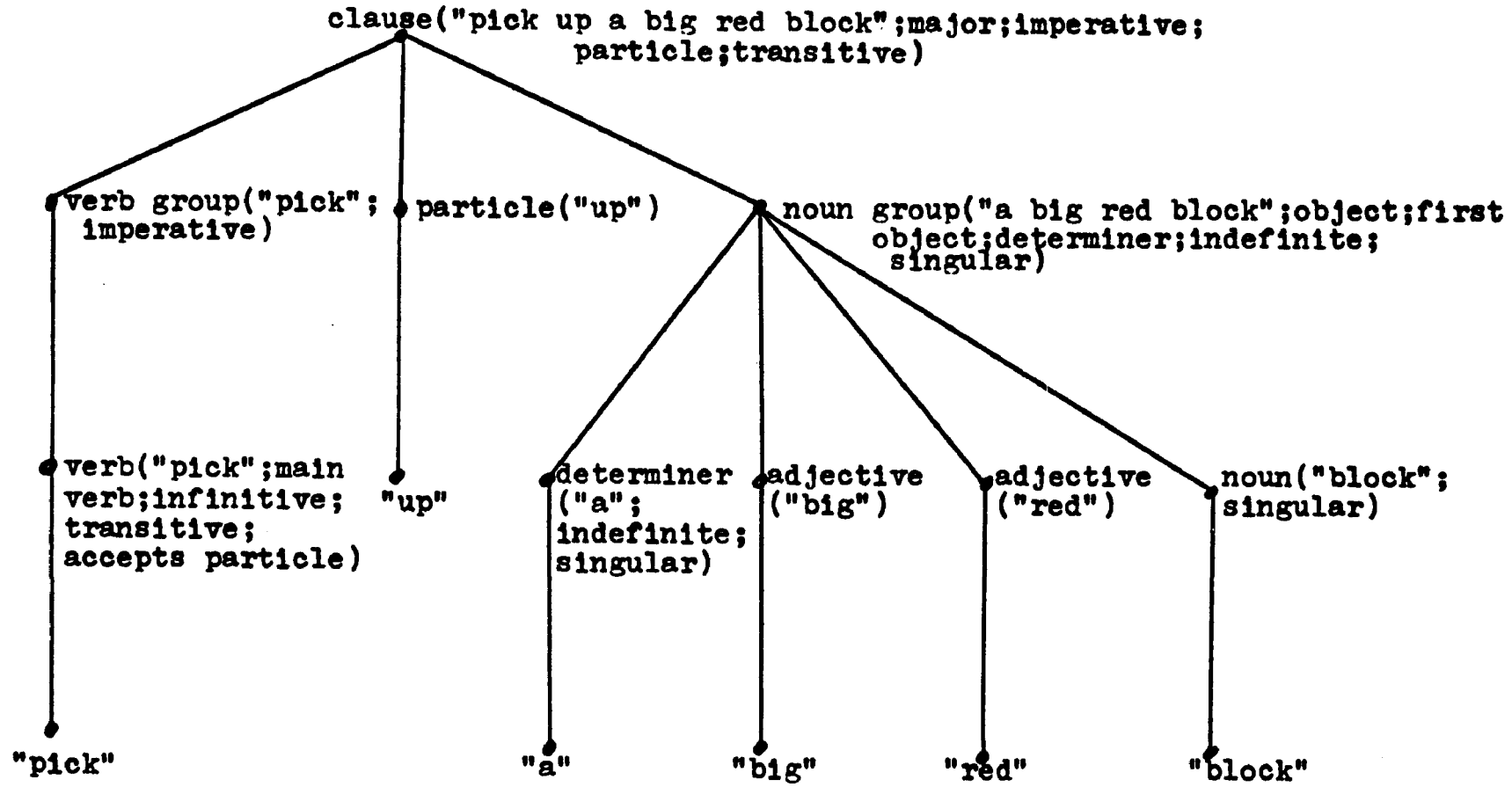


Figure 15. A parse tree for "Pick up a big red block," after Winograd (1972, section 5.3.1). Features of nodes appear in parentheses.

ent of the hypothesized clause. Such a possibility is reflected by the constituency arcs in the tree of figure 15 and, in addition, by the appropriate syntactic markers in parentheses at each node. For example, the marker ("pick") at the node named verb represents the fact that the extent (terminal leaves) of the subtree originating at that node is the sequence of one word, "pick." Similarly, the marker (main verb) at that node reflects the hypothesized role of the corresponding extent ("pick") within the total verb group.

One way in which top-down systems like this invariably behave in a bottom-up manner is by the automatic addition to the parse tree of semantic features (e.g. transitive, accepts particle) of constituent words (e.g. "pick") which are taken directly from a lexical table or "dictionary." Although this generation of markings may seem trivial, it reflects the same type of predictive analysis, albeit simpler, that underlies the bottom-up activities of the hybrid systems. That is, information about some substructure in the sentence is judged sufficient to cause the automatic addition of supplementary information which later acts to limit or direct the search for related possibilities. For example, the addition of the feature (accepts particle) to the verb node, causes the clause program to look for a successor particle word ("up"). Later, I will show that the addition of such supplementary information is the essential computational device in effective, exclusively bottom-up,

systems.

The entire parse tree of figure 15 is produced by programs that execute in a corresponding left-to-right, depth-first manner. Specifically, each node in the tree is produced by a program which successively calls subroutines whose functions correspond to the parts of speech represented by that node's subordinates, one after another in left-to-right order. As each subroutine regains control from its subordinates, it passes desired semantic and syntactic markers up the tree to its caller. As a result, the parse is completed at the same time the noun specialist returns (upon identifying "block" as a noun) to the noun group specialist which returns to the clause specialist which determines that the entire input sentence has been grammatically parsed (accepted).

At the same time this syntactic parse tree is developed, corresponding semantic structures are produced. Because they are not of central importance here, it suffices to say that the associated semantic description is a command to the robot's effectors to do anything it can to grasp and subsequently levitate ("pick up") any object x as long as it is known that x is a block, x is red, and x is big. While the production of such semantic interpretations is interesting in its own right, it adds little to an appreciation of the relative merits of alternative computational organizations or learning. In the next section, the parsing structure in figure 15 is produced by a com-

pletely stimulus-driven process to illustrate the generality of SDC systems and their ready applicability to such problems.

7.4 SDC as a Natural, Efficient, Learnable Basis for Bottom-up Understanding

Representation of a bottom-up understander

In order to expedite the presentation here, I will exploit a few shorthand notations in place of precise uniform PSRs. It should be understood, however, that the underlying formal representation is that of the SR system previously described. In particular, a sequence pattern like (... x y ... z) will be used to mean that two words or higher order phrase structures named by the parameters x and y occurred in that sequential order, preceded and succeeded by anything, and that some other structure of words named by the parameter z occurred directly at the end of the same sentence. Instead of any parameter in a sequence pattern, a specific literal (e.g., "boy") or another sequence pattern may be used with the obvious interpretations. The presence or absence of markers may be appended to any of these constituents by succeeding them by "+" or "-" signs and the corresponding feature names enclosed by slashes.

For example, the notation (... x/+feature1-feature2/ (y/+feature3/ (z1 ... z2/+feature3/))...) will be interpreted as follows: the sentence contains at least one phrase structure, called x, marked as exhibiting a property called

featurel and marked as not exhibiting feature 2; the phrase group x comprises exactly one subordinate, y, which is marked as exhibiting feature 3; the group y comprises at least two subordinates, z1 at the start and z2 at the end, with z2 exhibiting feature3.

Finally contingency-response TXs will simply be represented by the schema C➔R with the usual interpretation that when the current contents of STM match C, the structures specified by R are added to STM. Using this greatly abbreviated notation, the task of writing a set of TXs to parse the exemplary sentence is practically trivial. Assume that the result of stimulus encoding at time t_o is to place the following parameterized relations (the parameter sets are omitted for clarity) in STM:

STM(t_o) = {{p:"pick", name:w1}, {p:"up", name:w2}, {p:"a", name:w3}, {p:"big", name:w4}, {p:"red", name:w5}, {p:"block", name:w6}, (w1 w2 w3 w4 w5 w6)}. (7.1)

In this context, a general set of TXs which are sufficient for the given understanding problem comprises the following rules, where the special symbol $ indicates that a new, unique parameter symbol is to be generated and substituted for it and where comments are enclosed in brackets, as [this is a comment]:

lex1:{{p:"pick", name:x}}➔{(... $/+verb+infinitive+transitive+accepts particle+"pick"/ (x) ...)}. [If the word "pick" (x) is there, generate a new higher-order node ($)

having x as its only subordinate. Mark $ as having the features verb, infinitive, transitive, accepts particle, and "pick" (indicating that the extent of the tree named $ is the phrase "pick").]

lex2:{{p:"up", name:x}}⇒{(... $/+particle+"up"/ (x) ...)} [If the word "up" (x) is there, generate a new higher-order node ($) having x as its only subordinate. Mark $ as having the features particle and "up".]

lex3:{{p:"a", name:x}}⇒{(... $/+determiner+indefinite+singular+"a"/ (x) ...)}. [As above.]

lex4:{{p:"big", name:x}}⇒{(... $/+adjective+"big"/ (x) ...)}. [As above.]

lex5:{{p:"red", name:x}}⇒{(... $/+adjective+"red"/ (x) ...)}. [As above.]

lex6:{{p:"block", name:x}}⇒{(... $/+noun+singular+"block"/ (x) ...)}. [As above.]

mvbl:{(... x/+verb+infinitive/ (x1) ...), (... y/-verb/ (y1) ...), (... x1 y1 ...)}⇒{(... x/+main verb/)}. [If x, comprising the constituent x1, is a verb and y, comprising the constituent y1, is not a verb, where x1 precedes y1, assert that x is the main verb.]

vgl:{(... x/+verb+infinitive+"xword"/ (xl) ...), (xl ...)} ->{($/+verb group+imperative+"xword"/ (x) ...)}. [If x is an infinitival verb, associated in extent with the actual word "xword," and its only constituent xl is the first word of the sentence, assert that $ is a new verb group representing an imperative command. Its extent is "xword" and its only constituent is the verb x.]

adj*1:{(... x/+adjective+"xword"/ (xl) y/+adjective+"yword" / (yl) ...)}->{(... $/+adjective+"xword yword"/ (xl yl) ...)}. [This rule recognizes that any series of two (... x y ...) or more adjacent adjectives constitutes a single adjectival phrase and concatenates the two constitutent words ("xword yword") under the name ($) of a new adjectival node.]

ngl:{(... x/+determiner+anyxfeature+"xword"/ y/+adjective+ "yword"/ z/+noun +anyzfeature+"zword"/ ...)}->{(... $/ +noun group+determiner+anyxfeature+anyzfeature+"xword yword zword"/ (x y z) ...)}. [This rule constructs a new noun group node ($) out of any adjacent series of a determiner (x), an adjectival phrase (y), and a noun (z). In addition, the use of variables called anyxfeature and anyzfeature causes such features as singular and indefinite to be passed to $ from the determiner and noun subordinate nodes.

clause1:{(x/+verb group+imperative+"xword"/ (x1/+transitive +accepts particle/) y/+particle+"yword"/ z/+noun group +"zword"/)}→{($/+clause+major+imperative+particle+transitive+"xword yword zword"/ (x y z)), (... z /+object+first object/ ...)}. [This rule recognizes that an imperative verb group (x) comprising a particle accepting transitive verb (x1) at the start of the sentence followed by a particle (y) followed by a noun group (z) at the end of the sentence together constitute an imperative main clause in which the noun group z represents the object of the transitive verb x.]

Several very important points should be made about this SDC system of TXs: (1) it is easy to generate, read, and understand and is less complicated and theoretically more efficient than comparable top-down or hybrid procedures; (2) it computes in a temporal order comparable to that of alternatively organized parsers; (3) it is easily extended; and (4) it is entirely learnable in a straightforward way. Each of these points is now considered in turn.

(1) An SDC understanding system is simpler and more efficient than the alternatives

This assertion derives from numerous observations about the structure and performance of variously organized parsers. In all top-down parsers, the programmatic code is obfuscated by the necessity for backtracking mechanisms, recursion ("pushing") and return of control ("popping").

Although in many cases the grammar may be conveniently represented as an augmented transition network diagram (Woods & Kaplan, 1971), there is no easy way to represent the complex actions and "demons" that are associated with many arc transitions. Thus, a good understanding of the grammar inevitably requires a detailed investigation of the program code and this, in turn, requires any student of the grammar to generate and retain memory records of the past states and previously "accepted" words in order to comprehend exactly what function is served by any particular subsequence of program statements.

In representing SDC systems, on the other hand, every computational change may be independently evaluated. That is, the knowledge necessary for each behavior rule is represented and utilized independently of everything else that is otherwise known or knowable about the language. Thus, a student of an SDC parser could begin his inquiry with any of its rules and attempt to grasp its meaning without concern for rules previously or subsequently investigated. For the same reason, useful SDC systems are easy to design and repair.

Perhaps more importantly is the theoretical efficiency of SDC systems. If it is supposed that direct memory contact matching is done with a fixed expense which is independent of the complexity of stimulus or template patterns and that all templates may be simultaneously checked, SDC clearly superior in efficiency to the alternatives.

This is true because at each point, these systems insure that only those computations are done which construct higher-order grammatical or logical structures; furthermore, no heuristic or backtracking devices are needed. In practical modern day computers, such an assumption is nonetheless clearly incorrect, and this fact militates for a detailed cost-benefit analysis of alternative system organizations for any practical problem. Nevertheless, this paper is essentially predicated on an assumption--the possibility of which was demonstrated in Chapter 5--that such immediate matching is the basis for natural, intelligent, SDC systems. As a result, difficulties of practical (artificial) implementation of SDC should not deter investigation of the basic features of these systems in "theory."

(2) The comparability of temporal behavior sequences of SDC and its alternatives

It is interesting that, although the TXs of an SDC system are conceived to operate asynchronously, in parallel, whenever their contingencies are satisfied, the actual time course of behaviors is quite comparable to behaviors arising from alternative organizations. One minor problem, of course, is to define what the actual temporal behavior sequence is for top-down methods. However, if it is agreed that each node on the parse tree "occurs" when it is completed by Winograd's program, then the top-down temporal sequence is, roughly: (a1) "pick is a verb; (2) this verb is a main verb; (a3) the main verb is a verb group; (a4)

"up" is a particle; (a5) "a" is a determiner; (a6) "big" is an adjective; (a7) "red" is an adjective; (a8) "block" is a noun; (a9) "a big red block" is a noun group; (a10) the verb group, particle, and noun group constitute a clause. This sequence is determined by the successful completion of a corresponding top-down, depth-first, left-to-right parse.

Each of these ten actions has a correspondent in the activity of the proposed SDC understander. In the latter case, if probable simultaneity of behaviors is denoted by membership in the same set, the actual sequence of TXs is: (b1) {lex1 ("pick" is a verb), lex2 ("up" is a particle), lex3 ("a" is a determiner), lex4 ("big" is an adjective, lex5 ("red" is an adjective), lex6 ("block" is a noun)}; (b2) {mvb1 (the verb is a main verb), adj*1 (the two adjectives are a compound adjective)}; (b3) {vg1 (the main verb is a verb group), ng1 ("the big red block" is a noun group)}; (b4) {clause1 (as in a10 above)}. Thus, while all lowest-order (lexical) operations (b1) occur first, the important subordinate-before-superior temporal structure is preserved in the SDC system. In contrast to top-down or hybrid behaviors, however, no superior node is hypothesized (or its corresponding searching theorem proving program invoked) in SDC unless it is guaranteed to be successful because all of its necessary constituents have been previously identified.

(3) SDC systems are easily extended

Just as it was easy to generate a list of eleven general TXs capable of parsing the exemplary sentence and any other sentence of comparable grammatical structure, so is it easy to add new rules to handle additional types of problems. If the needed matching of STM and rule contingencies is automatic and performed by direct memory contact, the increase in size of the total rule set would be without negative effect. On the other hand, in order to modify a top-down system in comparable ways, it would be necessary to "patch" or perhaps totally rewrite numerous programs and to inspect carefully the resultant logic for unanticipated side effects. Moreover, because such systems operate by transfer of control from superior to subordinates and backtracking upon a lower-order failure in order to reconsider previously untried higher-order alternative paths, any significant extension of these systems entails a serious threat of reduction of their "throughput" or processing capacity.

(4) Learning language processing rules in an SDC system

The reader will probably have noticed the structural similarity of the parsing rules of this section and the rules of transformational grammar studied in the preceding chapter. Where those rules related one grammatical tree to another of comparable complexity, the parsing rules in the current section were more "constructive" in that they always supplied information which is represented in figure 15

as arcs or markers connected to higher order nodes. It is obvious that such differences are immaterial to the two-step rule learning procedure presented in Chapter 6, however. That is, if appropriate training information is presented, each of the rules used for bottom-up language parsing will be abstracted in precisely the same way in which those of transformational grammar (proof in Appendix IV).

In the current context, the necessary stimulus-response training information would be pairs of less and more complicated grammatical structures representing the results of successively recognizing some additional grammatical regularity pertinent to specific stimulus components. Thus, it is supposed that the training information for the language learner is a set of numerous pairs of temporally successive structures resulting from successful transformation of less recognized or analyzed sentence structures into more detailed ones. In an artificial context, such sequential pairs would be extremely simple to generate and provide to a synthetic learner. In a similar way, it is assumed that such sequences are the informational basis for language learning in natural systems. This learning process is conceived to operate in the same way as that which underlies general skill acquisition where repeated, temporal behavior sequences which approximate desired muscular skills, in programs of response-contingent reinforcement (operant conditioning), supposedly stimulate the formation (abstraction) and strengthening of the desired skill rules.

In general, theories of operant conditioning do not propose theoretical mechanisms to explain how the organism produces such repeated approximations which can then be refined by learning mechanisms to a skill of precision. It would seem however that the generation of such behaviors is of major theoretical importance if, in fact, they are the raw material for continuous and undiscerning learning mechanisms which convert them directly into persistent and refined behavior rules. In the next section, the theoretical nature of this process is considered in detail.

7.5 Reconstruction, Supplementary Information, and Trial-and-Error Learning

To this point, it has been shown how SDC systems allow for natural representations of numerous skills (pattern perception, transformational grammar, and language understanding) and how simple, general procedures are adequate for the induction of the rules of such computational systems from suitable training information. In order to complete the analysis of the central question, "how can natural and artificial systems abstract intelligent behavior rules," it is necessary to identify mechanisms which can make the training-learning process a closed system. That is, it must be shown how the appropriate training information for the organism's learning is provided to it endogenously rather than exogenously.

Toward this end it is necessary to describe two rela-

ted functions: first, the presumed natural process of pattern reconstruction or, more generally, stochastic responses to partial stimulus-contingency matches and, second, supplementary or free-associative behavior in which logical dependencies are automatically retrieved (or activated) in the presence of related stimulation. Hebb (1949) was the first to propose that learned patterns are internally represented by organized stereotyped spatiotemporal neural firing patterns among individual cell assemblies corresponding to the internal representations of components of the stimulus pattern. Moreover, he posited, the interneuronal synapses are modified in such a way that the occurrence of any significant portion of such a learned firing pattern reconstructs (reactivates, redintegrates) the entire pattern. As a result, the partial-match between a stimulus and a previously learned pattern probabilistically results in the "perception" of the entire pattern. The feasibility of Hebb's theory, as modified by other cell assembly theorists, was empirically demonstrated in a simulation by Finley(1967) in the context of a periodic stimulus pattern (e.g. a repeated piano scale). Furthermore, Cavanagh (1972) demonstrated the general mathematical feasibility of reconstructive holographic processes in neural nets. A good discussion of such processes in typical memory tasks may also be found in Horowitz & Prytulak (1969).

In fact, the ability of a partial-match between a sti-

mulus and template to evoke the recognition of the entire template pattern is so general that it must obviously be carefully controlled by any system which intends to exploit it in "intelligent" behavior. While reconstruction is a useful basis for "creative leaps" from one set of conditions (real) to another (similar, but imagined), it is equally likely to produce a plethora of undesirable overgeneralizations (Hayes-Roth, 1973a; Moore & Newell, 1973). That is, while it may be generally desirable for stimuli to evoke every related conceptualization, it is absolutely essential that detailed mechanisms be available to control effectively the selection of more "pertinent" information from the entire set of possibilities.

To be precise, it is supposed (after J. Locke, J. S. Mill, D. O. Hebb, B. F. Skinner and many others) that temporal contiguity and temporally proximate reinforcement are the principal preconditions for the establishment of mutually excitatory connections between those neural structures that represent similarly continguous stimulus or response components. These connections are, presumably, the mechanism underlying a general class of symmetric contingencies, for example the previously discussed conceptual dependencies of Schank (1972). As a result of the development of such mutually excitatory connections, the occurrence of components of a pattern probabilistically reconstruct the other elements of the pattern. Excessive generalization, on the other hand, is controlled by the development of ap-

propriate inhibitory interconnections between units whose successive temporal activations have consistently preceded unpleasant experiences.

The exact realization of such proposals in the SR system may be achieved in a variety of ways. For example, it may be supposed that any known event is reconstructed and placed in STM whenever any "good" partial-match occurs between it and something already in STM (see, for examples of such mechanisms, Hayes-Roth, 1973a; Becker, 1973). While this is a simple function that could be easily simulated by use of the IM procedure for partial-matching, it is grossly insufficient as the sole basis for the control problems just suggested. Specifically, such a mechanism would have no capacity to "remember" that certain reasons (partial matches) for reconstruction of some patterns were adequate, while others were not. For example, while the sequence of letters "S T O P" on a slightly deformed, hence not strictly octagonal, red sign might be a desirable reason for invoking a stopping response while driving, the reverse letter sequence, "P O T S," on a perfectly octagonal red sign would not be.

It is obvious that there can be, unfortunately, no cheap solution to this representation and memory problem. Wherever responses appropriate to two distinct but very similar stimulus patterns are radically different, it will be necessary to maintain and use much specific information for proper control of excitatory and inhibitory processes. It

seems obvious, too, that such discrimination powers are the rule, rather than the exception, in real world intelligences.

For these reasons, it is suggested that any SR model of a complex intelligence should operate as follows. A priori, any partial-match between a stimulus and a known pattern (a PSR) or between a stimulus and a TX contingency component should have a probability of activating the corresponding memory unit which is an increasing function of the degree of commonality between stimulus and template. At each point in time, however, pairs of the successive complete state descriptions of the organism are supplied to a learning procedure. This procedure operates exactly as described in the discussions of rule learning in chapter 6 and pattern learning in chapter 4. The results of such a procedure are hypothesized transformations which are stored in memory. They are reconciled or consolidated with existing cooperative and competitive behavior rules--that is, the overall performance or future strength of their responses is evaluated--according to arbitrary performance criteria like those of schematic classification theory (section 4.5). As a result of such continual rule learning and reconciliation, strong TXs develop whenever they are consistently reinforced (adaptation). They are produced, on the other hand, whenever the organism's successive state descriptions manifest a related stimulus and response which include those components pertinent to the desired behavior.

In most cases, the first occurrence of such pertinent structure will result from the idental generation (stochastic reconstruction) of an appropriately related pattern.

The complete character of typical trial-and-error learning, which is the essential mechanism at work in this framework, is easily understood in these terms. Consider, for example, the remaining problem of identifying an endogenous supplier of the structural relations whose availability would enable the abstraction of the rules used in parsing the exemplary sentence of the preceding section. At any stage of development, the organism will have learned various syntactic and semantic relationships. These may be as complex as the conceptual dependencies of Schank(1972) or the conceptual relations in the semantic memories of Quillian (1968, 1969) and Rumelhart, et al. (1972), or they may be as simple as the grammatical constructs of clause, verb group, and noun group constituency and the lexical categories (e.g. adjective, noun, determiner) which would be required for the induction of the parsing rules used in the last section.

In any case, the effect of the combined presence of such knowledge, represented as PSRs or TXs, in LTM and related stimuli in STM is the automatic introduction into STM of essential supplementary (free-associative) information due either to partial or complete matching of transformational contingencies or to reconstruction of mutually exci-

tatory components of PSRs. Subsequently, this supplementary information becomes part of the complete state descriptions which directly regulate effectors and observable behaviors and which are processed by the continuous learning procedures. As behaviors are systematically positively or negatively reinforced, so are the simultaneously abstracted TXs made more or less excitatory in their future control of behavior.

In sum, the natural processes of rule learning are considered to be controlled by two logically simple internal mechanisms: (1) stochastic stimulus-driven (associative) activation of event or transformational response patterns with probabilities that are directly related to prior frequency and magnitude of reinforcement; and (2) continuous automatic induction of transformational rules from pairs of successive complete state descriptions and the reconciliation of newly produced rules with the existing stock. Whereas the limited practical application of such rule learning procedures is quite feasible, when carefully selected training information is supplied to IM-based rule constructors, it is obvious that artificial computing systems of the present and foreseeable future will be woefully insufficient to aproximate the apparently vast capacity for learning possessed by natural systems. The problem with using contemporary hardware as a basis for learning is, simply, that there are too many things to learn, not enough places to put them, and not enough time to com-

pute them. Nevertheless, the procedures described in this paper represent a clearly operationalized description of what must be done and feasible, albeit evolutionarily primitive, mechanisms for general programmatic behavior learning.

Chapter 8: EXPANDING KNOWLEDGE

8.1 Chapter Overview

Previous chapters have described the ways in which existing knowledge, in terms of stimulus encodings and supplementary information, may be the basis for the acquisition of new TXs and how such Txs may be adaptively strengthened in response to systematic reinforcement. The current chapter is intended to provide a model and representation for the expansion of knowledge in other principal ways. The development of new predicate relations in response to the recognition of occurrences of newly learned patterns is one such expansion. Similarly, the formation of categories, identification of the conceptual contexts or domains for such categories, and the generalization of specific rules over such domains are discussed and modeled. Finally, one problem of cognitive development, learning to learn, is discussed in terms of its role in information compression and knowledge expansion. A model of such "higher-order" learning is then described. The problems discussed in the current chapter are considered to be the major issues in the distinction between primitive and sophisticated adaptive information processing systems. None of the capacities discussed here--origination of predicates, category learning, generalization, or learning to learn--is theoretically necessary for intelligent behavior. On the other hand, such mechanisms may be essential for the

extensive cognitive development of systems with limited processing capacities or limited time for development. While it may be possible and important to analyze the information processing gains and losses associated with various investments in one or another "unnecessary" but "useful" function (e.g. the development of a highly structured retinotopic mapping from the eye to the lateral geniculate nucleus to the visual cortex vs. a less structured projection system), this is not currently feasible. As an alternative, functions like those discussed in this chapter which are intuitively "important" may be first formally analyzed and investigated in the hope that, as a result, their "significance" will be more clearly understood. That is the course followed here.

8.2 Creating New Predicates: The Representation of Patterns

The necessary development for the explanation and proposed model mechanism for the creation of new predicate relations was provided in the discussion of pattern learning in Chapter 4. The motivation for the creation of a new predicate is that it will allow a more efficient representation of a known pattern than is naturally provided by the possibly complicated PSR which is its template. Thus, while it was shown that any PSR may be used as a template and that matching of any stimulus and template can be performed in a direct, associative manner (Chapter 5), it is plausible that in natural systems some benefit may be de-

rived from the assignment of an appropriate simple label to all occurrences of a pattern. Such a simple label, like "the structure at coordinates (31, 67, 567) is an 'A'," would facilitate subsequent processing of the related structure because transformational contingencies which were encoded in terms of a relation like "is an 'A'" would be much less complicated than those necessarily encoded in terms of a corresponding graphical template pattern relating numerous lines and nodes.

The advantage of such information compression is obvious for the current genre of artificial intelligence computer programs. The desirability or necessity of such hierarchical compression in natural systems is less obvious. One possibility for arguing the value of such hierarchical information organization derives from the nature of the permutation problem in SANER nets. If natural SDC systems are organized for the recognition of stimuli matching contingencies in ways similar to SANER nets, the problem of permuting classes of multiple occurrences of the same predicate is greatly reduced by a hierarchical structure of sets of increasingly complex (informative) predicates. This follows from the necessary multiplication of the factorials (number of possible permutations) of relations within each predicate equivalence class. As the specificity of predicates increases, the number of stimulus instances of each one inevitably decreases and the permutation problem is, correspondingly, reduced. While such ar-

guments do not directly support the proposition that higher order predicates are fundamental to natural intelligence, they make clear some interesting bases for further mathematical and physiological research. The types of related mathematical questions are clear (see, for example, the discussion of hierarchical matching in Barrows, et al., 1972). Physiological correlates of these principles would presumably be evidenced in the visual system, for example, by directed processing networks evidencing numerous, repetitive, crossing, and fixed recognition circuits at the lowest (most peripheral) levels proceeding to increasingly plastic, unstructured, association circuits at higher levels. The low level circuits would be like an enumerative solution to the permutation problem involved in simultaneously processing numerous objects of equivalent types (e.g. lines, edges, contours), while the higher level circuits would provide for the learning and recognition of fewer but more complex patterns (e.g. an "A" or a face).

In any case, the SR system may be used to model the development of new predicates in a simple way. When a new pattern is abstracted, its PSR template is associated with some appropriate categorical response. In general, it would be reasonable to suppose that one component of that response should be a new parameterized relation identifying that some object currently matches this pattern template. A new unique predicate type may be coined and used in this relation. Furthermore, each parameter in the cor-

responding template PSR parameter set identifies a prototypic object or structure which has necessarily been assigned a correspondent from the stimulus structure. Each prototypic parameter is then interpreted as a distinct object type and its stimulus correspondent is viewed as the associated object. For example, the temple T(A) in (3.8) for the gothic capital "A" in figure 5 uses five parameters, A1 - A5, to represent nodes in the line drawing of an ideal "A." As a result, any stimulus which matches it, such as the "A" in figure 3 and (3.5) under the parameter correspondences a_i=A_i (i=1,2,...,5), may simply be represented by the predicate relation:

{p:T(A), A1:a1, A2:a2. A3:a3. A4:a4, A5:a5}. (8.1)

Thus, the predicate base of the SR system is easily extended. A precise understanding of the value of such extensions, as suggested above, is not yet available. For a completely different sort of view relating to the value of such "mappings" as that represented by (8.1) in the context of general problems of knowledge expansion and problem solving, see Moore & Newell (1973). In that paper, it is shown that many interesting problems are solvable simply by comparing two PSRs (called "beta structures" there) representing current and goal states by successive matchings between intermediate states accessible from the given states by transformations (their name is "actions"). In each case the results of such mappings, indentifying correspondent elements in the transformed structures, is a

simple, one-to-one assignment of prototype ("goal") and stimulus ("current state") objects like that in (8.1).

In sum, a pattern template comprising a structural description using n parameters, is representable as a corresponding n-ary predicate (the pattern name) and n predicate objects (the stimulus objects corresponding to each template parameter). Such patterns may be issued as supplementary messages in SDC as a result of matching stimulation. This behavior would be like the assignment of the label "determiner" to the word "a" in the recognition of "Pick up a big red block" (chapter 7). The addition of new predicates in response to the development of pattern recognition templates serves two purposes. It permits a useful and unlimited expansion of the elementary representational units, the primitive predicates, and it allows for the compression of complex contingencies into simple descriptions in a hierarchical SDC system.

8.3 Learning Concept Domains and Categories

The opinions expressed in the preceding section regarding the possibilities of advantages associated with development of higher-order predicates apply, with appropriate substitutions, to the value of a system which develops higher-order category concepts and an understanding of the contexts or domain in which members of a particular category are used. Because, by now, the reader is probably capable of generating "pros" and "cons" by herself or himself, I

limit my efforts to an explication of the mechanisms and uses of category formation.

A concept is defined to be any knowledge rule (a contingency-response TX) specifying the cases in which a computational operation should be performed and what its consequences are. A concept domain is, simply, the extension of the intensional contingency description of the associated rule. For example, a rule which recognizes "a," "the," "my," etc. in some verbal contexts as "determiner" has as its domain just those situations which match its criterial contingency. These terms will become increasingly clear as a result of their use in the following discussion of the development of the lexical categories of English.

Suppose, for the sake of simplicity, the following definitions: a verb is any word which generally precedes or follows a noun and occurs in the presence of some salient changes (actions) in the environment; a noun precedes or follows a verb and usually occurs in the presence of some salient, stable characteristics of the environment (objects); an adjective precedes a noun, follows a determiner, and usually occurs in the presence of a reliably encoded aspect of a variety of objects; and a determiner usually precedes an adjective or a noun as well as disambiguates alternative referents satisfying the same adjective and noun description. Specifically, suppose that each of the relations in the above concept definitions represents a feature which is determinable on the basis of regularly

computed, pre-existing conceptual rules and that, at the current stage of its development, the organism faces the task of abstracting the structural meaning of the underlined lexical concepts and their grammatical interrelationships. Each of these concepts has been chosen because it is apparently useful in parsing sentences, corresponds to a repeated structural pattern to which the language learning organism is frequently exposed and, most importantly here, is an example of a category comprising numerous distinct members which, nevertheless, occur in consistent if abstract structured contextual domains which are described by the above supposed definitions.

Thus, the learning problem here is to discover the consistent interrelationships defining each lexical category, the members of that category, and the contexts in which they are applicable. The mechanism which is needed to solve this problem is that of partial-matching (Chapter 4). As in Chapter 7, the assumption is made that supplementary relations describing the known relationships among present stimulus objects are automatically introduced into STM and that current stimulation and comparable past events are the basis for rule learning and pattern abstraction. In the current example, this would mean that all of the temporal patterns and physical attributes of the objects associated with the stimulus words in the current speech as well as those reconstructed memorial patterns which are similar to these, would be available for rule learning and pattern

abstraction.

Consider the nature of the patterns abstractable from such training information as this by use of the IM procedure for partial-matching. Abstractions will be produced which reflect absolute commonalities between the compared pattern representations. Simultaneously, irreconcilable differences among the compared patterns are identified as residuals. Here is the crucial point: at the same time that several components x1, x2, ..., xn in events E1, E2, ..., En are presumed to correspond in a common abstraction, other components y1, y2, ..., yn which relate to each of the x_i in identical ways within each of the E_i cannot be forced into strict equivalence by binding because they are different. For example, the PSRs of the two sentences "a dog barks" and "the dog barks" will match completely except where the first will use a parameter y1 to refer to the word "a" (e.g. {p:"a", name:y1} and {p:sequence, 1=start, 2=y1, 3=parameter name for "dog", 4=parameter name for "barks", 5=end}), the other will use a parameter y2 to refer to the word "the," the true correspondent of "a" in the two sentences.

It was in anticipation of such problems that uniform PSRs were introduced (section 4.7). If uniform PSRs were used in the partial-matching of these two sentences, the correspondence of y1 and y2 would be explicit in all ways but one: they would correspond in their similar occurrence as the first word, as the word directly succeeded by "dog"

and then by "barks," and in any other way in which the two sentences are equivalent; they would differ, however, only in their occurrence in the two distinct relations {p:"a", name:y1} and {p:"the", name:y2}. Thus, while the same parameter will participate several times in the common abstraction of uniform PSRs, its simultaneous use in different ways in the different reference events is also clearly indicated by the careful way in which models of uniform PSRs are constructed. All similarities are incorporated into an abstraction and all differences remain as differences of some content predicates between objects which are, otherwise, corresponding parameters.

Of course, noting the correspondence but irreconcilable differences between the predicates "a" and "the" (identifying labels for previously learned phonetic patterns) is exactly what is needed to generate members of the category determiner. This result is generalized in a straightforward way by the following rule which is a complete solution to the stated category formation problem: when the uniform PSRs of comparable patterns differing only in the substitution of one category member for another are partial-matched, an abstraction is produced in which the residuals comprise single relations of different unary predicates (such as "a," "the," "my," etc.) with bound correspondent parameters; the set of these predicates is the related category and the common abstraction is an element of the concept domain of the category concept. Moreover, the partial matching of all

such elements of the same concept domain will produce: (1) an abstraction which constitutes, precisely, the intensional description of the concept; and (2) residuals which are logically clustered on the basis of parameter correspondences and include, among others, one cluster which comprises all members of the related category.

To complete the description of the categorizing process, then, it is only necessary to apply the general process of retrieving supplementary information on the basis of pattern reconstruction to the case of a category member reconstructing the name of the category. For example, one might model this by assigning to each category D_i a unique predicate name C_i symbolizing the occurrence of that category and require that the occurrence of any stimulus, say x, which is a member of category D_i result in the supplementary message $\{p{:}C_i, name{:}x\}$. This was, in fact, just the sort of rule used in marking each word in "Pick up a big red block" with its appropriate lexical category type in the SDC understanding system previously described (section 7.4, rules lex1 - lex6).

To summarize the results to this point, category formation is viewed as the process of identifying a set of corresponding predicate residuals of partial-matched uniform PSRs and categorization is the subsequent production of a supplementary relation whose predicate is the name of that category (set of residuals) and whose object is the stimulus element belonging to that set. Because learning

is viewed as a continuous process of matching current and related event patterns, category formation is seen to be a by-product of the comparison of stimulus structures differing primarily on the basis of arbitrary selections of representatives of the same category.

8.4 Generalization over Members of the Same Category

Once category formation has occurred, categorization is the basis for new and more generalized learning. By virtue of the automatic production of supplementary categorizing messages about components of the current stimulus, subsequent learning takes place by rule induction from patterns which manifest, in addition to specific stimulus properties such as x occurs before y, statements about the more general facts, like those that x is a determiner and y is an adjective. When these patterns, in turn, are used for partial-matching and rule induction, the best abstractions obtained will probably (in increasing frequency) reflect regularities based on these higher-level properties rather than some based on lower-level properties (e.g. the repetition of exactly the same word). But the process of inducing such higher-level rules has already been described in relation to exactly this sort of supplementary training information.

In this view generalization is the induction of rules which are based on examples manifesting several members of the same category and which retain, as criterial

for their activation the requirement that x be a member of a specific category. This is a view of generalization which emphasizes its (extensional) relationship to categories *per se* rather than its (intensional) relationship to the properties which define the category. Given the one-to-one identification of categories with their defining concept domains, any distinction between the two as bases for describing how generalization learning occurs is thus arbitrary.

In short, rule learning is conceived to be, invariably, an act of generalization. The informational basis for rule learning is assumed to be the total set of successive pairs of complete state descriptions while the principal mechanism of learning is the partial-matching of these information structures. As a result, rules are always produced by an act of abstraction which throws away some information in order to discover reliable and desirable regularities. Were it not for the presumed capacity of intelligent systems to reconstruct supplementary information which augments the "concrete" stimulation as a basis for learning, such abstraction would be very limited. Augmented by the mechanisms of pattern and category formation and associative production of supplementary information, these systems have capacities that are clearly synthetic (ideogenetic) and creative. In this context, then, abstraction (identification of commonality) is not only a sufficient model of learning; it is a parsimonious one, too.

8.5 Learning to Learn

The higher the level of process considered, the more abstract the discussion becomes. In this section, an attempt is made to describe in very abstract terms how intelligent SDC systems may be used to model learning to learn phenomena. A general behavior (set of rules) is a learning procedure if its effect, in general, is to increase the probability of the acquisition of rewarded knowledge (patterns or rules). Thus, behaviors that obtain useful information, generate valuable hypotheses, and aid in an evaluation of their quality and applicability are principal examples of learning procedures. Learning to learn in this context means, simply, acquiring and using such learning procedures.

From all that has heretofore been shown about the induction of rules by partial-matching procedures in the SR framework, it is immediately apparent that learning procedures are acquirable in exactly the same way as are other behaviors. In this view, learning to learn is a direct result of the generalization of behavior rules from one "problem" context to another due solely to the representational comparability of the "training" and "transfer" tasks. While this is not a concrete proof that the general problem solving strategy x is induced from its use (in the sense that its criterial aspects were manifested) in problems p1, p2, ..., pn, such learning to learn could be explicated in the same manner in which other behaviors were modeled if

the complete problem descriptions necessary for rule learning were provided.

Here, however, I would like to indicate briefly three classes of apparently important and general learning procedures whose acquisition would seem to be easily modeled in the current framework. The three learning procedures considered in turn are attentional control, problem solving heuristics, and problem solving algorithms.

Attentional control

A prototype of attentional control problems is looking in a particular direction and focussing on a particular object in response to the recognition of a related problem. For example, someone may ask, "Who is that?" and the organism must, before being able to answer at all, orient its visual receptors in a particular way merely to obtain necessary information. An SDC system would presumably learn to make the appropriate response on the basis of trial-and-error learning in the same way as frequently discussed. That is, aspects of the current task situation (e.g. "who" indicates a person; people look like pattern Q and usually subtend such and such angles of the visual field at various distances; the questionner is pointing in direction X) and the current stochastically determined behavior are related as stimulus and response pairs, are partial-matched with comparable previous stimulus-response pairs, and the resultant abstracted transformation is strengthened in proportion to the desirability of the ensuing outcome. In

such ways, although many distinct rules may be learned, the organism comes to make continually more pertinent and succesful responses. As a result, since sense organ orientation is one of the principal devices for controlling information flow, the organism acquires a procedure which directly affects its subsequent ability to "attend" to pertinent aspects of a new task.

Problem solving heuristics

A problem solving heuristic is, in the framework of SDC systems, a response which is based on the recognition of some criterial aspect of an unsolved task which the organism has reason to believe may lead to an eventual solution. When expressed in this way, it is obvious that anything the organism does which precedes successful resolution of a problem is the basis for the potential induction of a generalized heuristic. If connecting two sticks precedes the chimpanzee's reaching and eating bananas over its head and out of its reach, connecting two sticks is a heuristic which is applicable to an attempt to reach bananas out of the cage and out of its reach. The representation of such a heuristic, in terms of the contingency-response rule, "bananas out of reach" → "connect two sticks," is as obvious as is the way it would come to be inferred by the previously described two-step rule learning procedure. In short, heuristics are the manifestation of making comparable responses to problem stimuli which match the contingencies of previously rewarded TXs.

Problem solving algorithms

The most general and precise mechanisms for expanding knowledge which humans possess are the problem solving algorithms learned in various formal courses and training programs, such as those of mathematics, engineering, logic, mechanics, and so forth. In these courses, procedures are presented which dictate how problems fitting a certain description are to be solved by applying known operators to those objects which correspond to the parameters of the general problem solving procedure. For example, all "word problems" involving one unknown are to be solved as follows: the unknown is labelled x; an equation relating x to the constants of the problem is generated; the rules of transposition, multiplication, etc. are used to obtain an equivalent equation with x on one side of "=" and the desired value of the unknown on the other.

In the current framework, such algorithms are viewed as systems of substitution transformational rules of exactly the same sort as previously considered in detail in Chapters 6 and 7 (and like the production systems of Newell & Simon, 1972). Here however, these rules are presumably conditioned to a point of "overlearning;" that is, each rule is reinforced and repeated until its response occurs reliably in appropriate contexts. Although many such algorithms are formally communicated and more or less consciously controlled, many more are not. For example, the procedures underlying the understanding of novel sentences may be viewed as problem solving algorithms yet, for the most

part, these are not known in this sense. They are known by our nervous systems, but not by our intellects.

8.6 Summary

This chapter has shown how the SR system provides a basis for modeling many of the higher-level constructs of cognitive development that appear, at first glance, least tractable to stimulus-response analyses. The success of the same partial-matching mechanisms to account for these general capacities suggests the possibility that a vast amount of behavior and development is the product of a very limited number of functional mechanisms whose products may seemingly exhibit qualitatively different features as the level of processing advances from concrete to abstract. But, in the model proposed, such an advance is the natural result of the development of categories and their subsequent use in the induction (through abstraction) of rules which generalize previous experiences and produce behaviors which are structurally comparable to those leading systematically to desirable consequences in the past.

Chapter 9: CONCLUSIONS

9.1 Chapter Overview

In this chapter the major mechanisms of representation, matching and learning developed in this paper are reviewed, the major deficiencies of the current approach are summarized, and guidelines for future research are suggested.

9.2 Mechanisms of Intelligent Behavior

Ten fundamental ideas have been explored in this paper and related to the task of modeling intelligent behavior. Stimulus-driven computation (SDC) is characterized by a complete determination of subsequent behavior by its antecedents and the decomposition of the representation of the machine's decision function into independently operating contingency-response transformations (TXs). Parameterized structural representations (PSRs) are sets of relations over arbitrarily named objects, and TXs comprise a pair of such sets. Together these structural representations provide a basis for describing complex stimuli, memory templates, and general rules of behavior. Presumably, the ability to represent and transform structured events is fundamental to intelligent systems.

Because of the associative nature of SDC, an ability to perform all-or-none matching for recognition is also essential. Two completely different techniques for all-or-

none matching were developed in this paper. One employed the flexible interference matching (IM) procedure (chapter 4) to compare a stimulus and a template and to determine whether any set of parameter correspondence bindings would entail the complete reduction of the template's residual.

SANER nets and canonical abstraction powersets, however, provided computational bases for structured all-or-none matching which eliminated the parameter binding problem by introducing arcs between nodes representing predicate object loci occupied by the same parameter. Furthermore, while immediate recognition could be achieved by simple frequency comparisons in SANER nets, in general, only if all relations of identical predicate types in a stimulus representation were exhaustively permuted in individual subnets, no such permutation was required in the context of abstraction powersets. This powerset representation, as a solution to the permutation problem, necessitated vast numbers of canonical abstractions and corresponding frequencies to preserve all the structuring information implicit in a PSR and required for all-or-none matching.

Thus, it would appear that in "the representation problem," a direct trade-off may be made between the volume of information which is used to represent stimulus and memory template patterns and the complexity of computations required to perform the basic all-or-none matching operation. PSRs, on the one hand, are extremely compact bases for the description of structured events but require com-

plex procedures to solve the related binding and matching subproblems. Conversely, powerset representations use the complete list of frequencies of all subrepresentations contained by a PSR but permit all-or-none matching to be performed by a direct memory contact process requiring only numerous threshold frequency gates.

Partial-matching is the operation of comparing two or more representations to produce abstractions which are subrepresentations common to all of them and to identify residuals which are irreconcilable differences among the otherwise similar representations. Abstractions are the essential elements used in learning. Pattern learning is achievable by partial-matching events associated with a common response class and using the derived abstractions as templates for corresponding classificatory behavior rules. Similarly, general rule learning is achieved by a two-step process in which paired stimulus and response representations are first partial-matched to identify comparable substructures which are probable bases for the substitution of stimulus components into response structures and, secondly, several pairs whose internal components have been so matched to one another to abstract a more general TX relating stimuli and responses whose commonality with the training examples is maximal.

In addition to describing how potentially useful rules may be produced using the interference matching procedure to compute partial-matches, the need to evaluate the expec-

ted performance of alternatives was discussed in the context of the normative schematic classification theory. This theory prescribes a method for maximizing the overall expected performance of competing alternative contingency-response rules on the basis of exogenously supplied criteria and feedback. While a strictly competitive organization of alternative rules, in which the only rule to execute is the one most strongly indicated on the basis of the commonality between the current stimulus and previous comparable events, is dictated by the normative theory, a feasible alternative is to allow a hierarchical organization of contingencies in which the excitatory tendency of each rule is determined by the induced summation from all lower order subrepresentations excited (matched) by the current stimulus.

The formation of categories and its relation to generalization, the development of parameterized rules which are evoked by any stimulus structure in the appropriate concept domain regardless of which specific category member is present, were shown to have natural interpretations based on the residuals obtained by partial-matching. Generalization is, in this view, the result of the successive abstraction of categories from training information which exhibits various category members in a similar structural context, the automatic production of supplementary messages about conceptual dependencies and category memberships of current stimulus components and, finally, the induction of

contingency-response TXs by the two-step rule learning procedure.

Lastly, it was shown how even learning to learn phenomena could be modeled by the trial-and-error development of rules which directly regulate the flow of information and processing within the organism. As a result, such learning procedures act as higher-level controllers of the information which is the substratum for subsequently acquired knowledge.

9.3 Major Deficiencies of the Current Work

There are certainly as many problems unsolved by the current work as there are ones apparently solved. Continous and noisy variables have been omitted in the discussions of pattern representation and matching. Applications of the proposed methods for SDC, either for simulating natural systems or constructing practical information systems, would probably require excessively large amounts of memory and preferably parallel computing power. Practical models of and devices for attentional and feedback control have not been considered. Sensory analysis and encoding have simply been assumed while, in fact, they represent largely unsolved problems in theory and application. Unit statistics representing facts about the strength, fatigue, magnitude, intensity, and precise timing of patterns and TXs have likewise been ignored or simply assumed where useful.

Although spatial organization of memory may be an important functional characteristic, it has not been considered, nor could it be easily modeled by the current SR system. Similarly, although it is widely believed that neural structures and processes underlie natural intelligence, this paper has not demonstrated any feasible means of reducing structural representations to a neural representational basis.

Finally, it should be noted that the current paper is more a framework for the functional analysis of intelligent behavior than a theory of the precise operations to be found in any particular natural information processing system. The mechanisms of intelligence operationalized here constitute a list of functions which are presumably essential to natural, intelligent, information processing systems and, therefore, this list is a primitive model of intelligence. This model contains, however, many unestimated parameters and each of the supposed mechanisms will be very difficult to demonstrate empiricially. On the other hand, the framework accounts for numerous capacities of intelligent systems and suggests many directions for future research. The avenues for related analytical research are considered in the next section. Examples of unusual, empirically testable propositions which arise from the current analyses include: (1) each subpattern of a stimulus will be conditioned separately and unitarily as the contingency of a TX (a similar statement was proposed, tested

and confirmed by B. Hayes-Roth, 1974); (2) controlling for input time, any well learned pattern may be recognized at a fixed rate regardless of its structural complexity; (3) because of the need for numerous exposures to a pattern to consolidate its internal (neural) representation, the number of patterns of a particular type which will be detected in a fixed time interval from a stimulus manifesting multiple occurrences of that pattern in a controlled context of distractors will increase uniformly over practice upto some maximum value; and (4) when several components are overlearned in particular conceptual dependency relations, the presentation of any large subset of these will result in an enhanced perceptual or cognitive experience due to the automatic internal production of supplementary information; further, such information will interact with any on-going learning task and should be part of the effective stimulus which is conditioned to whatever responses are learned at that time.

9.4 Suggested Guidelines for Future Research

There are several suggestions which I would like to make on the basis of considering the apparent strengths and weaknesses of this research. First, although it has been seriously neglected by cybernetics and S-R psychologists, much attention should be paid to the structure of patterns and of their transformations. Any model of complex behavior should take into account the enormous degree

of organization in the environment and the behavioral rules in whose terms this organization is cognized about and perceived by the organism. Moreover, it should be realized that two sorts of parallelism are major properties of mechanisms of intelligence. First, there is the assumed parallelism in computation: each of the numerous rules of varying complexities are automatically and independently invoked in the presence of appropriate stimuli. Second, there is the extremely important parallelism in representation: the number of reasonable rules that may be simultaneously abstracted from any training set is very large and each has, a priori, an equivalent claim to the control of subsequent behavior. It might be presumed then, on the basis of its obvious adaptive value, that natural mechanisms of learning have evolved in such a way as to compute and evaluate these rules simultaneously.

Thus, in my opinion, modeling of intelligence faces a fundamental analytical problem: how can the vast amount of structured information potentially available in training information be efficiently, practically, and maximally exploited in learning and the subsequent control of behavior?

While the procedures discussed in this paper describe what must be done and some simple ways to do it, they fail to provide generally usable and inexpensive bases for solving this fundamental problem. The likelihood that the current genre of models of human memory (e.g. Rumelhart, et al., 1972; Collins & Quillian, 1972; Anderson & Bower,

1973), which provide little more than basic representational schemes for facts and at most simple, inflexible sequential processing procedures for fact retrieval and storage, will lead to answers to this fundamental problem is, in my opinion, virtually nil. Rather, successful solutions may be found only if new emphases are placed on: computational bases without enormous amounts of complicated sequential logic; the discovery of simple mechanisms with widescale applicability, in contradistinction to the popular practice of developing practical, if totally ungeneralizable, programs to solve narrowly defined tasks; and, finally, the formal analytical investigation of theories of adaptive memory and intelligent behavior.

It is only by first reasoning to an analytical solution of the fundamental problem, in my opinion, that cognitive psychology and artificial intelligence will ever progress to a point where a large-scale computational model of intelligence will be practicable.

Chapter 10: EPILOGUE: SPECULATIONS ON NEURAL INTERFERENCE PROCESSES AS NATURE'S SOLUTION TO THE FUNDAMENTAL PROBLEM

10.1 Chapter Overview

The major capabilities of an intelligent SDC system--structural representations (conjunctions of relations on objects and their transformations), direct contact matching and recognition, associatively excited supplementary messages, induction of rules through abstraction, and control of competitive responses--are compared to several corresponding features of natural neural memories. Speculations concerning what may be needed to close the gap between the previously functionally analyzed mechanisms of intelligence and the nervous system are considered.

Causal regression, the induction of necessary antecedents for particular responses, and the automatic, associative excitation of consequents of these antecedents in proportion to historical frequency and reward are the only mechanisms of neural control apparently required. In addition, a widely distributed neural code is required by the assumption that all patterns be learnable, that all transformations between patterns be learnable, and that no neural firing pattern significantly disrupts processing in any non-negligible volume of association memory.

10.2 Speculations on Neural Representations

The essential problems in a theory of knowledge representation are the representation of structured events and the representation of transformations between events. While a need for brevity precludes a detailed consideration here of the motivations behind my speculations, their functional consequences will be developed and taken as prima facie sources of motivation. Thus, proceeding directly, are the following speculations concerning neural bases for the representation of knowledge.

(A1) A unary predicate relation such as "the object x (at location y) is red," is locally represented by a stable modulation (increase or decrease) in the mean firing rate of a set of neurons whose location set is determined in some arbitrary way as a one-to-one function of the locus y and the predicate "red," where the object x is considered positioned in some space (e.g. the visual field, acoustic frequency-temporal space, etc.).

(A2) An n-ary relation, such as "the object w (at location x) is bigger than the object y (at location z)," is locally represented by the modulated firing rate of two groups of neurons, one whose location set is a (one-to-one) function of the locations x and z and the property "is bigger than," and one whose address is a function of x and z and the converse property "is smaller than." In general, the locations of the related objects in some space plus the (predicate, object) two-tuple uniquely determine the local addresses of the neurons whose firing rates are sup-

posedly modulated.

(A3) The number of neurons in each group participating in the representation of a relation and the number of axons, dendrites, synapses and their a priori strengths are sufficiently large and the spatial distributions of these is sufficiently diverse to insure that synaptic transmission coefficient modifications will in all cases be suitable to reflect pattern learning (see A7, below) and to preclude the possibility that any single component of a pattern disrupts recognition (see A9, below) of a well learned pattern which is simultaneously present.

(A4) The local representation of an event is the set of augmented neural firing patterns associated with the translation of its PSR into the codes prescribed by A1, A2 and A3.

(A5) The basic global representation of an event is the distribution of mean rate modulations of neurons due solely to the changes in their base rate behavior resulting from the spatiotemporal interference (summation) of all components of the local representation. It must be assumed further that the global representation is relatively stable under limited temporal averaging.

(A6) Due to innate transformational structure and previous learning, the basic global representation is automatically modified by all pertinent TXs. A TX is pertinent if the current firing pattern in memory causes a stable new firing pattern to be added to it. Thus, the

neural coding of the relation "x at location y is red" might be a sufficient condition for the transformation of distance and the precise modulation of certain neurons entailing the interpretation that "if x were at y+k, it would be red and displaced k units to the right of its current position." Thus, the augmented global representation of any event is the stable neural mean firing rate modulations due to the encoding of the event, the interference of local patterns, and the subsequent supplementation by all pertinent transformations.

10.3 Causal Regression and Learning

Causal regression is the process of modifying synapses to increase the probability that neural sequences which lead to (precede in temporal proximity) gratifying consequences are repeated with a frequency that increases in direct relation to their desirability. Conversely, those which lead to undesirable consequences are modified to reduce their probability of recurrence in direct relation to the magnitude of their undesirability. Causal regression is supposedly effected by increasing or decreasing synapses which are firing at above normal rates in the presence of feedback which is positive or negative, respectively (Hebb, 1949; Rochester, et al., 1956; Rosenblatt, 1958; Samuel, 1959; Selfridge, 1959; Good, 1965; Finley, 1967; Plum, 1972; Nilsson, 1965; Hayes-Roth, 1973a).

(A7) It is assumed that any distinct combination of

relations is discriminable as a distinct distribution of firing rate modulations and that causal regression continually operates to modify structure to affect the prescribed functions. As a result, it follows that causal regression, in fact, serves to abstract structured TXs in the same as previously described in the SR framework (Chapters 4, 6 and 7).

The ability of causal regression to produce TXs of the same type as those produced by the two-step learning procedure in the SR framework may be seen by noting the relationship between the underlying supposed neural processes and the functions of the SR system. Principally, each PSR is translated into a set of coordinated cell firing patterns. Thus, each subrepresentation corresponds to a discriminable set of firing patterns which are not, by assumption A3, obliterated by the presence of additional pattern codings. Since each subrepresentation is logically connected to the behaviors which ensue by virtue of the fact that all TXs are assumed learnable (i.e. excitatory pathways exist), each subrepresentation of a stimulus pattern may be associated to each response subrepresentation by causal regression. Finally, because all computation is representable in terms of organized sets of parameterized relations which are here directly represented as mean rate modulations of sets of neurons, structured behavior rules are acquired in the same way perceptrons learn to discriminate between two distinct patterns.

10.4 The Hypothesized Nature of Neural Computation

The major "trick" in the foregoing was the assumption that "every pertinent transformation" is automatically computed. In one swift logical leap, most of the problems in realizing SDC systems and reconstructing patial-matched patterns have been eliminated. It is simply assumed that the re-occurrence of stable interference patterns (spatio-temporal distribution of firing rates produced by summation of interacting constituent patterns) is a sufficient precondition for the reconstruction of those patterns which were previously associated with them. In brief, the positive results of Finley (1967) and Cavanagh (1972) are summarized and generalized:

(A9) Whenever any set of patterns S occur which contain among them a particular subset C which has previously been conditioned (via causal regression) to another set R, there is a positive probability that the pattern R will be reconstructed. Moreover, this probability is an increasing function of the strength of prior conditioning of C to R and of the salience (intensity, contrast, etc.) of C vis-a-vis the rest of S; conversely, this probability is a decreasing function of the strength of associations between other components of S and patterns other than R (the fundamental principle of predictive power).

Finally, some of the implications of the supposition that all pertinent transformations are automatically invoked should be considered. The notion here is quite simi-

lar to that suggested by Pitts & McCulloch (1947) as a possible basis for recognizing all physical transformations of the same stimulus pattern as equivalent. In that case, they suggested, detectors for each possible transformational variant would be connected to an averaging mechanism. If any of these were on, a constant result would obtain and the pattern would be recognized as if invariant under the related transformational group. In the current case, however, I am suggesting that every learned and innate TX is automatically invoked by the direct excitation of a rule's response representation by the occurrence of a stimulus firing pattern matching (containing) the rule's contingency (a particular interference pattern). For example, it might be supposed that any particular line segment in the visual field would be propagated through the brain while undergoing a variety of structured TXs, such as expansions, contractions, translations and rotations.

Such changes could be conditioned, the speculation goes, by repeated experiences of comparable "real" transformations in a person's development. For example, each time the eye is moved or refocussed, the neural firing patterns undergo exactly one of these sorts of transformational alterations. Then, if causal regression is a blind, faithful, and continuous operation, the subsequent ability of a line to induce the excitation of related transformational patterns would be acquired in the same way the ability of any other sign to evoke a corresponding response is.

Finally, the speculation goes, the permutation problem facing simple SANER nets is eliminated in neural nets by a very simple mechanism, but one which requires either much innate structure or massive amounts of conditioning. In brief, any structured event is first represented locally and then augmented by the interference of all of its consequent transformational forms. Thus, spatial transformations essentially cause the repetition of the same pattern to be repeated at various loci of memory regardless, for example, of the actual position of the stimulus in the visual field. As a result, it is not necessary to identify one-to-one matches between parameters and relations since the original parameter-free but position-dependent neural representation is made position-independent by augmentation by automatic TXs. Once parametrically and spatially independent patterns are present, recognition and invocation of related responses are accomplished by direct excitation of preconditioned pathways.

10.5 Conclusions

The preceding represent nine weakly supported speculations and related observations on the functioning of adaptive neural nets. At this point, each of the nine represents a statement of a valuable and difficult problem in theoretical psychology. Whether or not they are true individually, it is clear that with any set of comparably powerful and interacting properties, nervous systems are, on one

hand, potent computing machines and, on the other hand, a promising domain for rigorous analytical study. If, as a result of such study, any of the suggested functions could be demonstrated, the way would be paved for much more fruitful analysis and application of the principles of natural intelligence than has heretofore been possible.

APPENDIX I: SPACE LIMITED INTERFERENCE MATCHING MODEL BUILDER (SLIMMOB)

The Space Limited Interference Matching Model Builder (SLIMMOB) algorithm is a robust, generally applicable solution to the problem of partial-matching two parameterized structural representations (PSRs) or graphs when both are expressed as sets of relations over some number of arbitrary parameter symbols (or nodes). The principle of space limited interference matching (Hayes-Roth, 1974b), developed originally as a solution to the problem of identifying maximally performing classificatory feature set schemata when computation is primarily constrained by the amount of intermediate storage that may be used, is generalized here to the problem of producing models (essentially parameter correspondence binding functions) which are associated with maximally informative abstractions of the two PSRs being partial-matched.

The algorithm employs the following conventions. First, the two representations to be matched, S with M relations (S_1, S_2, ..., S_M) and T with N relations (T_1, T_2, ..., T_N) are separately partitioned into predicate equivalence classes; S(p) and T(p) denote the sets of relations in S and T, respectively, of predicate type p. Second, the m parameters of S and the n parameters of T are arbitrarily ordered: s_1, ..., s_m and t_1, ..., t_n, respectively. A computational form <u>model</u> of {S,T} comprises four bit strings and one numeric variable as follows. The string AS, of

length M, records which of the relations of S have been and have not been incorporated into the abstraction of the current model. If the k-th bit of AS is 1, denoted AS(k)=1, relation S_k is included in the current abstraction; otherwise, when AS(k)=0, S_k is in the residual of S. The string AT, of length N, plays the same for for T as AS does for S, m.m. Two other strings, ${}_S T$ and ${}_T S$ describe partial parameter binding functions from S to T and T to S, respectively. ${}_S T$ is of length nh and ${}_T S$ is of length mh, where h is some number such that $2^h-1>m$ and $2^h-1>n$. Presumably, h is chosen to conform in some natural way to the size of a byte or word in the computer being used to effect matching. Each of the binding functions is considered to be a list of successive h-bit indexes, where ${}_S T(k)$ and ${}_T S(k)$ denote the k-th h-bit indexes of ${}_S T$ and ${}_T S$, respectively. Whenever ${}_S T(k)=j$ for some $j = 1, 2, \ldots, n$, this is interpreted to mean that the assigned correspondent of the parameter s_k is t_j, and it will be simultaneously true that ${}_T S(j)=k$. Whenever ${}_S T(k)=0$ or ${}_T S(k)=0$ the interpretation is that no correspondent has yet been assigned to s_k or t_k, respectively. Finally, each model contains one real-valued variable, called sig, which is a measure of the significance or informativeness of the abstraction of S and T represented in the current model. The value of sig may, for example, simply be a count of the number of relations retained by the model's abstraction; alternatively, sig may be a complex function of contextually determined priorities in model

building.

Now, let the model building process be constrained by a limitation of the amount of memory space which it can use to compute alternative models. The list of models currently held in memory is called MLIST and, as a result of this limitation on space, the maximum length of MLIST is some value L. In brief, SLIMMOB operates as follows. Each relation in S is selected, one-at-a-time, without replacement, in some arbitrary order. Successively, each relation in T which matches the chosen relation from S is used to generate a set of parameter bindings between S and T entailed by the matched relations. In turn, each model in MLIST is examined to see if these parameter bindings are consistent with its current ${}_{S}T$ and ${}_{T}S$ lists. If so, a new model is generated with updated AS and AT lists which merge the matched relations with the set of previously matched relations, with updated ${}_{S}T$ and ${}_{T}S$ lists which incorporate the new parameter correspondences, and with a new value of sig appropriately increased to reflect the inclusion of an additional relation into the abstraction. Finally, the new model is stored in MLIST according to its sig value and, if necessary, the model with lowest sig value is deleted to make room.

A few details concerning the implementation of SLIMMOB deserve special attention. (1) The order in which relations from S are selected to determine which matching relations in T are used to generate parameter correspondence bind-

ings should reflect--in their serial order probabilities--any judged differences in the importance of relations of various predicate types. The most significant relations should be matched sooner (in probability) than others. (2) Although no truly simple operation has been found to determine whether the correspondences specified by two binding functions are consistent, the use of two separate functions, ${}_{S}T$ and ${}_{T}S$, is suggested with one simple approach in mind. For example, let the two ${}_{S}T$ functions to be merged be denoted A and B. The mapping from S to T is consistent if, for all values of k=1,2,...,m, $A(k)\neq 0$ and $B(k)\neq 0$ if and only if $A(k) = B(k)$. Such a condition is usually quite simply verified with a small number of computer instructions or could be easily microprogrammed as a single instruction. The consistency of the corresponding mappings C and D from T to S is determined in a completely similar way. If the pairs of merged binding functions in both directions are consistent, the merged bi-directional binding function is, then, clearly consistent (at most one-to-one in both directions). (3) When models are chosen from MLIST to be combined with newly matched relations, they should be sampled in an order which is a direct function of their current sig values, the highest first.

As a result of such a design, SLIMMOB reflects the following desirable properties. If enough space is available, all models of {S, T} are produced. In the absence of unlimited space, however, models compete for storage on the

basis of sig values which are an increasing function of the informativeness or significance of their abstractions. Thus, the procedure dynamically optimizes the expected overall sig values of those models actually preserved in storage and, hence, the informativeness of the obtained abstractions. Finally, the SLIMMOB algorithm is easily implemented, primarily in terms of simple bit string operations, and is completely general in its applicability.

APPENDIX II: SCHEMATIC CLASSIFICATION PERFORMANCE MEASURES

The essential theoretical principle underlying schematic classification is that, a priori, each characteristic manifested by an object is an equally plausible hypothetical basis for its classification as a member of some class C or its complement D. After collecting observations on the frequency with which various characteristics occur among exemplars of C and D, it is necessary to reevaluate the plausibility of each hypothetical classificatory basis in accordance with the empirical evidence. The ability of each characteristic to predict membership in any class may be evaluated independently of all others. Furthermore, whether or not a randomly selected test item S which manifests a certain characteristic T, denoted S(*)T, is actually a member of class C is assumed to be a Bernoulli process with unknown binomial parameter p. If p=1.0, T perfectly indicates membership in C; if p=0.5, the presence of T is uncorrelated with membership in C; and, so forth.

In general, the analytical problem is more complicated than simply determining the true value of p. This is because the goal is usually to maximize the expected benefit of using the rule, "S(*)T implies S belongs to C," in the context of certain expectations about how often elements of C will occur during testing and how significant a successful classification of S as an element of C is compared to an incorrect one. The measure which is proposed to reconcile these problems is called the performance of T as an

indicator of C, denoted U(T⇒C). It is the Bayesian posterior expected net gain of weighted correct less incorrect decisions resulting from the classification of all items matching T to the class C, under the assumption that the conditional membership of an item S in C given that S(*)T is a Bernoulli process with some probability p to which a uniform prior distribution is assigned.

Formally, if E denotes the expected value operator, G_C and L_C denote the gain and loss, respectively, associated with the correct and incorrect classification of an item as an element of C:

$$U(T\Rightarrow C) = E\{G_C\,[\text{proportion correct classifications of C indicated by T}] - L_C\,[\text{proportion incorrect classifications of S indicated by T}]\} \tag{A2.1}$$

$$= G_C\,E\{N_c(T\Rightarrow C)\} - L_C\,E\{N_w(T\Rightarrow C)\}. \tag{A2.2}$$

In (A2.2), the terms $E\{N_c(T\Rightarrow C)\}$ and $E\{N_w(T\Rightarrow C)\}$ indicate, respectively, the expected proportion of correct and incorrect classifications of items matching T as elements of C over a series of test items.

Evaluating these terms is straightforward. Let the random variable P_C be the unknown parameter of the Bernoulli process by which it is determined that any test item is drawn from C and, moreover, suppose P_C has an a priori density distribution which is rectangular (uniform) over the interval [0, 1], i.e. $P_C \sim R[0, 1]$. Similarly, let $P_{T|C}$ be the true value of the parameter of the Bernoulli process

underlying the probabilistic manifestation of characteristic T by any randomly chosen member of C and suppose that, a priori, the random variable $P_{T|C} \sim R[0, 1]$. Finally, let the random variable $P_{T|D}$ represent the parameter of the Bernoulli process underlying the conditional manifestation of the characteristic T by a randomly chosen member of D, non-exemplars of C. A priori, the uncertainty about the true value of $P_{T|D}$ is reflected by the density function $P_{T|D} \sim R[0, 1]$. Further, let RC and RD denote reference sets of exemplars of C and D, respectively, and let RC(*)T and RD(*)T denote the subsets of RC and RD, respectively, whose elements match T. These reference sets can be considered samples which provide additional information about the true values of P_C, $P_{T|C}$ and $P_{T|D}$ and, when combined with their maximally uncertain a priori densities, yield posterior density distributions which can be used to determine the optimal classificatory behavior.

The calculation of the posterior probability density function for the random variable P_C, denoted $P''[x \in C]$, may be considered as an example. The number of sampled exemplars which are elements of C is $|RC|$. Those which are not contained in C are elements of RD and their number is $|RD|$. The posterior density for P_C is then

$$P''[x \in C] = Be(1+|RC|, 1+|RD|), \qquad (A2.3)$$

the Beta distribution with parameters $1+|RC|$ and $1+|RD|$ (Ferguson, 1967). Similarly, the posterior distributions for $P_{T|C}$ and $P_{T|D}$ are, respectively:

$$P''[x(*)T \mid x \in C] = Be(1+|RC(*)T|,\ 1+|RC| - |RC(*)T|), \tag{A2.4}$$

$$P''[x(*)T \mid x \notin C] = Be(1+|RD(*)T|,\ 1+|RD| - |RD(*)T|). \tag{A2.5}$$

Now under the assumption that the probability $P_{T|C}$ that an item's conditional manifestation of T given membership in C is independent of the probability P_C that test items will be drawn from C, it follows that:

$$E\{N_c(T \Rightarrow C)\} = E\{ P_{T|C} \cdot P_C \} \tag{A2.6}$$

$$= \frac{1 + |RC|}{2+|RC|+|RD|} \cdot \frac{1+|RC(*)T|}{2 + |RC|}. \tag{A2.7}$$

Similarly, if independence of the probabilities $P_{T|D}$ and $P_D = (1 - P_C)$ are assumed,

$$E\{N_w(T \Rightarrow C)\} = E\{ P_{T|D} \cdot P_D \} \tag{A2.8}$$

$$= \frac{1 + |RD|}{2+|RC|+|RD|} \cdot \frac{1+|RD(*)T|}{2 + |RD|}. \tag{A2.9}$$

Thus, the performance value of T as an indicator of C is

$$U(T \Rightarrow C) = [\,|G_C|(1+|RC|) \cdot \frac{1+|RC(*)T|}{2+|RC|} - |L_C|(1+|RD|) \cdot \frac{1+|RD(*)T|}{2+|RD|}\,] \,/\, [2+|RC|+|RD|]. \tag{A2.10}$$

This general solution is provided for the typical case in which the true value of P_C, the rate of occurrence of items in class C during testing, is unknown and is assumed to be the same as that underlying the original sampling of reference sets RC and RD. When P_C is known, however, a simpler form of the performance measure is used:

$$U(T \Rightarrow C) = |G_C|\, P_C \frac{1+|RC(*)T|}{2+|RC|} - |L_C|\,(1-P_C)\frac{1+|RD(*)T|}{2+|RD|} \qquad \text{(A2.11)}$$

These equations may be immediately generalized to predicting membership in any class C_i (i=1,2,...,n). In each case reference sets RC_i amd RD_i and the subsets matching T, $RC_i(*)T$ and $RD_i(*)T$ for class C_i and its complement D_i (all other classes C_j, $j \neq i$, combined) would be used in place of RC, RD, RC(*)T, and RD(*)T in (A2.10) and (A2.11). As explained in the text, a test item S is then classified as an element of class C_i only when, for all T' such that S(*)T', there is some T such that

$$U(T \Rightarrow C_i) \geq U(T' \Rightarrow C_j) \text{ , for all } j \neq i. \qquad \text{(A2.12)}$$

Two additional points should be made about these equations and their associated rule. First, loss minimization is only one of a number of possible bases for governing choice behavior. Second, the independence of the events "X is a member of class C" and "X (*) S, for some schema S" may not be correct. For example, there may be reasons why the schemata manifested over time are somehow correlated with the duration of the experiment, and it would be adaptive to realize that the schemata associated with a particular class are changing over time. These are complications that are beyond the scope of the current theory.

APPENDIX III: CANONICAL FORMS OF PARAMETERIZED STRUCTURAL REPRESENTATIONS

The purpose of this appendix is to explain how any parameterized structural representation (PSR) or graph can be converted to a canonical form (CF) in which each parameter or node is assigned a specific numeric index label which is deterministically computed from the role it plays in the overall structure and each relation or arc is then re-expressed as a literal string over the assigned parameter or node indexes. As a result, the body of the PSR (a set of relations) or graph (a set of arcs) can be sequenced, concatenated, and written as a single string of alphabetic characters. Two PSRs or graphs are equivalent (isomorphic) if and only if their CFs are identical strings. Thus, it is suggested that structures be stored and manipulated in CF and that equivalence be determined by typical self-terminating character-by-character string comparisons. Furthermore, because any structure may be uniquely identified by its CF string, CFs provide a basis for associative recognition or retrieval of structured events in which an event's address is, in effect, its CF.

Suppose, without loss of generality, that any PSR is represented as a set of n-ary relations R_1, R_2, ..., R_m over elements of the parameter set $P = \{p_1, p_2, \ldots, p_k\}$. The principle underlying the development of CFs is that each parameter may be effectively identified (distinguished) by its inverse structural representation (ISR) which des-

cribes the structure of the relations in which it participates. If such ISRs can be constructed without the introduction of arbitrary parameter symbols and can be shown to identify each original parameter uniquely (upto equivalence of provably structurally indistinguishable parameters), each parameter's ISR (or some one-to-one function thereof) can be used as its new label in the production of a parameter-free structural representation. When the relations in this representation are sequenced by some arbitrary lexicographic order (which is trivial once the parameters have been removed), the resulting sequence of relations, considered as a single concatenated string, is the desired CF.

The details of this procedure are now given. The unconditional first order ISR of any parameter p_i, $ISR^1(p_i)$, is the lexicographically ordered bag (a set with possibly repeated elements) of all relations in which p_i occurs modified as follows: wherever p_i occurs, it is replaced by the literal "1"; wherever any other parameter p_j occurs ($j \neq i$), it is replaced by another symbol, "Ø". As an illustration the following PSR for the labelled directed graph G in figure 16 and its unconditional first order ISRs (one for each node) are given.

```
G = {p:event, parms:{a, b, c},
     body:{{p:J, from:a, to:a}, {p:J, from:b, to:b},
           {p:J, from:c, to:c}, {p:K, from:a, to:b},
           {p:K, from:b, to:c}, {p:K, from:c, to:a},
           {p:L, from:a, to:a}, {p:L, from:b, to:b}}}
```

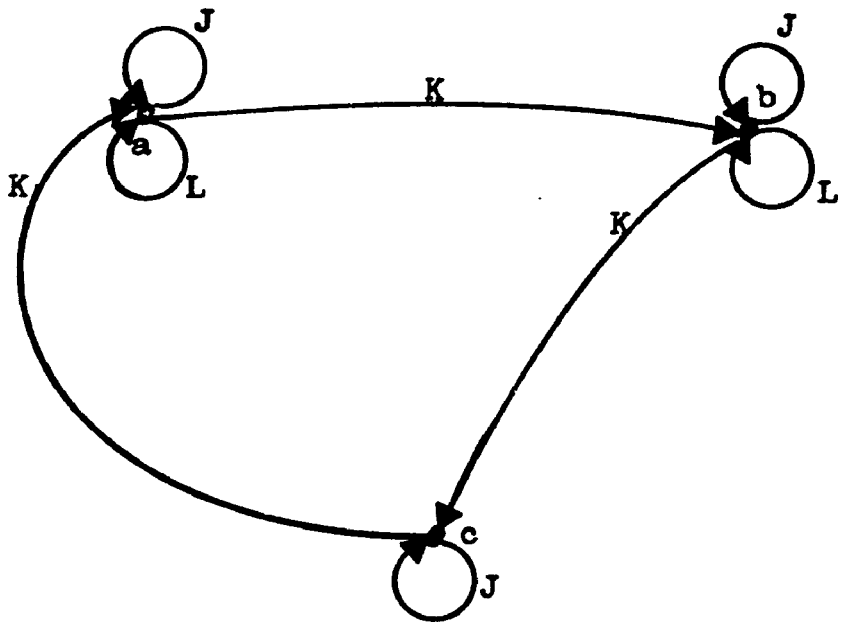


Figure 16. An example of a labelled, directed graph which can be represented in a unique canonical form.

(A3.1)

$ISR^1(a)$ = ({p:J,from:1, to:1}, {p:K, from:1, to:Ø},
{p:K, from:Ø, to:1}, {p:L, from:1, to:1}) (A3.2)

$ISR^1(b)$ = ({p:J, from:1, to:1}, {p:K, from:1, to:Ø},
{p:K, from:Ø, to:1}, {p:L, from:1, to:1}) (A3.3)

$ISR^1(c)$ = ({p:J, from:1, to:1}, {p:K, from:1, to:Ø},
{p:K, from:Ø, to:1}) (A3.4)

Notice that each of these ISR^1s is written as an ordered set (an n-tuple) because its elements (relations) are lexicographically ordered, where the predicate, object type, "1" and "Ø" symbols are necessarily assigned some arbitrary ordering values.

At this point, the set of unconditional first order ISRs, $\{ISR^1(p_i): i = 1, 2, \ldots, n\}$, is partitioned into natural equivalence classes on the basis of identical bags of ordered relations. The equivalence class containing $ISR^1(p_i)$ is denoted $[ISR^1(p_i)]$, and the equivalence class of parameters including p_i with identical ISR^1s is denoted $[p_i]_1$. In the current example, since $ISR^1(a) = ISR^1(b)$, these two belong to the same class, so $[a]_1 = [b]_1 = \{a,b\}$. $ISR^1(c)$ is the sole member of $[ISR^1(c)]$ and thus $[c] = \{c\}$. Because c is the only member of [c], it is <u>identifiable</u> on the basis of ISR^1s; c is identified by assigning to it the new index label "1", indicating that it is the parameter of all those thus far identifiable which has the lowest lexicographically ordered ISR. The next parameter which is identifiable with the lowest ISR value will be renamed "2",

the one after that "3", and so on.

In order to continue the process of identifying additional parameters, first order ISRs for a and b conditional on the prior identification of c, denoted $ISR^1(a:\{c\})$ and $ISR^1(b:\{c\})$, must be computed. Because $[a]_1 = [b]_1$ and this set is associated with the lowest (albeit only) lexicographically ordered common ISR^1, a will be replaced wherever it occurs in its own conditional ISR^1 by the next available index label, "2", and so will b in its own conditional ISR^1. Any other parameter p_i occurring in $ISR^1(a:\{c\})$ or $ISR^1(b:\{c\})$ will be replaced by its most recently produced ISR. Thus, in the current example, the following conditional first order ISRs for a and b are:

$$ISR^1(a:\{c\}) = (\{p:J, from:2, to:2\}, \{p:K, from:2, to:ISR^1(b)\}, \{p:K, from:1, to:2\}, \{p:L, from:2, to:2\}) \quad (A3.5)$$

$$ISR^1(b:\{c\}) = (\{p:j, from:2, to:2\}, \{p:K, from:2, to:1\}, \{p:K, from:ISR^1(a), to:2\}, \{p:L, from:2, to:2\}) \quad (A3.6)$$

As before, the current set of ISRs for unidentified parameters, $\{ISR^1(a:\{c\}), ISR^1(b:\{c\})\}$, is partitioned, and the induced partition of the set of unidentified parameters, {a, b}, is produced. Here, because $ISR^1(a:\{c\}) \neq ISR^1(b:\{c\})$, $\{a\}=[a]_1 \neq [b]_1=\{b\}$, and a and b are identifiable. It is determined that the (arbitrary) lexicographic order value of the most current ISR of a is less than that of b, denoted $L[a]_1 < L[b]_1$. Thus, a is assigned the next

successive index ("2") as its label and, finally, b is assigned the index label "3".

Having succeeded in identifying all parameters, the CF for the PSR G, denoted CF(G), is immediately derived by replacing each parameter by its new integer label and sequentially ordering the relations in the body of G which now have no symbols other than literal alphabetic constants. Thus, CF(G) might simply be represented as follows, without any parameters and under the assumption of some arbitrary lexicographic ordering rule:

```
CF(G) = ({p:J, from:1, to:1}, {p:J, from:2, to:2},
         {p:J, from:3, to:3}, {p:K, from:1, to:2},
         {p:K, from:2, to:3}, {p:K, from:3, to:1},
         {p:L, from:2, to:2}, {p:L, from:3, to:3}).
                                                    (A3.7)
```

It is immediate from the parameter-independent nature of the foregoing procedure than any other PSR H which is equivalent to G, denoted $H \equiv G$ and meaning that H(*)G and G(*)H (see chapter 4), would be converted to a CF in exactly the same sequence of steps and, thus, $G \equiv H$ if and only if CF(G) = CF(H).

The procedure which was used for this simple example required only first order ISRs in order to identify all of the parameters. Unfortunately, the CFs of certain PSRs cannot be derived this easily. Specifically, this is true of PSRs in which several parameters represent objects which are, at the level of ISR^1s, equivalent to one another be-

cause they participate in the same relations with one another or with other unidentified parameters. Such situations arise in the previously described procedure when at least two parameters remain to be identified but are indistinguishable on the basis of their most recent conditional ISR[1]s despite the substitution of distinct constants for each previously identified parameter.

In general this deadlock situation (impasse) is characterized by the simultaneous presence of m lexicographically ordered classes, C_1, C_2, ..., C_m, of parameters equivalent by their conditional ISR[1]s when all previously identified parameters have been replaced by their identification index numbers. Suppose $C_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,n_i}\}$, for i = 1, 2, ..., m and that $ISR^1(p_{i,j}:Q) = ISR^1(p_{i,k}:Q)$ for all i, j, k where $Q = \{p_{q_1}, p_{q_2}, \ldots, p_{q_h}\}$ is the set of all h previously identified parameters in the order of their identification. In order to eliminate the impasse in the identification procedure, successively higher order ISRs must be computed and considered as potential bases for distinguishing among equivalent parameters. To explicate this procedure, it is necessary to introduce the concepts of a conditional PSR and higher order parameter ISRs.

At any point in the identification procedure when such an impasse is encountered, the original PSR, say G, is rewritten as a PSR conditional on the set Q of previously identified parameters, denoted G:Q, by replacing each parameter p_{q_j} in Q by j, its identification index. Then, an

attempt is made to resolve the impasse by arbitrarily assigning to one parameter $p_{1,j}$ in C_1 the next available index number (h+1, since $|Q| = h$), computing a new conditional PSR reflecting this identification, and continuing to compute ISRs for the remaining parameters. If the result of subsequent parameter identifications are indistinguishable (in terms of the resultant conditional PSRs) regardless of which parameter $p_{1,j}$ in C_1 is indexed as h+1, it follows that whichever one is selected to be indexed h+1 is immaterial to the identification process and does not affect the CF produced.

Procedurally, this process is easily described. Let $ISR^2(p_{k,j}:Q/\{p_{1,j}\})$ denote the <u>second order ISR</u> of $p_{k,j}$, conditional on the prior identification of the parameters in Q and on the arbitrary assignment of the next available index number to $p_{1,j}$. The <u>order</u> of such ISRs is one plus the number of parameters arbitrarily indexed (or classes <u>enumerated</u>) to resolve the impasse situation. To continue the identification process upon arriving at the previosly described impasse situation, the following procedure is followed.

<u>Step 1</u>: For each $p_{1,j}$ $(j=1,\ldots,n_1)$, assign the next available index number $(|Q|+1)$ to $p_{1,j}$ and rewrite the PSR G as $G:(Q \cup \{p_{1,j}\})$, the PSR conditional on the prior identification of Q followed by the identification of $p_{1,j}$. This produces n_1 distinct conditional PSRs.

<u>Step 2</u>: For each unidentified parameter in each condi-

tional PSR, compute a new higher order conditional ISR. Where possible, subsequently index all parameters which become identifiable by their new conditional ISRs, compute new conditional PSRs, and reiterate if any parameters are still unidentified.

Step 3: If every parameter in any conditional PSR is identified, go to step 4 to choose the CF. Otherwise, it must be true that the identification process for every conditional PSR has reached an impasse. In this case, the new parameter equivalence classes for any PSR $G'_j = G(Q \cup \{p_{1,j}\})$ are $C'_1, C'_2, \ldots, C'_{m'}$. For each $p'_{1,k}$ in C'_1, compute a new conditional PSR $G'' = G:(Q\ \{p_{1,j}, p'_{1,k}\})$, where $p'_{1,k}$ is replaced by the next available index $= |Q| + 2$. Then, for each such conditional PSR, return to step 2. If you arrive at this point in the future, simply extend the technique to higher order conditional PSRs by separately evaluating each PSR which is conditional on all previously arbitrarily indexed parameters and enumerating each parameter in the currently lowest ordered parameter equivalence class, (C''_1, $C'_1{}''$, and so on) and continuing at step 2.

Step 4: For each conditional PSR arising from a distinct series of arbitrary assignments of indexes to one parameter from each lowest lexicographically ordered parameter equivalence class, compute a CF by lexicographically ordering the relations in the body of the PSR conditional upon the index assignments previously chosen for each parameter. Considering this set of relations as a single string, the low-

est lexicographically ordered set is chosen as the CF.

Because this procedure is deterministic, guaranteed to halt, and independent of the original symbols used for each parameter, because each original parameter in the PSR G is replaced by a unique index, and because the CF contains all relations in the original PSR, it follows that two PSRs will have identical CFs if and only if they are equivalent.

To see how it may be necessary to enumerate distinct conditional PSRs for each parameter in C_1 (then C_1', then C_1'') when all ISRs fail to distinguish them, consider the following PSR H for an undirected, unlabelled graph (figure 17) with five nodes (u, v, w, y, z) with the classes C_1 = {u, v, w} and C_2 = {y, z} of ISR^1-equivalent parameters:

H = {p:event, parms:{u,v,w,y,z},
body:{{p:arc, 1:u, 1:y}, {p:arc, 1:u, 1:z},
{p:arc, 1:v, 1:y}, {p:arc, 1:v, 1:z},
{p:arc, 1:w, 1:y}, {p:arc, 1:w, 1:z},
{p:arc, 1:y, 1:z}}}. (A3.8)

For each parameter in C_1, one distinct conditional PSR is computed. H:{u} may be considered as an example. In this conditional PSR, u is replaced by 1 (the next available index), and new ISR^2s are computed for the remaining parameters. As a result, two new classes C_1'={y,z} and C_2'={v,w} of parameters equivalent by second-order conditional ISRs are produced. Again, it is necessary to compute conditional PSRs for each parameter (y and z) in C_1', and reiterate. The PSR based on assigning the index 2 to y, H:{u,y} is

adequately conditionalized to allow the identification of z with index 3. However, a new impasse is reached with $C_1'' = \{v, w\}$. A conditional PSR for each parameter in C_1'' is computed and the identification process continues, to completion immediately in this case.

In this example, the first enumeration of conditional PSRs for parameters in C_1 generated 3 separate PSRs; for each of these, 2 enumerations of parameters in C_1' were needed; and finally, for each of these, 2 PSRs resulting from enumerating C_1'' were computed. Thus 3 x 2 x 2 = 12 distinct PSRs were separately converted to CFs. For each of these, exactly the same CF results, and is given by:

CF(H) = ({p:arc,1:1,1:2}, {p:arc, 1:1, 1:3},
{p:arc, 1:2, 1:3}, {p:arc, 1:2, 1:4},
{p:arc, 1:2, 1:5}, {p:arc, 1:3, 1:4},
{p:arc, 1:3, 1:5}). (A3.9)

If any of the distinct CFs had a lower lexicographic order value, it would be chosen as CF(H) instead. In fact, the

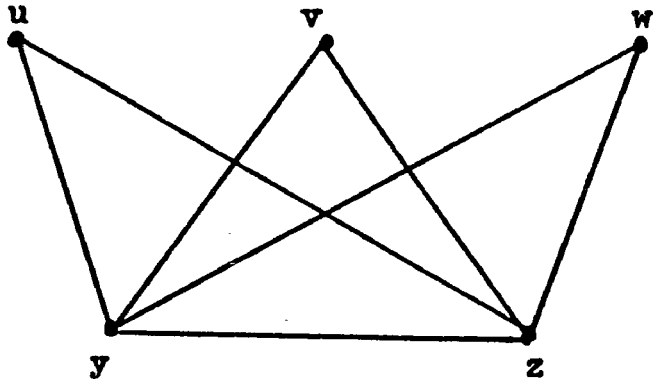


Figure 17. An unlabelled, undirected graph with two classes of structurally indistinguishable nodes: $C_1=\{u,v,w\}$ and $C_2=\{y,z\}$.

number of distinct automorphisms on H (isomorphisms between H and H) is 12, exactly the number of distinct conditional PSRs resulting from the successive enumerations of C_1,C_1', C_1''.

Thus, while this appendix has demonstrated the existence of an effective procedure to convert PSRs into CFs, it requires in cases with structural symmetries (classes of structurally equivalent objects) an enumeration of possible index assignments within each successive lowest lexicographically ordered class of structurally indistinguishable parameters. As a practical basis for determining graph isomorphism in cases where numerous symmetries (automorphisms) occur, it may actually be preferable to resort to heuristic techniques like that of Unger (1964), which uses first-order ISRs to identify corresponding classes of equivalent parameters in two compared graphs and then successively generates and tests possible one-to-one bindings in an attempt to establish isomorphism, or the non-deterministic procedure of Corneil and Gottlieb (1970) which computes, in addition to first order ISRs, some properties of each node based on connections among equivalent parameters (akin to second order ISRs). On the other hand, where computational systems are to be designed to recognize the occurrence of complex contingencies automatically or to retrieve records associatively on the basis of structured relational characteristics, the proposed procedure provides a feasible and effective technique for generating unique canonical forms.

APPENDIX IV: A FORMAL ANALYSIS OF THE PRINCIPAL RESULTS

The purpose of this appendix is to provide a formal treatment of the major results of this paper which have been treated informally in the body of the text. The results considered are: (1) the relationship of PSRs to formulae and the relationship of TXs to theorems in the first-order predicate calculus; (2) the computation universality of the SR system; (3) advantages of PSRs over predicate calculus formulae and typical graph representations of relational structures; (4) the effectiveness of the IM algorithm for abstracting pattern characteristics; (5) the effectiveness of the IM algorithm for deciding if S(*)T and its relationship to previous algorithms for the graph monomorphism problem; and (6) the effectiveness of the two-step rule-learning algorithm for rule induction.

(1) Relationship of the SR system to first-order predicate calculus

As Barrow, et al. (1972) have previously noted, conjunctive relational descriptions of pictures are equivalent to conjunctive formulae in the first-order predicate calculus. Although PSRs are more general than the representations considered by Barrow, et al., their argument carries over almost directly.

The explicit PSR Q = {p:event, parms:{$c_{1,1}$, ..., c_{N,m_N}}, body:{{p:Q_1, 1:$c_{1,1}$, ..., m_1:c_{1,m_1}} : 1 = 1, ..., N}, where the order of each predicate Q_1 is m_1, is representationally equivalent to the instantiated form (IF) IF(Q) = Q_1($c_{1,1}$,

$\ldots, c_{1,m_1}) \;\&\; \ldots \;\&\; Q_N(c_{N,1}, \ldots, c_{N,m_N}) = \prod_{i=1}^{N} Q_i(c_{i,1}, \ldots, c_{i,m_i})$ of the variable form (VF) $VF(Q) = \prod_{i=1}^{N} Q_i(w_{i,1}, \ldots, w_{i,m_i})$. The parameters $c_{1,1}, \ldots, c_{N,m_N}$ of the PSR Q are the constants of IF(Q) and the instances of the free variables $w_{1,1}, \ldots, w_{N,m_N}$, respectively, of VF(Q). The representational equivalence of Q and IF(Q) is denoted $Q \leftrightarrow IF(Q)$. Both IF(Q) and VF(Q) are conjunctive formulae or clauses and each $Q_i(c_{i,1}, \ldots, c_{i,m_i})$ is called a term in IF(Q).

Suppose Q is a PSR as defined above and Q' is similarly defined; i.e., Q' = {p:event, parms:{$c'_{1,1}, \ldots, c'_{m'_{N'}}$}, body:{{p:$Q'_1$, 1:$c'_{1,1}$, ..., m'_1:c'_{1,m'_1}} : i=1,...,N'}} is an explicit PSR where the $c'_{i,j}$ are constants in IF(Q') which are instances of the free variables $w'_{1,1}, \ldots, w'_{N',m'_{N'}}$ of VF(Q'). Furthermore, suppose that there is some set J of index 4-tuples such that $(i,j,i',j') \in J \Leftrightarrow c_{i,j} = c'_{i',j'}$. Then the TX T = {p:TX, parms:{$c_{1,1}, \ldots, c_{N,m_N}$}, C:body(Q), R:body(Q')} is representationally equivalent to the universally quantified implication VF(T) = $(\forall w_{1,1}, \ldots, w_{N,m_N})$ $VF(Q) \Rightarrow VF(Q')$ & ($\forall$ i', j' such that $\forall$ i, j, $(i,j,i',j') \notin J$) $w'_{i',j'} = c'_{i',j'}$ & ($\forall (i,j,i',j') \in J$) $w_{i,j} = w'_{i',j'}$. That is, for all (i,j,i',j') in J, the variable $w_{i,j}$ in VF(Q) is the same as that in VF(Q'), and for all $c'_{i',j'}$ occurring only in the response component Q', $w'_{i',j'}$ remains instantiated as $c'_{i',j'}$ in VF(T).

A PSR S = {p:event, parms:{$s_{1,1}, \ldots, s_{M,r_M}$}, body:

$\{\{p{:}R_i, 1{:}s_{i,1}, \ldots, r_i{:}s_{i,r_i}\} : i = 1, \ldots, M\}$ matches the PSR Q (as defined above) in the all-or-none sense, $S(*)Q \Leftrightarrow$ ($\exists f$, a 1-1 binding function) $f{:}\{c_{1,1}, \ldots, c_{N,m_N}\} \rightarrow \{s_{1,1}, \ldots, s_{M,r_M}\}$ & $(\forall i = 1, \ldots, N)$ $(\exists k \in \{1, \ldots, M\})$ $\{p{:}Q_i, 1{:}f(c_{i,1}), \ldots, m_i{:}f(c_{i,m_i})\} \subset \{p{:}R_k, 1{:}s_{k,1}, \ldots, r_k{:}s_{k,r_k}\}$. The PSR S matches the (contingency of the) TX T (as defined above) if S(*)Q. When S matches T (under some binding function f) the response of T is the PSR $T_{S,f}$ = $\{p{:}event, parms{:}\{f(c_{1,1}), \ldots, f(c_{N,m_N}), b_1, \ldots, b_n\}, body{:}$ $\{\{p{:}T_i, 1{:}g(z_{i,1}), \ldots, q_i{:}g(z_{i,q_i})\} : i = 1, \ldots, H\}\}$, where $(\forall i,j)$ $g(z_{i,j}) = f(c_{i',j'})$ if $z_{i,j} = c_{i',j'}$ for some i' and j' and, in all other cases, $g(z_{i,j}) = b_i$, (where $z_{i,j} = b_i$, for some i'). This PSR $T_{S,f}$ is, as previously shown, representationally equivalent to an IF, namely $T_{S,f} \leftrightarrow IF(T_{S,f})$.

(2) Computation universality of the SR System

Although in any practical problem, memory and processing constraints may be important, the universality of a computational framework always requires infinite memory capacity and is independent of computing time (if finite). Thus, in order to expedite the exposition, the assumption is made that the active memory of the SR system is infinite and retains whatever is put into it indefinitely. It is then straightforward to show that any computing system represen-

ted as a set of Post normal productions (Minsky, 1967) can be effected by a comparable set of TXs which are invoked in response to the recognition that the contents of active memory match their contingencies.

Specifically, the following conventions are observed. The computing system manipulates (derives) strings of vocabulary symbols which are momentarily (at some time $t \in \{1,2,3, \ldots\}$) in active memory, C_t, before being transformed into new strings (which are subsequently stored in memory, C_{t+1}). The set of vocabulary symbols is V and the set of all strings of symbols from V is denoted V*. The symbols [,] are delimiter symbols not in V and are used to enclose every string from V* maintained in C_t. If $R = r_1 \ldots r_m$ and $S = s_1 \ldots s_n$ are elements of V*, R <u>concatenated with</u> S is RS $= r_1 \ldots r_m s_1 \ldots s_n$ and $RS \in V^*$. Any computing system is represented in this framework by a finite set of productions of the form $(\forall X \in V^*)\ [AX] \rightarrow [XB]$ where $A \in V^*$ & $B \in V^*$. Each such production is implemented by the production system to insure the following: $(\forall X \in V^*)(\forall t)\ [AX] \in C_t \Rightarrow [XB] \in C_{t+1}$. By convention, computing begins at t=1 with a single string [a] in memory; i.e. $C_1 = \{[a]\}$.

To show that such behavior (inferences) can be performed by TXs in the SR system, it is necessary to introduce a fifth-order relation with predicate term p:S (for "string") or, correspondingly, to employ a fifth-order instantiated form $S(\alpha_1, \ldots, \alpha_5)$ with the following meaning: α_3 is the <u>name</u> of the string defined by this IF; it comprises two suc-

cessive substrings, named α_1 and α_2; the first and last symbols of the string named α_3 are named α_4 and α_5, respectively. Briefly, any string in memory is first coded as a set of relations defining which characters occur, which parameters name these, and which symbols directly succeed one another in the string. Then TXs (or equivalent implications) are used to form and name every possible subsequence (string) of symbols using an appropriate $S(\alpha_1,\ldots,\alpha_5)$ IF. Finally, each Post production is expressed as a TX in terms of the concatenation of the context symbol strings (A and B) and the name of a string X (a parareter in the contingency component) which completes the string (i.e. AX is immediately enclosed within the delimiters and so is XB).

Specifically, the following conventions are used. The starting string is [a], $a \in V$. Its PSR is equivalent to the following VF:

$$C([,S_{[}) \;\&\; C(a,\alpha) \;\&\; C(],S_{]}) \;\&\; N(S_{[},\alpha) \;\&\; N(\alpha,S_{]}).$$

Any IF $C(q,\delta)$ means that there is a distinct symbol of type q in a string in memory and that symbol is named uniquely δ. An IF of the sort $N(\delta,\lambda)$ means that the symbol (or string) named δ directly precedes that named λ.

To compute and identify each distinct substring of symbols, several "compositional" TXs are needed. These are described in predicate calculus notation (where the previously established equivalence between such forms and TXs is now assumed). As in Chapter 7, special notation is used to identify where new parameters are to be introduced into

response patterns by the SR system to name newly identified objects (instances of variables). Specifically, any response component of a TX T may contain several symbols of the sort $\$_{x_1}, \ldots, \$_{x_n}$. Whenever T is invoked because some S matches its contingency under a binding function f, n unique constants $c_1, \ldots, c_n$ are generated and c_i is substituted for $\$_{x_i}$ (i=1,...,n) in the response $T_{S,f}$. Such a process could be achieved by identifying the c_i with the unique date and time when each is actually required (computed) or with the successive outputs of any of the typical system index generators which produce an infinite sequence of distinct labels (e.g., 1, 2, 3, ...). The special symbol $\emptyset$ is interpreted as a globally defined constant, meaning nil, which is not an element of V. The four rules needed, T1 - T4, are:

T1: $(\forall a \in V)(\forall \alpha)\ C(a,\alpha) \Rightarrow S(\emptyset, \alpha, \alpha, \alpha, \alpha)\ \&$ $S(\alpha, \emptyset, \alpha, \alpha, \alpha)$. [Each character named α is used to produce two new strings, each named α, having α as the first or second substring, respectively, and having a as their first and last characters. The other substring, in each case, is nil. The <u>extent</u> of the string named α is a.]

T2: $(\forall a \in V)(\forall b \in V)\ C(a,\alpha)\ \&\ C(b,\beta)\ \&\ N(\alpha,\beta) \Rightarrow$ $S(\alpha, \beta, \$, \alpha, \beta)$. [Any two successive symbols, named α and

β, are combined into a new string named \$. Note, here and hereafter, unqualified variables in the left-hand sides of rules are universally quantified, e.g. α and β in T2. In T2, the first symbol of \$ is named α and the last is named β. The extent of any string (x_1,x_2,x_3,x_4,x_5) is the concatenation of the extents of x_1 and x_2. The extent of $\emptyset$ is empty (the string of no symbols).]

T3: $S(x_1,x_2,x_3,x_4,x_5)$ & $S(y_1,y_2,y_3,y_4,y_5)$ & $N(x_3,y_3)$ & $N(y_3,z) \Rightarrow S(x_3,y_3,\$,x_4,y_5)$ & $N(\$,z)$. [Any two successive non-empty strings named x_3 and y_3 are concatenated in a new string named \$, whose first symbol is named x_4 as in the string named x_3 and whose last symbol is named y_5 as in the string named y_3. Further, since the string named y_3 is succeeded by z, so is the new string named \$.]

T4: $S(x_1,x_2,x_3,x_4,x_5)$ & $S(y_1,y_2,y_3,y_4,y_5)$ & $N(x_3,y_3)$ & $N(z,x_3) \Rightarrow S(x_3,y_3,\$,x_4,y_5)$ & $N(z,\$)$. [As for T3, but here z precedes \$ because it precedes x_3.]

After all satisfiable rules are applied, every subsequence $T' = t_j\ t_{j+1} \cdots\ t_k$ in any string $T = [t_1 \cdots\ t_n]$ ($j \geq 1$ and $k \leq n$) present in memory is the extent of at least one string named x_3 defined by $S(x_1,x_2,x_3,x_4,x_5)$. Thus any Post normal production $[AX] \rightarrow [XB]$, where $A=a_1 \cdots a_n$ and $B=b_1 \cdots b_m$, is equivalently computed by the TX whose predicate calculus equivalent is:

$$\prod_{i=1}^{n} C(a_i,\alpha_i)\ \&\ \prod_{i=1}^{n-1} N(\alpha_i,\alpha_{i+1})\ \&$$

$$S(x_1,x_2,X,X_F,X_L) \;\&\; N(S_{[},\alpha_1) \;\&\; N(X_L,S_{]}) \;\&\; N(\alpha_n,X_F) \Rightarrow$$

$$\prod_{i=1}^{m} C(b_i,\$_i) \;\&\; \prod_{i=1}^{m-1} N(\$_i,\$_{i+1}) \;\&\; N(\$_m,S_{]}) \;\&\; N(X_L,\$_1) \;\&\; N(S_{[},X_F).$$

[The first two product terms specify that A must be in memory. The third says that X must name a string whose first and last symbols are named X_F and X_L, respectively. The fourth through sixth specify that AX must be directly embraced by [and]. On the right-hand side (the response term), the first two product terms specify that B is to be placed in memory and the i-th symbol of B, b_i, is to be assigned a unique name $\$_i$. The third term specifies that b_m directly precedes]. The fourth says that the last symbol of X, named X_L, directly precedes b_1. The last term specifies that the first symbol of X, named X_F, is to succeed [directly.]

Thus, each Post normal production is computable by TXs in the SR system and, as a result, the SR framework is computation universal.

(3) Advantages of PSRs over predicate calculus formulae and typical graphic representations.

There are four sorts of advantages of PSRs over their corresponding predicate calculus or typical graphic representations. The first two concern preferability owing to ease of use (writing), while the last two concern the suitability of PSRs for abstraction problems. These are discussed in turn.

First, the ability to represent numerous relations (predicate calculus terms) in a compact form in the SR system--as illustrated by the compact relation {p:line, node: x1, node:x2, node:x3} representing the conjunctive product

$$\prod_{\substack{i=1,2,3 \\ i\neq j=1,2,3}} L(x_i,x_j)$$

of six distinct terms, where $L(x_i,x_j)$ is the predicate calculus term denoting that x_i and x_j are "nodes" connected by a "line"--seems to simplify the use of such systems by people.

Second, another reason why PSRs appear easier to use than corresponding formulae in predicate calculus is that every relation in a PSR may carry as much information as needed in the choice of object term symbols to clarify the semantic role of each object cited. For example, while the VF action(give, Mary, Bill, Sam, happy, park, none, e_1) is completely uninterpretable by someone who is unfamiliar with the details of the specific semantic roles of each object position in the VF with predicate action, it would appear that this is much less true for the corresponding PSR relation, {p:action, verb:give, agent:Mary, recipient:Bill, object:Sam, agent's purpose was to make her/him feel:happy, location of the transaction:Park, expense to the recipient to obtain the object:none, name of this event:e_1}. In a similar manner, a relation in a PSR may take any number of additional modifying object terms if desired without confusion--for example, the object term "original cost of object:

$7.50," could be added to the PSR body without difficulty--but to allow for such changes in the predicate calculus framework, it is necessary to use variable forms in which every currently unspecified object is represented by an existentially quantified variable, whose quantifer can be removed later if a specific substitution is desired. Further, variable forms corresponding to symmetric n-ary relations or to relations which are symmetric over n of m object positions, may be easily represented in the SR system because each relation is represented as a set, rather than a list, in which the symmetries are self-documented, so to speak, by repeated object types.

Third, the use of "in" and "name" predicate object type symbols in PSRs to indicate, respectively, that a relation belongs to a particular set and that the relation itself may be referred to by a particular name symbol, are devices which facilitate the representation of hierarchical relational structures in PSRs without resorting to the type of nesting of predicate terms characteristic of higher-order predicate calculus. On the other hand, given the aforementioned equivalence of PSRs and conjunctive clauses of the first-order predicate calculus, the desirability of this device is formally independent of which of the two representational frameworks is employed. Nevertheless, because of the ease with which numerous object terms can be introduced into compact PSR relations, it seems that a greater advantage can be obtained from these devices in the SR framework. Further

since the abstraction procedures used for learning in the SR system operate by matching sets of equivalent relations, the use of these devices to eliminate explicit nesting of relations insures that abstraction can be performed without recursive matching procedures which would otherwise be apparently required.

Exactly what function is served by these "in" and "name" devices is to allow the interpretation that two PSRs S and T are such that S(*)T, where $S'=\{S_1,\ldots,S_N\}$ and $T' = \{T_1,\ldots,T_M\}$ are <u>sets</u> of relations (e.g. {p:Q, i:x_i : for some i}) whose conjunctive products equal S and T, respectively, if and only if $(\forall i)(\exists j)\ T_i \subset S_j$ or, recursively, $(\forall i)(\exists j)\ S_j(*)T_i$. If S and T are trees, for example, where the fact that any node q is the descendant of a node r is represented by {p:node, name:q, in:r}, then S(*)T $\Leftrightarrow$ T is a subgraph (a subtree) of S. Because the orientation in this paper is toward "extracting" information to induce rules, it was felt that this type of construct should be used. Thus, the rule for S(*)T as defined above is actually a graph structure generalization of the linear (string) matching rules for formal grammars, where the string $A = [a_1 \ldots a_n]$ matches the template $B = [B_0\ X_1\ B_1\ X_2 \ldots X_M\ B_M]$ only if A can be expressed as B for some choice of $X_i \in V^*$ (where each $B_i \in V^*$ is a prespecified constant string).

Conversely, if one wished to preclude such matching on the basis of recursive set containments, one could insist that S (or any of its contained sets) have the same number

of elements as T (or any of its corresponding contained sets) by including relations like {p:count=M, in:T'} in the set T' and {p:count=N, in:S'} in the set S' so that S(*)T only if they have the same number of elements. If this is done for each level of nested sets in the representation, S(*)T ⇒ T(*)S ⇒ they are equivalent (S=T).

Fourth, uniform PSRs provide a symbolic representation (and a straightforward graphic equivalent) which facilitates the abstraction of subrepresentations (subgraphs) common to two or more uniform PSRs (uniform graphs) in a way which no alternative representation scheme known to me can. Specifically, by "fractionating" each parameter (object name, instance) into several distinct symbols which are uniquely associated with each separate occurrence of one original parameter as an object of a predicate in the non-uniform PSR, the potential need to assign two or more correspondents from one representation to one parameter in another partial-matched representation during abstraction is eliminated.

For example, if the two non-uniform representations to be matched had the following compact bodies which employ the binary symmetric "line" predicate, R={{p:line, node:a, node:b, node:c}} and S={{p:line,node:d, node:e}, {p:line, node:f, node:g}}, representing a triangle and a pair of non-intersecting lines, respectively, it would appear that the maximal abstraction representing the fact that both events contain two lines, would require the assignment of two dis-

tinct parameter correspondents from S to one parameter in R. For example, one might establish the following bindings, q={R.a, S.d}, r={R.a, S.f}, S={R.b, S.e}, t={R.c, S.g}, in order to infer the common maximal abstraction {{p:line, node:q, node:s}, {p:line, node:r, node:t}}.

Because allowing many-to-one correspondences in matching greatly expands the number of possible binding functions and is, as well, unesthetic (as is allowing several nodes in one graph to correspond to the same node in a second graph while searching for common subgraphs), uniform representations provide a distinct advantage over the alternatives. This is a result of Theorem 1, below, whose proof requires several definitions which are now introduced.

Definition [Implicit Form]: An IF $Q = P/I(x_{i_1},\ldots,x_{i_{|I|}})$ $= (\exists y_1) \ldots (\exists y_n)\ P(y_1,\ldots,y_n)\ \&\ (\forall j=1,\ldots,|I|)\ y_{i_j} = x_{i_j}$ is implicit in $R = P(x_1,\ldots,x_n)$ if $I = \{i_1,\ldots,i_{|I|} : j < k \Rightarrow i_j < i_k;\ (\forall j)\ i_j \in \{1,\ldots,n\};\ n \geq |I| \geq 1\}$. Further, Q' is implicit in Q if $Q' = P/I'(x_{i_1},\ldots,x_{i_{|I'|}})$ and $I' \subset I$. In these cases, Q and Q' are called implicit forms (in R).

By definition, P may not be either SP (same parameter) or SS (same set). By definition, only SP(x,y) is implicit in SP(x,y) and only SS(x,y) is implicit in SS(x,y). Finally, if Q is implicit in Q' and vice versa, Q = Q'.

The meaning of the implicit form Q is that there is some $|I|$-th-order predicate, denoted P/I, such that $P(x_1,$

..., $x_n) \Rightarrow P/I(x_{i_1}, \ldots, x_{i_{|I|}})$, whenever both forms are fully instantiated. Q represents a proposition about $|I|$ objects related by the predicate P which is obtained by eliminating information about particular instances (x_j) of the free variables y_j ($j \in \{1, \ldots, n\} - I$) and existentially quantifying them.

For example, Q = <u>action</u>/{1,3}(give, Bill), which is implicit in R = <u>action</u>(give, Mary, Bill, money), represents the proposition that "there exists some x_2 and x_4 such that x_2 gives Bill x_4," which is implicit (immediately implied) by the proposition R that "Mary gives Bill money." The significance of implicit forms derives from their use in abstractions; e.g. as defined subsequently, Q is an abstraction of R, denoted R(*)Q.

<u>Definition [Uniform Clause]</u>: Given any fully instantiated clause $C(y_{i,1}, \ldots, y_{m,n_m}) = \prod_{i=1}^{m} P_i(y_{i,1}, \ldots, y_{i,n_i})$, where $(\forall i)\ P_i \notin \{SS, SP\}$, the corresponding <u>uniform clause</u> $C^*(z_1, \ldots, z_r) = P_1(z_1, z_2, \ldots, z_{n_1})$ & $P_2(z_{n_1+1}, \ldots, z_{n_1+n_2})$ & ... & $P_m(z_{n_1+n_2+\ldots+n_{m-1}+1}, \ldots, z_{n_1+n_2+\ldots+n_m})$ & $\prod_{(z_j,z_k) \in R} SP(z_j, z_k)$ & $\prod_{D \in T} D$, where: $r = n_1+n_2+\ldots+n_m$; $(\forall j=1,\ldots,r)(\forall k = 1, \ldots, r)$ $z_j = z_k \Rightarrow j=k$ (the z_j are all distinct); $R = \{(z_j, z_k)$: z_j has replaced $y_{i,j'}$ in $P_i(y_{i,1},\ldots,y_{i,n_i})$ and z_k has replaced $y_{g,k'}$ in $P_g(y_{g,1},\ldots,y_{g,n_g})$ and $y_{g,k'} = y_{i,j'}$ (are the same constants or parameter symbols)$\}$; $SP(z_j, z_k)$ means that the distinct constants z_j and z_k represent the same constant in

in $C(y_{1,1},\ldots,y_{m,n_m})$; and, finally, T is the set of all distinct forms implicit in any of the $P_i(z_{q_i+1}, z_{q_i+2}, \ldots, z_{q_i+n_i})$, where $q_i = \sum_{j=1}^{i-1} n_j$. Moreover, any conjunctive clause directly derivable from a uniform clause by successive applications of the conjunction-out rule [A & B $\Rightarrow$ A] is also a uniform clause.

Definition [Equivalent]: Two conjunctive clauses $C(x_1,\ldots,x_n)$ and $D(z_1,\ldots,z_n)$, uniform or non-uniform, are equivalent if there exists a bijective mapping $f:\{x_1,\ldots,x_n\}\rightarrow\{z_1,\ldots,z_n\}$ such that $C(x_1,\ldots,x_n) = D(f(x_1), \ldots, f(x_n))$.

Definition [Abstraction]: The uniform clause Q is a common abstraction of N conjunctive clauses R_1, R_2, ..., R_N ($N\geq 1$) if Q is equivalent to conjunctive clauses S_1, S_2, ..., S_N and for all $i=1,\ldots,N$, each term in S_i is identical to one term in R_i*, the uniform clause corresponding to R_i. (Equivalently, S_i is produced from R_i* by successive applications of the conjunction-out rule.) When it is asserted that Q is an abstraction of R_i, denoted $R_i(*)Q$, it may be assumed that Q and R_i are conjunctive clauses.

Definition [Abstraction Set]: The abstraction set of $R = \{R_1,\ldots,R_N\}$, denoted A(R), is $\{Q: (\forall R_i\in R)\ R_i(*)Q\}$.

Definition [Maximal Abstraction]: Q is a maximal abstraction of R if $(\forall Q'\in A(R))\ Q'(*)Q \Rightarrow Q(*)Q'$.

Definition [Equality]: $Q(*)Q'$ and $Q'(*)Q \Leftrightarrow Q=Q'$, i.e. Q equals Q' (are equivalent clauses).

Definition [Submaximal Abstraction]: Q is a submaximal

abstraction of R if $Q \in A(R)$ and Q is not maximal, i.e. for some Q' in A(R), $Q' \neq Q \,\&\, Q'(*)Q$.

Definition [Minimal Complete Abstraction Set]: The minimal complete abstraction set of R is $A^*(R) = \{Q : Q \text{ is a maximal abstraction of } R\}$.

Definition [Uniform Graph]: The uniform graph $U(Q) = (X,A,P)$ of a PSR with uniform representation Q = {p:event, parms:$\{x_1,\ldots,x_n\}$, body:$\{\{p:(Q_i,j_i), 1:x_{h_i}\} : i=1,\ldots,w\} \cup \{\{p:SP, 1:x_{j_i}, 1:x_{k_i}\} : i=1,\ldots,s\} \cup \{\{p:SS, 1:x_{m_i}, 1:x_{r_i}\} : i=1,\ldots,t\}$} is constructed from Q as follows. The node set $X = \{x_1,\ldots,x_n\}$. The arc set $A = \{(SS,y,z) : \{p:SS, 1:y, 1:z\}$ is in the body of Q$\} \cup \{(SP,y,z) : \{p:SP, 1:y, 1:z\}$ is in the body of Q$\}$, where SS and SP are labels on arcs connecting nodes y and z. Finally, the node property set P $= \{P(x) : x \in X\}$ where $(q,j) \in P(x)$ if $\{p:(q,j), 1:x\}$ occurs in the body of Q, meaning that x occurs as the j-th object under the predicate q. (Although at this point only one property is in P(x) for each x because of the way uniform PSRs are generated, when rule induction is discussed later, additional properties will be needed.)

Definition [Common Subgraph]: A uniform graph $G^* = (X^*,A^*,P^*)$ is a common subgraph of the uniform graphs $G_i = (X_i,A_i,P_i)$, denoted $G^* \subset^* G_i$ $(i=1,2,\ldots,N)$ if $(\forall i=1,\ldots,N)$ $(\exists f_i)$ $f_i : X^* \rightarrow X_i$ is 1-1 such that $(\forall t \in \{SS,SP\})$ $(t,x,y) \in A^* \Rightarrow (t,f_i(x),f_i(y)) \in A_i$ and $P^*(x) \subset P_i(f_i(x))$, where $P^* = \{P^*(x) : x \in X^*\}$ and $P_i = \{P_i(x) : x \in X\}$. If G_1 and G_2 are subgraphs of one another, they are equivalent, $G_1 = G_2$.

Definition [Representationally Equivalent]: If R is a conjunctive clause in the predicate calculus (with no terms implicit in any other terms and with no SP(x,y) forms) and S is a PSR (with no relations contained in any others and with no SP or SS relations) with corresponding uniform clause R* and uniform PSR S*, respectively, then R and S and R* and S* are representationally equivalent, denoted $R \leftrightarrow S \leftrightarrow R^* \leftrightarrow S^*$ (where $\leftrightarrow$ is an equivalence relation), if there exists a bijective mapping f from the set of instances in R* to the parameters in S* such that the term $Q/I(x_{i_1},\ldots,x_{i_{|I|}})$ is in $R^* \Leftrightarrow (\forall i_j, i_k \in I)$ $\{p:SS, 1:f(x_{i_k}), 1:f(x_{i_j})\} \in$ body(S*) & ($\exists$r, a relation in body(S*)) $\{p:(Q,i_j), 1:f(x_{i_j})\} \subset$ r and $SP(x_k,x_j)$ is a term in $R^* \Leftrightarrow \{p:SP, 1:f(x_k), 1:f(x_j)\} \in$ body(S*). In short, $R \leftrightarrow S$ if they mean the same thing.

Theorem 1: The uniform clause R* is an abstraction of the clauses $R_1, R_2, \ldots, R_N$ if and only if the uniform graph U(S*) of the uniform PSR S* is a common subgraph of the uniform graphs $U(S_1), U(S_2), \ldots, U(S_N)$ where $R^* \leftrightarrow S^*$, $R_1^* \leftrightarrow S_1$, $R_2^* \leftrightarrow S_2$, ..., $R_N^* \leftrightarrow S_N$ are pairs of representationally equivalent uniform clauses and uniform PSRs, respectively, and R_i^* is the uniform clause corresponding to R_i (i=1,...,N).

Proof: [Part I: R* is an abstraction $\Rightarrow U(S^*) \subset^* U(S_i)$, for all i=1,...,N.]

Let $R^* = \prod_{k=1}^{m} P_k(x_{k,1},\ldots,x_{k,n_k})$. Note that P_k may be an implicit predicate of the sort T_k/I_k if $T_k \neq SP$. Similarly let $R_i^* = \prod_{j=1}^{s_i} Q_{i,j}(y_{i,j,1},\ldots,y_{i,j,r_{i,j}})$ with the same inter-

Definition [Representationally Equivalent]: If R is a conjunctive clause in the predicate calculus (with no terms implicit in any other terms and with no SP(x,y) forms) and S is a PSR (with no relations contained in any others and with no SP or SS relations) with corresponding uniform clause R* and uniform PSR S*, respectively, then R and S and R* and S* are representationally equivalent, denoted $R \leftrightarrow S \leftrightarrow R^* \leftrightarrow S^*$ (where $\leftrightarrow$ is an equivalence relation), if there exists a bijective mapping f from the set of instances in R* to the parameters in S* such that the term $Q/I(x_{i_1},\ldots,x_{i_{|I|}})$ is in $R^* \Leftrightarrow (\forall i_j, i_k \in I)$ $\{p:SS, 1:f(x_{i_k}), 1:f(x_{i_j})\} \in$ body(S*) & ($\exists$r, a relation in body(S*)) $\{p:(Q,i_j), 1:f(x_{i_j})\} \subset$ r and $SP(x_k,x_j)$ is a term in $R^* \Leftrightarrow \{p:SP, 1:f(x_k), 1:f(x_j)\} \in$ body(S*). In short, $R \leftrightarrow S$ if they mean the same thing.

Theorem 1: The uniform clause R* is an abstraction of the clauses $R_1, R_2, \ldots, R_N$ if and only if the uniform graph U(S*) of the uniform PSR S* is a common subgraph of the uniform graphs $U(S_1), U(S_2), \ldots, U(S_N)$ where $R^* \leftrightarrow S^*$, $R_1^* \leftrightarrow S_1$, $R_2^* \leftrightarrow S_2$, ..., $R_N^* \leftrightarrow S_N$ are pairs of representationally equivalent uniform clauses and uniform PSRs, respectively, and R_i^* is the uniform clause corresponding to R_i (i=1,...,N).

Proof: [Part I: R* is an abstraction $\Rightarrow U(S^*) \subset^* U(S_i)$, for all i=1,...,N.]

Let $R^* = \prod_{k=1}^{m} P_k(x_{k,1},\ldots,x_{k,n_k})$. Note that P_k may be an implicit predicate of the sort T_k/I_k if $T_k \neq SP$. Similarly let $R_i^* = \prod_{j=1}^{s_i} Q_{i,j}(y_{i,j,1},\ldots,y_{i,j,r_{i,j}})$ with the same inter-

pretations and restrictions.

By the definition of abstraction, $(\forall i)\ R_i(*)R^* \Rightarrow (\forall i)(\exists d_i)(\exists g_i)\ d_i:\{x_{k,j_k} : k=1,\ldots,m;\ j_k=1,\ldots,n_k\} \rightarrow \{y_{i,j,k_j} : j=1,\ldots,s_i;\ k_j=1,\ldots,r_{i,j}\}$ is 1-1 and $g_i:\{1,2,\ldots,m\} \rightarrow \{1,\ldots,s_i\}$ is 1-1 such that $(\forall k=1,2,\ldots,m)\ P_k(x_{k,1},\ldots,x_{k,n_k}) = Q_{i,g_i(k)}(d_i(x_{k,1}),\ldots,d_i(x_{k,n_k}))$ and $(\forall j)\ y_{i,g_i(k),j} = d_i(x_{k,j})$.

Let $U(S^*) = (X^*,A^*,P^*)$ and $U(S_i) = (Y_i,A_i,T_i)$ be the uniform graphs of S^* and S_i respectively $(i=1,\ldots,N)$. Specifically, the node sets $X^*=\{a_{k,j_k} : k=1,\ldots,m;\ j_k=1,\ldots,n_k;\ a_{k,j}=a_{k',j'} \Leftrightarrow x_{k,j}=x_{k',j'},\}$ and $Y_i=\{b_{i,k,j_k} : k=1,\ldots,s_i;\ j_k=1,\ldots,r_{i,k};\ b_{i,k',j'}=b_{i,k',j'} \Leftrightarrow y_{i,k,j}=y_{i,k',j'},\}$ are mutually exclusive sets of symbols which represent the distinct parameters of S^* and S_i and are in 1-1 correspondence with the distinct instances of R^* and R_i^*, namely $x_{k,j}$ in R^* corresponds to $a_{k,j}$ in X^* and $y_{i,k,j}$ in R_i^* corresponds to $b_{i,k,j}$ in Y_i (because of the 1-1 correspondence between the instances of a clause and the parameters in a representationally equivalent uniform PSR and the 1-1 correspondence of the latter with the nodes of the associated uniform graph). Because of these 1-1 correspondences, the chosen symbols are irrelevant and hereafter, without loss of generality, it will be assumed that $a_{k,j}=x_{k,j}$ and $b_{i,k,j}=y_{i,k,j}$.

Now, $(\forall i=1,2,\ldots,N)$ define $f_i:X^* \rightarrow Y_i$ by $f_i(a_{k,j}) = b_{i,g_i(k),j} = d_i(x_{k,j}) = y_{i,g_i(k),j}$. Under these mappings, it can be seen that $U(S^*)$ is a common subgraph of $U(S_i)$ for each i. This follows from the three deductions below show-

ing that each component (arc or property) in U(S*) is also present in $U(S_i)$ under the morphism defined by f_i.

(1) $(SP,a_{k,j},a_{k',j'}) \in A^* \Leftrightarrow$ the term $SP(x_{k,j},x_{k',j'})$ is in $R^* \Leftrightarrow$ the term $SP(x_{k,j},x_{k',j'}) = P_k(x_{k,j},x_{k',j'}) \Leftrightarrow Q_{i,g_i(k)}(d_i(x_{k,j}),d_i(x_{k',j'})) = SP(b_{i,g_i(k),j}, b_{i,g_i(k'),j'})$ is in $S_i \Leftrightarrow (SP,b_{i,g_i(k),j}, b_{i,g_i(k'),j'}) \in A_i$.

(2) $(SS,a_{k,j},a_{k,j'}) \in A^* \Leftrightarrow (\exists I$, a non-null ordered subset of $\{1,2,\ldots,n : n \geq n_k; n_k \geq j; n_k \geqslant j'\}$ $P_k(x_{k,1},\ldots, x_{k,n_k}) = T/I(x_{k,1},\ldots,x_{k,n_k})$ is in $R^* \Leftrightarrow T/I(y_{i,g_i(k),1},\ldots, y_{i,g_i(k),n_k}) = Q_{i,g_i(k)}(y_{i,g_i(k),1},\ldots,y_{i,g_i(k),n_k})$ is in $S_i \Leftrightarrow (SS,b_{i,g_i(k),j},b_{i,g_i(k),j'}) \in A_i$.

(3) $C = (P_k,j) \in P^*(a_{k,j})$ (C is a predicate, object-position two-tuple) $\Leftrightarrow C = (Q_{i,g_i(k)},j)$ (since $P_k(x_{k,1},\ldots, x_{k,n_k}) = Q_{i,g_i(k)}(y_{i,g_i(k),1},\ldots,y_{i,g_i(k),r_{i,g_i(k)}}) \Rightarrow C \in T_i(b_{i,g_i(k),j}) \Rightarrow P^*(a_{k,j}) \subset T_i(b_{i,g_i(k),j})^1$ for all i.

Thus U(S*) is a common subgraph of $U(S_1),\ldots, U(S_N)$ as claimed.

[Part II: $U(S^*) \subset^* U(S_i) \Rightarrow R_i(*)R^*$, for all i.] Let U(S*) and $U(S_i)$ be defined as in Part I. Since $U(S^*) \subset^* U(S_i)$ for all i, it is possible to rename the node sets Y_i such that, if $X^* = \{x_1,\ldots,x_n\}$, $Y_i=\{x_1, x_2,\ldots, x_n, y_{n+1},\ldots, y_{i,r_i}\}$, $A^* \subset A_i$, and the property sets $P^*(x_j) \subset T_i(x_j)$, for all i=1,..., N and j = 1,...,n. Further, it is also given that $R^* \leftrightarrow S^*$ and $(\forall i)$ $R_i^* \leftrightarrow S_i$.

(1) Now, for any $I=\{i_1,\ldots,i_k\}\neq\emptyset$, the term $Q/I(x_{j_1},\ldots,x_{j_k})$ is in $R^* \Leftrightarrow (Q,i_h) \in P^*(x_{j_h})$ and $(\forall h_1,h_2=1,2,\ldots,k)$ $(SS, x_{j_{h_1}},x_{j_{h_2}}) \in A^* \Rightarrow (\forall i=1,2,\ldots,N)$ $(Q,i_h) \in T_i(x_{j_h})$ and

$(\forall h_1,h_2=1,2,\ldots,k)$ $(SS, x_{j_{h_1}}, x_{j_{h_2}}) \in A_i \Leftrightarrow Q/I(x_{j_1},\ldots,x_{j_k})$ is in R_i^*.

(2) Since $A^* \subset A_i$, $(SP,x_j,x_k) \in A^* \Rightarrow (SP,x_j,x_k) \in A_i$

$\Rightarrow$ $SP(x_j,x_k)$ is in R^* and $SP(x_j,x_k)$ is in R_i^*.

Thus, by (1) and (2), any term in the uniform clause $R^* \leftrightarrow S^*$ is also in R_i^*. Thus R^* is an abstraction of R_i^* and an abstraction of R_i (for all i). [QED]

Two conclusions are immediate from Theorem 1.

Corollary: There is a 1-1 correspondence between the set of maximal abstractions $A^*(R)$ of $R = \{R_1,\ldots,R_N\}$ and the set of maximal subgraphs $G^*(S)$ of $S = \{U(S_1),\ldots,U(S_N)\}$ where $R_i \leftrightarrow S_i$, where a uniform graph g is maximal in the set $G(S) = \{g: (\forall i=1,\ldots n)\ g \subset^* U(S_i)\} \Leftrightarrow (\forall g' \in G(S))\ g \subset^* g' \Rightarrow g' \subset^* g$.

Corollary: Any procedure which generates all common subgraphs G(S) of a set of uniform graphs $S = \{U(S_1), \ldots, U(S_N)\}$ where $(\forall i)$ $S_i \leftrightarrow R_i$ is effective for the problem of generating the abstraction set A(R) of $R = \{R_1,\ldots, R_N\}$, because $(\forall Q)$ $Q \in A(R) \Leftrightarrow (\exists T$, a uniform PSR) $T \leftrightarrow Q$ and $U(T) \in G(S)$.

Aside from clarifying the role of uniform PSRs, Theorem 1 provides a basis for understanding the weaknesses of typical graph representations (e.g. Barrow, et al. (1972), Anderson & Bower (1973)) as bases for abstraction procedures. Because each node in these graphs corresponds to a distinct object being described, every relation involving the same object is represented by arcs connecting to the same node.

Thus, in order to abstract a relation (a set of arcs) which is common between two event descriptions, the related objects must correspond (be nodes which are in 1-1 correspondence in the subgraph extraction procedure). Any relations which are common to both descriptions but which relate objects not represented by nodes in 1-1 correspondence are lost. This is why SP arcs are used and each occurrence of the same parameter in distinct predicate-object positions is distinguished by a separate node in the uniform graph.

In addition, if n-ary relations are to be represented ($n \geq 2$), it is necessary either: (1) to use SS arcs, as in uniform graphs, to enable the extraction of subgraphs corresponding to clauses containing non-trivial implicit terms of the sort $P/I(x_1,\ldots,x_k)$ and, in addition, to identify each node in the graph as the 1-1 correspondent of one (predicate, object position) two-tuple arising from the original PSR; or (2) to use one node for each distinct original PSR parameter and one node for each relation (corresponding to the parameter which would occupy the <u>name</u> object position) which is connected by (object-type) labelled arcs to the node corresponding to each related object. Graphs of the second type are essentially two level networks, one level of nodes representing objects and one representing relations. These are similar to those used by Anderson & Bower(1973), except that in their case, higher order nodes were used to conjoin lower-order nodes representing either the conjunction of several relations or the citation of one

relation as the object within another.

In short, both types of typical graph representations have one or more inadequacies with respect to the problem of identifying abstractions by producing subgraphs common to several event graphic representations. The uniform graph, on the other hand, has none and is, therefore, the preferred basis for such problems.

(4) The effectiveness of the IM algorithm for abstracting pattern characteristics.

Let $R = \{R_1, \ldots, R_N\}$ be a set of uniform clauses describing events $E_1, \ldots, E_N$ each of which manifests a common (non-trivial) class characteristic described by the uniform clause C; that is, $(\forall i)\ R_i(*)C$. The effectiveness of the IM algorithm for abstracting C from R is a result of the following:

Theorem 2: The IM procedure applied to the set S of uniform PSRs $S_1, \ldots, S_N$ where $(\forall i)\ S_i \leftrightarrow R_i$ is guaranteed to produce one model whose abstraction is a uniform PSR A and $A \leftrightarrow C$.

Proof: Because C is a common abstraction of $R \Leftrightarrow C \leftrightarrow A$ for some uniform PSR A with uniform graph U(A) which is a common subgraph of $U(S_1), \ldots, U(S_N)$ (by Theorem 1), it is sufficient to show here that the IM procedure produces all non-trivial common subgraphs. This is proved by showing that $(\forall G,$ G a uniform PSR) U(G) is a common subgraph of each $U(S_i) \Rightarrow G$ is equivalent to the abstraction of at least

one model produced by the IM procedure applied to the set S.

For any uniform PSR G, let U(G) = (X,A,P), its uniform graph, be a common subgraph of $U(S_1) = (X_1,A_1,P_1),\ldots, U(S_N) = (X_N,A_N,P_N)$ identified with node correspondence mappings $f_i: X \rightarrow X_i$ where $X=\{x_1,\ldots,x_n\}$, $X_i=\{x_{i,1},\ldots,x_{i,n_i}\}$, $P = \{P(x): x \in X\}$, and $P_i = \{P_i(x): x \in X\}$, and where f_i is 1-1. Without loss of generality, partition each X_i into two sets, $Y_i = \{y_{i,1},\ldots,y_{i,n}\}$ and $Z_i = \{z_{i,1},\ldots,z_{i,k_i}: k_i \geq 0\}$ such that $(\forall i)(\forall j)\ f_i(x_j) = y_{i,j}$. It follows from the definition of subgraph that:

(1) $(\forall i,j,k)$ $\{p:SS,\ 1:x_j,\ 1:x_k\} \in$ body of $G \Leftrightarrow (SS,x_j, x_k) \in A \Rightarrow (SS,y_{i,j},y_{i,k}) \in A_i \Leftrightarrow \{p:SS,\ 1:y_{i,j},\ 1:y_{i,k}\} \in$ body of S_i;

(2) $(\forall i,j,k)$ $\{p:SP,\ 1:x_j,\ 1:x_k\} \in$ body of $G \Leftrightarrow (SP,x_j, x_k) \in A \Rightarrow (SP,y_{i,j},y_{i,k}) \in A_i \Leftrightarrow \{p:SP,\ 1:y_{i,j},\ 1:y_{i,k}\} \in$ body of S_i;

(3) $(\forall i,j)$ $\{p:(Q,k)\ 1:x_j\} \in$ body of $G \Leftrightarrow (Q,k) \in P(x_j) \Rightarrow (Q,k) \in P(y_{i,j}) \Leftrightarrow \{p:(Q,k),\ 1:y_{i,j}\} \in$ body of S_i.

These three results may be summarized simply by the rule, if $SS(x_j,x_k)$, $SP(x_j,x_k)$, or $C(x_j)$ is a relation in the body of G (SS, SP, or unary content types, respectively), then $SS(y_{i,j},y_{i,k})$, $SP(y_{i,j},y_{i,k})$, or $C(y_{i,j})$ is a relation of the same type in the body of S_i.

Step 1: For each arc and node property in U(G), there is an associated abstraction with one relation in a first-step IM model resulting from matching the corresponding relations in each S_i. This is true because for any relation r

in G over x_j and/or x_k (r may be unary or binary), there is a corresponding relation r_i of the same predicate type over the parameters $y_{i,j}$ and/or $y_{i,k}$ in S_i and all sets of matchable relations (one from each S_i) are matched by the IM algorithm. Thus, there is one model M_r resulting from the matching of r_1, ..., r_N under the simultaneous correspondence bindings $w_j = \{S_1.y_{1,j}, \ldots, S_N.y_{N,j}\}$ and/or $w_k = \{S_1.y_{1,k}, \ldots, S_N.y_{N,k}\}$.

Step 2: Because (1) every distinct relation r in G is associated with exactly one distinct relation r_i in S_i, (2) r is represented by the abstraction of M_r under the correspondence bindings w_j and w_k (for some j, k), and (3) any set $\{w_{j_1}, w_{j_2}, \ldots, w_{j_k} : (\forall m=1,2,\ldots,k)\ j_m \in \{1,2,\ldots,n\}\}$ of correspondence bindings is consistent (at most 1-1), it follows that the set of models $V = \{M_r : r \text{ is a relation in } G\}$ is mergeable. Since all sets of mergeable models are merged by the IM procedure, one model $M = (\{w_1, w_2, \ldots, w_n\}, A', R')$ corresponding to the uniform PSR {p:event, parms: $\{w_1, \ldots, w_n\}$, body:A'} results from the merger of the models in V (under the correspondences w_j) and is equivalent to G since it contains all relations in G (when x_j is replaced by w_j).

Thus, for any G, there is one model M representing an equivalent uniform PSR, as claimed. [QED]

(5) The effectiveness of the IM procedure for deciding if S(*)T and its relationship to previous algorithms for the

graph monomorphism problem.

Denote the set of all abstractions (uniform PSRs) produced by applying the IM procedure to the set $\{S_1,\ldots,S_N\}$ by $S_1*\ldots*S_N$. S(*)T (T is an abstraction of S)$\Leftrightarrow$ the uniform graph U(T) corresponding to T is a subgraph of U(S), the uniform graph corresponding to S$\Leftrightarrow(\exists f)$ f: parameter (or node) set of T$\rightarrow$parameter (or node) set of S, f is 1-1, such that every relation in T is also in S when each parameter t in T is renamed f(t)$\Leftrightarrow$ all relations of S are represented in one abstraction A of {S, T} in a model M = (P, A, R) produced by the IM procedure with parameter correspondence bindings P = {{S.f(t), T.t} : t is a parameter in T} $\Leftrightarrow$ the residual of S in M, R(S) = $\emptyset$, since each time a relation in S is incorporated into the abstraction A, it is deleted from R(S) which originally equals the set of all relations in S.

From these equivalences, the following corollary to Theorem 2 follows directly.

Corollary. S(*)T$\Leftrightarrow$ one model of {S,T} produced by the IM procedure has a null residual for S, R(S) = $\emptyset$.

Proof: The IM procedure produces all abstractions of {S,T} (Theorem 2). S(*)T$\Rightarrow$ one abstraction of {S,T} is equivalent to S and this will be represented in a model with R(S) = $\emptyset$. Conversely, $\sim$S(*)T$\Rightarrow$ no such model will be produced.

Previous approaches to the momomorphism problem (Barrow, et al., 1972) have resulted from generalization of

graph isomorphism procedures. Specifically, the sets X_S and X_T of nodes (parameters) of S and T are separately classified according to a variety of "properties" of each node, for example the number of distinct sorts of labelled arcs into and out of each node. If such properties of a node x are represented by the set P(x), a partition Q_T of $X = \{[x]: (\forall x,y \in X_T)\ y \in [x] \Leftrightarrow P(x) = P(y)\}$, while the classification Q_S of X_S nodes $= \{[x_t]: (\forall y \in X_s)(\forall x_t \in X_T)\ y \in [x_t] \Leftrightarrow P(x_t) \subset P(y)\}$. That is, a node y from S is in $[x_t]$ in Q_S if and only if its property set contains all properties of the node x_t. While Q_T is a partition, Q_S defines a partial order on some of the subsets of X_S induced by $\subset$ applied to property sets of the contained nodes. Then, enumeratively, one correspondent in X_S for each x_t in X_T is chosen from $[x_t]$ in Q_S, subject to the 1-1 correspondence requirement. It is then determined whether the two graphs are monomorphic (that of S contains that of T) under these node correspondences.

It is possible to refine the classification Q_S before enumerating by computing additional properties of each node, such as the number of nodes of each type of class in Q_S which are reachable from a given node by a path traversing exactly k arcs. In any case, the same basic procedure applies.

Such an algorithm is, in general, preferable to the IM procedure for deciding whether S(*)T, because it will require

fewer calculations since the predicate-object position two-tuples of all relations in which a parameter participates may be used to eliminate obviously fruitless bindings which would, nevertheless, result from step one relation matches in the IM procedure. The IM procedure, however, is primarily intended for use as a tool for partial-matching (abstraction, subgraph extraction), for which purpose the alternative graph monomorphism procedures are unsuited. It may be considered a fortunate (if impractical) formal result that the same IM procedure required for partial-matching can be used to explicate (in an intuitive manner) and solve the all-or-none matching problem, "does S(*)T?"

(6) The effectiveness of the two-step rule learning algorithm for rule induction.

The notions of pattern characteristic and abstraction are generalized to those of rule and inference, and rules are related to TXs as follows.

Definition [Rule]. A rule is a quantified form F of the sort, $F = (\forall x_1)\ldots(\forall x_n)\ C(x_1,\ldots,x_n) \Rightarrow R(x_1,\ldots,x_n)$ where $C(x_1,\ldots,x_n)$ and $R(x_1,\ldots,x_n)$ are uniform clauses of m terms C_i (i=1,...,m) and k terms R_j (j=1,...,k) respectively.

Definition [Representationally Equivalent]: A TX T = {p:TX, parms:$\{y_1,\ldots,y_n\}$, C:A, R:B} is representationally equivalent to the rule F (as defined above), denoted $F \leftrightarrow T$, if A and B are bodies of uniform PSRs (A' and B', respectively) over the parameters $\{y_i : i=1,\ldots,n\}$ and there is a bi-

jective mapping $f:\{x_1,\ldots,x_n\} \rightarrow \{y_1,\ldots,y_n\}$ under which it is simultaneously true that $C(x_1,\ldots,x_n) \leftrightarrow A'$ and $R(x_1,\ldots,x_n) \leftrightarrow B'$. Because of this equivalence, the TX T may also be called a rule.

Definition [Inference]: Given a set $I = \{F_1,\ldots,F_m\}$ of implications between two instantiated uniform clauses, $F_i = [C_i(x_1,\ldots,x_n) \Rightarrow R_i(x_1,\ldots,x_n)]$ for i=1,...,m, the rule $F^* = [(\forall y_1,\ldots,y_n)\ C^*(y_1,\ldots,y_n) \Rightarrow R^*(y_1,\ldots,y_n)]$ is an inference from I if $(\forall i=1,\ldots,m)(\forall j=1,\ldots,n)$ $(\forall a_j$, an instance of $y_j)$ $C_i(x_1,\ldots,x_n)$ (*) $C^*(a_1,\ldots,a_n)$ & $R_i(x_1,\ldots,x_n)$ (*) $R^*(a_1,\ldots,a_n)$. Equivalently, F^* is an inference from I if $(\forall i=1,\ldots,m)$ F^* & $C_i(x_1,\ldots,x_n) \Rightarrow R^*(x_1,\ldots,x_n)$.

Definition [Implication Uniform Graph]: Let $F = [C(x_1,\ldots,x_n) \Rightarrow R(x_1,\ldots,x_n))$ be an implication of two uniform clauses with uniform graphs $U(C) = (X_C,A_C,P_C)$ and $U(R) = (X_R,A_R,P_R)$, where $X_C = X_R = \{x_1,\ldots,x_n\}$ and where the uniform graph of a uniform clause is the same as that of the representationally equivalent uniform PSR with identical parameters. Construct a new $U(R') = (X'_R,A'_R,P'_R)$ by replacing each occurrence of the symbol x_i by a new, unique symbol y_i throughout U(R), for i=1,...,n. Then the (implication) uniform graph U(F) of F is defined as (X,A,P) where: the node set $X = \{x_1,\ldots,x_n,y_1,\ldots,y_n\}$; the arc set $A = A_C \cup A'_R \cup \{(SP,x_j,y_k)$: j=k or $(SP,x_j,x_k) \in A_C \cup A_R\}$; and $P = \{P(x)$: $x \in X\}$ where $(\forall x \in X_C)(\forall y \in X'_R)$ $P_C(x) \subset P(x)$ & $P'_R(y) \subset P(y)$ & ("C",1) $\in$ P(x) & ("R",1) $\in$ P(y). Further, if the TX T is such that T = [p:TX, parms:$\{x_1,\ldots,x_n\}$, C:A, R:B] where

{p:event, parms:{$x_1,\ldots,x_n$}, body:A}$\leftrightarrow C(x_1,\ldots,x_n)$ & {p:event, parms:{$x_1,\ldots,x_n$}, body:B}$\leftrightarrow R(x_1,\ldots,x_n)$ under the correspondences $x_i=x_i$ (i=1,...,n), the <u>uniform graph</u> of T, U(T) = U(F).

This definition requires some comments. First, the intention is to represent in a single graph the two uniform clauses $C(x_1,\ldots,x_n)$ and $R(x_1,\ldots,x_n)$. Each is first represented by its own uniform graph and then the two graphs are combined. In order to preserve the explicit equality between x_i in the graph U(C) of $C(x_1,\ldots,x_n)$ and its new correspondent y_i in the graph U(R') of $R(x_1,\ldots,x_n)$ all pertinent SP arcs are added. Further, the property ("C",1), meaning "belongs to contingency component," is added to the node property set for each x_i in X_C and ("R",1), meaning "belongs to response component," is added to the node property set for each y_j in X'_R.

<u>Definition [Uniform Representation of a TX]</u>: The <u>uniform representation</u> T* of a TX T with uniform graph U(T) = (X,A,P) is the uniform PSR T* = {p:event, parms:X, body: {{p:SS,1:x_j,1:x_k}: (SS,x_j,x_k) $\in$ A} $\cup$ {{p:SP,1:x_j,1:x_k}: (SP,x_j,x_k) $\in$ A} $\cup$ {{p:C,1:x_j}: C $\in$ P(x_j), $x_j \in$ X}.

<u>Definition [Representationally Equivalent]</u>: If a rule F is representationally equivalent to a TX T, F$\leftrightarrow$T, F is also <u>representationally equivalent</u> to T*, F$\leftrightarrow$T*, and T is representationally equivalent to T*, T$\leftrightarrow$T*.

As a result of these representational equivalences, the following Theorem is a rule-induction counterpart to the

abstraction result (Theorem 1).

Theorem 3: The rule $F^* = [(\forall y_1,\ldots,y_n)\ C^*(y_1,\ldots,y_n) \Rightarrow R^*(y_1,\ldots,y_n)]$ is an inference from the set $F = \{F_i : i=1,\ldots,N\}$ of implications of instantiated clauses where $(\forall i)$ $F_i = [C_i(x_{i,1},\ldots,x_{i,n_i}) \Rightarrow R_i(x_{i,1},\ldots,x_{i,n_i})]$, if and only if the uniform graph U(S*) of the TX S representationally equivalent to F* is a common subgraph of the uniform graphs $U(F_1),\ldots,U(F_N)$ and each node property set in U(S*) contains either ("C",1) or ("R",1).

Proof: (1) By Theorem 1, the set of all abstractions of any set of uniform PSRs corresponds to the set of all subgraphs of their uniform graphs. (2) Moreover, it is clear that if F* is an inference from F_i, the uniform graph U(F*) is a common subgraph of $U(F_i)$. This is true because once parameter correspondences are identified, F* is derivable from F_i by eliminating terms by the conjunction-out rule; equivalently, once node correspondences are established, U(F*) is derived from $U(F_i)$ by eliminating corresponding (unary content) properties (any except ("C",1) and ("R",1)), arcs, and nodes. Thus, there is a 1-1 correspondence between the common subgraphs of the $U(F_i)$ (restricted to those possessing ("C",1) or ("R",1) properties for each node) and the possible inferences to be produced from the F_i (i=1,...,N). (3) As a result of (1) and (2), the claim is proved. [QED]

All that remains is to establish the effectiveness of the two-step rule learning procedure for the class of prob-

lems for which it is intended. This requires the following definitions.

Definition [Training Information Sufficient for Rule Induction]: $I = \{(S_i,R_i): i=1,\ldots,N\}$, where $S_i=P_i(x_{i,1},\ldots,x_{i,n_i})$ and $R_i=Q_i(y_{i,1},\ldots,y_{i,m_i})$ are uniform clauses (or representationally equivalent uniform PSRs), is training information sufficient for induction of the rule $F = [(\forall y_1,\ldots,y_n)\ C(y_1,\ldots,y_n) \Rightarrow R(y_1,\ldots y_n)]$ if $(\forall i)\ (\exists r_i)\ r_i \subset \{x_{i,1},\ldots,x_{i,n_i}\} \times \{y_{i,1},\ldots,y_{i,m_i}\}$ and $(x,y) \in r_i$ & $(x',y) \in r_i \Rightarrow x'=x$ (i.e. r_i is a set of bindings between instances (parameters) of S_i and R_i, where each x in S_i may correspond to many y in R_i, but each y in R_i may have at most one correspondent in S_i) such that F is an inference from $F_1,\ldots,F_N$ where $(\forall i)\ F_i = [(\forall(x,y) \in r_i)\ x=y \ \&\ S_i \Rightarrow R_i]$.

The interpretation of sufficient training information is best provided by identifying what it means to be insufficient.

Lemma. If $I = \{(S_i,R_i): i=1,\ldots,N\}$ is not sufficient for the induction of F, I must contain at least one pair (S_i,R_i) such that R_i cannot be proved from S_i & F regardless of which instances in S_i are supposed equivalent (on a 1-many basis) to some in R_i.

Definition [The Two-Step Rule Learning Procedure]: Let $I = \{(S_i,R_i): i=1,\ldots,N\}$ be training information for a rule induction problem. The two-step rule learning procedure is defined as follows. Step 1 (Prediction Matching): For each $i=1,\ldots,N$, use the IM' procedure to compute B_i =

$A([S_i,R_i]) = S_i * R_i$ (S_i,R_i uniform clauses or PSRs), the set of all abstractions obtainable under any 1-many binding relation between their parameters, X_i and Y_i respectively. Note, the IM' procedure is simply the IM procedure modified to allow 1-many parameter correspondences between X_i and Y_i but not vice versa. For each $B \in B_i$ defining an abstraction of $\{S_i,R_i\}$, let the associated binding rule (a 1-many relation on $X_i \times Y_i$) be denoted $f(B)$. Let S_i and R_i be re-expressed as $S_i(B)$ and $R_i(B)$, the representations resulting from replacing x and y by a new, distinct symbol, $a(x)$ for all $(x,y) \in f(B)$ such that $y \neq a(x) \neq x' \,\&\, a(x) = a(x') \Leftrightarrow x=x'$ (for all $x \in X_i$, $x' \in X_i$, $y \in Y_i$). Let $Q_i(I) = \{[S_i(B) \Rightarrow R_i(B)] : B \in B_i\} \cup \{[S_i \Rightarrow R_i]\}$ be the complete data set for (S_i,R_i) (i.e. essentially all possible interpretations of implication between S_i and R_i). Step 2 (Generalization Matching): Let $Q(I) = \{F : F = \{F_1,\ldots, F_N\} \,\&\, (\forall i)\ F_i \in Q_i(I)\}$ be the complete training information set derived from I. For each $I' = \{[S_1' \Rightarrow R_1'],\ldots, [S_N' \Rightarrow R_N']\} \in Q(I)$, construct the uniform graph $U(T_i')$ for each implication $T_i' = [S_i' \Rightarrow R_i']$ $(i=1,\ldots,N)$ and denote the corresponding uniform PSR W_i'. Using the IM procedure, compute the set $A(W_I')$ of all abstractions of $W_I' = \{W_1',\ldots, W_N'\}$ by computing $G(U_I')$, the set of all common subgraphs of $U_I' = \{U(T_1'),\ldots,U(T_N')\}$. Denote the complete set of resultant abstractions A(I) and the complete set of resultant uniform graphs G(I), where $A(I) = \bigcup_{I' \in Q(I)} A(W_I')$ and $G(I) = \bigcup_{I' \in Q(I)} G(U_I')$.

The effectiveness of the two-step rule learning procedure is established by the following corollary to Theorem 3.

Corollary: Using the notations of the preceding definitions, I is training information sufficient for the induction of the rule $F \Leftrightarrow (\exists U_T)\ U_T \in G(I)$, where U_T is a uniform graph (corresponding to a TX T). T is the uniform PSR whose graph is U_T (i.e. T = {p:event, parms:X, body:{C,S: C is a unary content relation corresponding to a property of any $x \in X$, where $U_T=(X,A,P)$, S is any (SS or SP) relation corresponding to any SS or SP type arc in A}}) and $F \leftrightarrow T$.

Proof: I is sufficient for the induction of $F \Leftrightarrow (\forall i = 1,\ldots,N)\ (\exists f_i)$ f_i is a 1-many binding rule from the instances (parameters) of S_i to those of R_i such that, if S_{f_i} and R_{f_i} denote the uniform PSRs obtained from S_i and R_i respectively by replacing each parameter x and y ($(\forall x,y)\ (x,y) \in f_i$) by a unique symbol a(x) such that F is an inference from $\{[S_{f_1} \Rightarrow R_{f_1}],\ldots,[S_{f_N} \Rightarrow R_{f_N}]\} \Leftrightarrow (\exists q)\ q = \{q_1,\ldots,q_N\} \in Q(I)$ (the complete training information set derived from I) and F is an inference from $q \Leftrightarrow (\exists T, U_T)$ where $U_T = U(T)$ & $F \leftrightarrow T$ & U_T is a common subgraph of the implication uniform graphs $U(q_1),\ldots,U(q_N) \Leftrightarrow$ T corresponds to one abstraction (is equivalent to the PSR represented by one model) produced by the IM procedure applied (successively) to each set $q \in Q(I)$. The claim is therefore proved. [QED]

As a practical alternative to the expansion of I into Q(I) and the successive computation of all abstractions of each $q \in Q(I)$, it is easy to see that the following three-

step rule-learning procedure computes exactly the same set of inferences.

Definition [The Three-Step Rule Learning Procedure]: Step 1 (Generalized Prediction Matching): Given I (as above), construct a uniform graph $U_i = (X_i, A_i, P_i)$ for each implication $[S_i \Rightarrow R_i]$ (i=1,...,N). Then $U_i' = (X_i, A_i', P_i)$ is constructed so that $A' = A \cup \{(SP,x,y)$: x is a parameter in S_i and y is a parameter in $R_i\}$. That is, all possible SP arcs between a contingency parameter X and a response parameter y are added. Step 2 (Generalization Matching): Using the IM procedure, compute the set $G'(I) = \{g : g$ is a common subgraph of $U_1', \ldots, U_N'\}$. Step 3 (Consistency Check and Read-out): Compute $G(I) = \{g : g = (X,A,P) \in G'(I)$ and $(\forall x,y,x')\ (SP,x,y) \in A \ \&\ (SP,x',y) \in A \ \&\ ("C",1) \in P(x) \ \&\ ("C",1) \in P(x') \ \&\ ("R",1) \in P(y) \Rightarrow x=x'\}$. That is, G(I) is the subset of G'(I) constrained by the requirement that each contingency parameter map onto at most one response parameter.

The advantage of the three-step procedure is that it reduces the number of times the IM procedure is invoked (to one) and, therefore, eliminates the necessity to compute sets of abstractions for each distinct set in the complete training information set derived from I.

Application [Transformational Grammar Rule Learning]: It follows from the preceding corollary that if I is a set of sentence pairs such that each R_i is derived from S_i by use of the rule F (i.e. $(\forall i)\ F \& S_i \Rightarrow R_i$), $F \leftrightarrow T$ where T is a

transformation whose uniform graph U(T) is one of the subgraphs produced by the two-step (or three-step) rule learning procedure. Moreover, the assumption seems plausible that the greater the complexity of F (the number of nodes and arcs in U(T)), the fewer the number (N) of training examples needed to support any level of confidence that U(T) is the actual rule underlying the generation of I. Equivalently, one would be well-advised to assign greater plausibility to an inference F than to an inference F' whenever the complexity of F exceeds that of F'. On the other hand, the plausibility of the assumption is empirically testable whenever there is, in addition to I, sets $I'(F) = \{(S_{F,i}, R_{F,i}): i=1,\ldots,n_F\}$ and $I'(F') = \{(S_{F',i}, R_{F',i}): i=1,\ldots, n_{F'}\}$ of counterexamples to the rules F and F', respectively; i.e. $(S_{F,i}, R_{F,i}) \in I'(F) \Rightarrow [(F \,\&\, S_{F,i} \Rightarrow Q) \,\&\, R_{F,i} \Rightarrow \sim Q]$ and, similarly for F' (m.m.). Such counterexamples are provided whenever certain response alternatives (Q and $R_{F,i}$) are mutually exclusive possibilities. For example, in pattern classification behavior, at most one response class may be selected. Similarly, if one supposes that people use transformational grammar and there is exactly one deep structure in memory at a time, any inferred rule $F = [(\forall x)\, C(x) \Rightarrow R(x)]$ could have as its counterexamples all pairs $(S_{F,i}, R_{F,i})$ of successive deep structures which satisfy the condition that Q is not a substructure (an abstraction) of $R_{F,i}$ where Q is the sentence structure derived from $S_{F,i}$ by using the rule F. It is proposed, therefore, that the performance

value of any inference be computed in the same way as for the categorical rules of Chapter 4, with the appropriate substitution of I for the set of all positive exemplars of the rule F and I'(F) for the set of all negative exemplars of F.

BIBLIOGRAPHY

Anderson, J. R., & Bower, G. H. Human associative memory. Washington, D.C.: Winston, 1973.

Arbib, M. A. Brains, machines, and mathematics. New York: McGraw-Hill, 1964.

Arbib, M. A. Memory limitations of stimulus-response models. Psychological Review, 1969, 76, 507-510.

Arbib, M. A. The metaphorical brain. An introduction to cybernetics as artificial intelligence and brain theory. New York: Wiley, 1972.

Banerji, R. Winston's theory of learning. ACM SIGART Newsletter, 1974, 45, 5.

Balzer, R. M. A global view of automatic programming. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Barrow, H. G., Ambler, A. P., & Burstall, R.M. Some techniques for recognising structures in pictures. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Bartlett, F. C. Remembering, a study in experimental and social psychology. London: Cambridge University Press, 1932.

Becker, J. D. A model for the encoding of experiential information. In R. C. Schank & K. M. Colby (Eds.), Computer models of thought and language. San Francisco: Freeman, 1973.

Bekesy, G. von. Sensory inhibition. Princeton: Princeton University Press, 1967.

Biermann, A. W., & Feldman, J. A. A survey of results in grammatical inference. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Block, H. D. A review of "Perceptrons." Information and Control, 1970, 17, 501-522.

Boyer, R. S., & Moore, J. S. Proving theorems about LISP functions. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Bruner, J. S., Goodnow, J. J., & Austin, G. A. A study of thinking. New York: Wiley, 1956.

Cavanagh, J. P. Holographic processes realizable in the neural realm: prediction of short term memory performance. Unpublished doctoral dissertation. Pittsburgh: Carnegie-Mellon University, 1972.

Chomsky, N. A review of Verbal Behavior by B. F. Skinner. Language, 1959, 35, 26-58.

Chomsky, N. Aspects of the theory of syntax. Cambridge: MIT Press, 1965.

Chomsky, N. Syntactic structures. The Hague: Mouton, 1967.

Chomsky, N. Language and mind. New York: Harcourt Brace Jovanovich, 1968.

Church, A. The calculus of lambda-conversion. Princeton:

Princeton University Press, 1941.

Clowes, M. B. Transformational grammars and the organization of pictures. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969.

Clowes, M. B. On seeing things. Artificial Intelligence, 1971, 2, 79-116.

Collins, A. M., & Quillian, M. R. How to make a language user. In E. Tulving & W. Donaldson (Eds.), Organization of memory. New York: Academic Press, 1972.

Corneil, D. G., & Gottlieb, C. C. An efficient algorithm for graph isomorphism. Journal of the Association for Computing Machinery, 1970, 17, 51-64.

Cornsweet, T. N. Visual perception. New York: Academic Press, 1970.

Darlington, J. L. Automatic program synthesis in second-order logic. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Davis, M. Computability and unsolvability. New York: McGraw-Hill, 1958.

Deutsch, S. Models of the nervous system. New York: Wiley, 1967.

Dodwell, P. C. Visual pattern recognition. New York: Holt, 1970.

Dreyfus, H. L. What computers can't do. A critique of artificial reason. New York: Harper and Row, 1972.

Eden, M., & Halle, M. Characterization of cursive writing. In C. Cherry (Ed.), Proceedings of the Fourth London Symposium on Information Theory. New York: Academic Press, 1961.

Eden, M., & Halle, M. Handwriting and pattern recognition. In E. E. David (Ed.), Special issue on sensory information processing, IRE Transactions on Information Theory, 1962, IT-8, 74-191.

Ernst, G. W., & Newell, A. GPS: A case study in generality and problem solving. New York: Academic Press, 1969.

Evans, T. G. A grammar-controlled pattern analyzer. In A. J. H. Morell (Ed.), Information processing 68, Proceedings of IFIP Congress 68. Amsterdam: North Holland, 1969a.

Evans, T. G. Descriptive pattern-analysis techniques: potentialities and problems. In S. Watanabe (Ed.), Methodologies of pattern recognition. New York: Academic Press, 1969b.

Evans, T. G. Descriptive pattern analysis techniques. In A. Grasselli (Ed.), Automatic interpretation and classification of images. New York: Academic Press, 1969c.

Feigenbaum, E. A. The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and thought. New York: McGraw-Hill, 1963.

Ferguson, T. S. Mathematical statistics, A decision theore-

tic approach. New York: Academic Press, 1967.

Fikes, R. E., & Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 1971, 2, 189-208.

Finley, M. R., Jr. An experimental study of the formation and development of Hebbian cell assemblies by means of a neural network simulation. Unpublished doctoral dissertation. Ann Arbor: The University of Michigan, 1967.

Fischer, G. L., Jr., Pollock, D. K., Raddack, B., & Stevens, M. E. (Eds.), Optical character recognition. Washington, D. C.: Spartan, 1962.

Fogel, L. J., Owens, A. J., & Walsh, M. J. Artificial intelligence through simulated evolution. New York: Wiley, 1966.

Franks, J. J., & Bransford, J. D. Abstraction of visual patterns. Journal of Experimental Psychology, 1971, 90, 65-74.

Friedberg, R. M. A learning machine, I. IBM Journal of Research and Development, 1958, 2, 2-13.

Friedberg, R. M., Dunham, B., & North, J. H. A learning machine, part II. IBM Journal of Research and Development, 1959, 3, 282-287.

Friedman, J. A computer model of transformational grammar. New York: American Elsevier, 1971.

Friedman, L. Instinctive behavior and its computer synthesis. Behavioral Science, 1967, 12, 85-108.

Fu, K. S. Sequential methods in pattern recognition and machine learning. New York: Academic Press, 1968.

Fu, K. S. (Ed.). Pattern recognition and machine learning. New York: Plenum Press, 1971.

Fu, K. S. On syntactic pattern recognition and stochastic languages. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Galton, F. Inquiries into human faculty and its development. London: Dent, 1907.

Gardner, R. A., & Gardner, B. T. Teaching sign language to a chimpanzee. Science, 1969, 165, 664-672.

Gibson, E. Perceptual learning. Annual Review of Psychology, 1963, 14, 29-56.

Gibson, E. Principles of perceptual learning and development. New York: Appleton, 1969.

Good, I. J. Speculations concerning the first ultra-intelligent machine. In Advances in computers, vol. 6. New York: Academic Press, 1965.

Green, D. M., & Swets, J. A. Signal detection theory and psychophysics. New York: Wiley, 1966.

Halle, M., & Stevens, K. Speech recognition: a model and program for research. IRE Transactions of Information Theory, 1962, IT-8, 328-333.

Harman, H. H. Modern factor analysis. Second edition. Chicago: University of Chicago Press, 1967.

Hayes-Roth, B. Interactions in the acquisition and utilization of structured knowledge. Unpublished doctoral

dissertation. Ann Arbor: The University of Michigan, 1974.

Hayes-Roth, F. The meaning and mechanics of knowledge. Working Paper 569-71. Cambridge: Sloan School of Management, M. I. T., 1971.

Hayes-Roth, F. Design for the construction of intelligence. MMPP 73-2. Ann Arbor: Michigan Mathematical Psychology Program, 1973a.

Hayes-Roth, F. A structural approach to pattern learning and the acquisition of classificatory power. Proceedings of the First International Joint Conference on Pattern Recognition. New York: I.E.E.E., 1973b.

Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. Proceedings of the Second International Joint Conference on Pattern Recognition. 1974a(in press).

Hayes-Roth, F. Schematic classification problems and their solution. Pattern Recognition, 1974b, (in press).

Hayes-Roth, F., & Hayes-Roth, B. A schematic model of abstraction. MMPP 74-2. Ann Arbor: Michigan Mathematical Psychology Program, 1974.

Hebb, D. O. The organization of behavior. New York: Wiley, 1949.

Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating objects in a robot. AI-TR-258. Cambridge: MIT Artificial Intelligence Laboratory, 1972.

Holland, J. H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1974 (in press).

Hopcroft, J. E., & Ullman, J. D. Formal languages and their relation to automata. Reading, Massachusetts: Addison-Wesley, 1969.

Horowitz, L. M., & Prytulak, L. S. Redintegrative memory. Psychological Review, 1969, 76, 519-531.

Hubel, D. H., & Wiesel, T. H. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. Journal of Neurophysiology, 1965, 28, 229-289.

Humby, E. Programs from decision tables. New York: American Elsevier, 1973.

Hunt, E. B. Concept learning: an information processing problem. New York: Wiley, 1962.

Hunt, E. B., Marin, J., & Stone, P. Experiments in induction. New York: Academic Press, 1966.

John, E. R. Mechanisms of memory. New York: Academic Press, 1967.

Joshi, A. K. Remarks on some aspects of language structure and their relevance to pattern analysis. Pattern Recognition, 1973, 5, 365-382.

Katz, S. M., & Zohar, M. A heuristic approach to program verification. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Kleene, S. C. General recursive functions of natural numbers. Mathematische Annalen, 1936, 112, 727-742.

Kleene, S. C. Introduction to metamathematics. Princeton: Van Nostrand, 1952.

Kochen, M. An experimental program for the selection of disjunctive hypotheses. Proceedings of the Western Joint Computer Conference, 1961a, 19, 571-578.

Kochen, M. Experimental study of "hypothesis formation" by computer. In C. Cherry (Ed.), Fourth London Symposium on Information Theory. London: Butterworth, 1961b.

Kolers, P. A., & Eden, M. (Eds.). Recognizing patterns: studies in living and automatic systems. Cambridge: MIT Press, 1968.

Kotovsky, K., & Simon, H. A. Empirical tests of a theory of human acquisition of concepts for sequential patterns. Cognitive Psychology, 1973, 4, 399-424.

LaBerge, D., & Samuels, S. J. Toward a theory of automatic information processing in reading. Cognitive Psychology, 1974, 6, 293-323.

Langendoen, D. T. The study of syntax. The generative transformational approach to the structure of American English. New York: Holt, Rinehart, Winston, 1969.

Lenneberg, E. H. Biological foundations of language. New York: Wiley, 1967.

Lindsay, P. H., & Norman, D. A. Human information processing. An introduction to psychology. New York: Academic Press, 1972.

Lindsay, R. K. Artificial evolution of intelligence. Contemporary Psychology, 1968, 13, 113-116.

Lindsay, R. K. Complexities of the mind at work. Review of "What computers can't do. A critique of artificial reason" by H. L. Dreyfus. Science, 1972, 176, 630-631.

Luce, R. D. Individual choice behavior. A theoretical analysis. New York: Wiley, 1959.

MacKay, D. M. The epistemological problem for automata. In C. E. Shannon & J. McCarthy (Eds.), Automata studies. Princeton: Princeton University Press, 1956.

McCarthy, J. Recursive functions of symbolic expressions. Communications of the ACM, 1960, 3, 184-195.

McCarthy, J., Abrahams, P. W., Edwards, D. J., Hart, T. P., & Levin, M. I. LISP 1.5 programmer's manual. Cambridge: MIT Press, 1962.

McCulloch, W. S. (Ed.). Embodiments of mind. Cambridge: MIT Press, 1965.

McCulloch, W. S., & Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 1943, 5, 115-133.

Meisel, W. S. Computer oriented approaches to pattern recognition. New York: Academic Press, 1970.

Melton, A. W., & Martin, E. (Eds.). Coding processes in human memory. Washington, D. C. : Winston, 1972.

Michalski, R. S. AQVAL/1--Computer implementation of a variable-valued logic system VL_1 and examples of its

application to pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition. New York: I.E.E.E., 1973.

Miller, G. A., & Chomsky, N. Finitary models of language users. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), Handbook of mathematical psychology, vol. 2. New York: Wiley, 1963.

Miller, G. A., Galanter, E., & Pribram, K. H. Plans and the structure of behavior. New York: Holt, 1960.

Milner, P. M. The cell assembly: mark II. Psychological Review, 1957, 64, 242-252.

Minsky, M. L. Steps towards artificial intelligence. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and thought. New York: McGraw-Hill, 1963.

Minsky, M. L. Computation: finite and infinite state machines. Englewood Cliffs: Prentice-Hall, 1967.

Minsky, M. L., & Papert, S. Perceptrons. An introduction to computational geometry. Cambridge: MIT Press, 1969.

Moore, J., & Newell, A. How can Merlin understand? Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1973.

Narasimhan, R. Labelling schemata and syntactic descriptions of pictures. Information and Control, 1964, 7, 151-179.

Narasimhan, R. On the description, generation, and recognition of classes of pictures. In A. Grasselli (Ed.),

Automatic interpretation and classification of images. New York: Academic Press, 1969.

Neisser, U. Cognitive psychology. New York: Appleton, 1967.

Newell, A. Documentation of IPL-V. Communications of the ACM, 1963, 6, 86-89.

Neisser, U. Cognitive psychology. New York: Appleton, 1967.

Newell, A. A theoretical exploration of mechanisms for coding the stimulus. In A. W. Melton & E. Martin (Eds.), Coding processes in human memory. Washington, D. C.: Winston, 1972.

Newell, A. Artificial intelligence and the concept of mind. In R. C. Schank & K. M. Colby (Eds.), Computer models of thought and language. San Francisco: Freeman, 1973.

Newell, A., & Simon, H. A. Human problem solving. Englewood Cliffs: Prentice-Hall, 1972.

Nilsson, N. Learning machines. New York: McGraw-Hill, 1965.

Nilsson, N. J. Problem solving methods in artificial intelligence. New York: McGraw-Hill, 1971.

Norman, D. A. Memory and attention: an introduction to human information processing. New York: Wiley, 1969.

Norman, D. A. (Ed.). Models of human memory. New York: Academic Press, 1970.

Patrick, E. A. Fundamentals of pattern recognition. Englewood Cliffs: Prentice-Hall, 1972.

Paxton, W. H., & Robinson, A. E. A parser for a speech understanding system. Proceedings of the Third International Joint Conference on Artificial Intelligence.

Menlo Park: Stanford Research Institute, 1973.

Pflatz, J. L., & Rosenfeld, A. Web grammars. Proceedings of the First International Joint Conference on Artificial Intelligence, 1969, 609-620.

Pitts, W., & McCulloch, W. S. How we know universals: the perception of auditory and visual forms. Bulletin of Mathematical Biophysics, 1947, 9, 127-147.

Plum, T. W. Simulation of a cell-assembly model. Unpublished doctoral dissertation. Ann Arbor: The University of Michigan, 1972.

Premack, D. Language in chimpanzee? Science, 1971, 172, 808-822.

Prewitt, J. M. S. Object enhancement and extraction. In B. S. Lipkin & A. Rosenfeld (Eds.), Picture processing and psychopictorics. New York: Academic Press, 1970.

Quillian, M. R. Semantic memory. In M. L. Minsky (Ed.), Semantic information processing. Cambridge: MIT Press, 1968.

Quillian, M. R. The teachable language comprehender: a simulation program and theory of language. Communications of the ACM, 1969, 12, 459-476.

Rabinow, J. The present state of the art of reading machines. In L. Kanal (Ed.), Pattern recognition. Washington, D.C.: Thompson, 1968.

Rastall, J. Graph-family matching. MIP-R-62. Edinburgh: Department of Machine Intelligence and Perception, University of Edinburgh, 1969.

Ratliff, F. Mach bands: quantitative studies on neural networks in the retina. San Francisco: Holden-Day, 1965.

Reboh, R., & Sacerdoti, E. A preliminary QLISP manual. AI Center Technical Note 81. Menlo Park: Stanford Research Institute, 1973.

Reddy, D. R., Erman, L. D., Fennell, R.D., & Neely, R. B. The HEARSAY speech understanding system: an example of the recognition process. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Reed, S. K. Psychological processes in pattern recognition. New York: Academic Press, 1973.

Reitman, J. S., & Bower, G. H. Storage and later recognition of exemplars of concepts. Cognitive Psychology, 1973, 4, 194-206.

Rochester, N., Holland, J. H., Haibt, L. H., & Duda, W. L. Test of a cell assembly theory of the action of the brain, using a large digital computer. IRE Transactions on Information Theory, 1956, IT-2, 80-93.

Rosenblatt, F. The perceptron, a probabilistic model for information organization and storage in the brain. Psychological Review, 1958, 65, 368-408.

Rulifson, J. F., Derksen, J. A., & Waldinger, R. J. QA4: a procedural calculus for intuitive reasoning. AI Center Technical Note 73. Menlo Park: Stanford Research Institute, 1972.

Rumelhart, D. E., Lindsay, P. H., & Norman, D. A. A process model for long term memory. In E. Tulving & W. Donaldson (Eds.), Organization of memory. New York: Academic Press, 1972.

Samuel, A. L. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 1959, 3, 210-229.

Sauvain, R. W., & Uhr, L. A teachable pattern describing and recognizing program. Pattern Recognition, 1969, 1, 219-232.

Schank, R. C. Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, 1972, 3, 552-631.

Selfridge, O. Pandemonium: a paradigm for learning. In Symposium on the mechanisation of thought processes. London: HM Stationery Office, 1959.

Selfridge, O., & Neisser, U. Pattern recognition by machine. Scientific American, 1960, 203, 60-68.

Selfridge, O. G., & Neisser, U. Pattern recognition by machine. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and thought. New York: McGraw-Hill, 1963.

Shaw, A. C. Parsing of graph-representable pictures. Journal of the Association for Computing Machinery, 1970, 17, 453-481.

Shaw, A. C. Picture graphs, grammars, and parsing. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.

Siklossy, L. A language-learning heuristic program. Cognitive Psychology, 1971, 2, 479-495.

Skinner, B. F. Verbal behavior. New York: Appleton-Century-Crofts, 1957.

Simon, H. A., & Kotovsky, K. Human acquisition of concepts for sequential patterns. Psychological Review, 1963, 70, 534-546.

Solmonoff, R. J. A new method for discovering the grammars of phrase structure languages. Proceedings of the International Conference on Information Processing (ICIP). Paris: UNESCO House, 1959.

Smith, J. E. K. Decision theoretic speaker recognizer. Journal of the Acoustical Society of America, 1962, 34, 1988.

Suppes, P. Stimulus-response theory of finite automata. Journal of Mathematical Psychology, 1969a, 6, 327-355.

Suppes, P. Stimulus-response theory of finite automata and tote hierarchies. A reply to Arbib. Psychological Review, 1969b, 76, 511-514.

Sussenguth, E. H. A graph-theoretic algorithm for matching chemical structures. Journal of Chemical Documentation, 1965, 5, 36-43.

Sussman, G. J. A computational model of skill acquisition. AI TR-297. Cambridge: MIT Artificial Intelligence Laboratory, 1973.

Swets, J. A. (Ed.). Signal detection and recognition by

human observers. New York: Wiley, 1964.

Thomason, M. G. Finite fuzzy automata, regular fuzzy automata, and pattern recognition. Pattern Recognition, 1973, 5, 383-390.

Thurstone, L. L. Multiple factor analysis. Chicago: University of Chicago Press, 1947.

Torgerson, W. S. Theory and methods of scaling. New York: Wiley, 1958.

Tulving, E., & Donaldson, W. (Eds.). Organization of memory. New York: Academic Press, 1972.

Turing, A. M. On computable numbers with an application to the Entscheidungs-problem. Proceedings of the London Mathematical Society, 1936, series 2, 42, 230-265. See also correction in 1936-37, 43, 544-546.

Uhr, L. Pattern recognition, learning, and thought. Englewood Cliffs: Prentice-Hall, 1973.

Uhr, L. Flexible linguistic pattern recognition. Pattern Recognition, 1971, 3, 363-370.

Uhr, L., & Vossler, C. Recognition of speech by a computer program that was written to simulate a model for human visual pattern perception. Journal of the Acoustical Society of America, 1961, 33, 14-26.

Uhr, L., & Vossler, C. A pattern-recognition program that generates, evaluates, and adjusts its own operators. In E. A. Feigenbaum & J. Feldman (Eds.), Computers and thought. New York: McGraw-Hill, 1963.

Uhr, L., Vossler, C., & Uleman, J. Pattern recognition

over distortions by human subjects and a computer model of human form perception. Journal of Experimental Psychology, 1962, 63, 227-234.

Underwood, B., & Zimmerman, J. The syllable as a source of error in multisyllable word recognition. Journal of Verbal Learning and Verbal Behavior, 1973, 12, 701-706.

Unger, S. H. GIT--a heuristic program for matching chemical structures. Communications of the ACM, 1964, 7, 26-34.

Uttal, W. R. The effect of deviations from linearity on the detection of dotted line patterns. Vision Research, 1973, 13, 2155-2163.

Walker, D. E. Speech understanding through syntactic and semantic analysis. Proceedings of the Third International Joint Conference on Artificial Intelligence. Menlo Park: Stanford Research Institute, 1973.

Watanabe, S. Subspace method in pattern recognition. Proceedings of the First International Joint Conference on Pattern Recognition. New York: I.E.E.E., 1973.

Weissman, C. LISP 1.5 primer. Belmont, California: Dickenson, 1967.

Winograd, T. A program for understanding natural language. Cognitive Psychology, 1972, 3, 1-191.

Winston, P. H. Learning structural descriptions from examples. AI-TR-76. Cambridge: MIT Artificial Intelli-

gence Laboratory, 1970.

Woods, W. A. Transition network diagrams for natural language analysis. Communications of the ACM, 1970, 13, 591-606.

Woods, W. A., & Kaplan, R. M. The lunar sciences natural language information system. Cambridge: Bolt, Beranek and Newman, 1971.

Wooldridge, D. E. The machinery of the brain. New York: McGraw-Hill, 1963.

Zadeh, L. A. Fuzzy sets. Information and Control, 1965, 8, 338-343.