COGNITIVE DIMENSIONS OF HUMAN PROBLEM SOLVING, INVENTION, AND CREATIVITY FROM CONVENTIONAL, CONNECTIONIST, AND INTEGRATED PERSPECTIVES

By

TIM P. McCOLLUM

Bachelor of Science
in Arts and Sciences
Oklahoma State University
1982

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1983

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1992

COGNITIVE DIMENSIONS OF HUMAN PROBLEM SOLVING, INVENTION, AND CREATIVITY FROM CONVENTIONAL, CONNECTIONIST, AND INTEGRATED PERSPECTIVES

Thesis Approved:

Robert J Weber
Thesis Advisor

Larry Hochbaum

Robert L Stanners

C. M. Baum

Thomas C. Collins
Dean of the Graduate College

PREFACE

The present work represents an attempt to understand the complex cognitive processes which underlie the general class of human behaviors known as creative problem solving. More specifically, I will use inventiveness to represent the broader class of creative human behaviors. Therefore, my major aim will be to examine human inventiveness, and any implications for the more general class of creativity will be limited by the degree with which human inventiveness accurately represents the breadth of human creative problem solving.

Two separate avenues of scientific inquiry were pursued during the course of the project. The first approach was an attempt to study human inventiveness in the laboratory. This investigative strategy attempted to extend classic research findings from traditional problem solving domains (e.g chess, game-playing) to the less constrained creative problem solving domain of invention. The technique used was a modification of the approach used by Chase and Simon (1973) to study Chess expertise. The first investigative strategy was also highly influenced by the techniques traditionally used in expert system development, knowledge engineering, and rule-oriented information processing analyses.

The second investigative tactic was to examine creative human problem solving at a more theoretical level in order to achieve three primary aims. First, I wished to provide an overview of what is now often referred to as the traditional symbolic approach to studying human problem solving which was advanced by Newell and Simon (1972). Next, I wished to examine a more recently developed approach to understanding human cognition known as neural or connectionist modeling. In doing so, I hoped to develop insights into human creativity and inventiveness that might be provided by this newer perspective on cognition. Lastly, I attempted to identify, describe, and integrate the advantages of both the traditional symbolic and connectionist perspectives, so that a more unified, and complete, view of human inventiveness and creativity might emerge.

I wish to express sincere appreciation to the faculty and students of the O. S. U. Psychology department for the support and inspiration I have received throughout my graduate program. Special thanks goes to Dr. Larry Hochhaus, Dr. Robert Stanners, Dr. Donald Fromme, and Dr. James Price for their advice, tutelage, and friendship which made the rigors of graduate study worthwhile and enjoyable. Likewise, I wish to express gratitude to Dr. Robert Weber for his mentorship, friendship, and encouraging me to follow my interests and instincts during the dissertation project. I also wish to thank the dissertation committee (Dr. Robert

Weber, Dr. Robert Stanners, Dr. Larry Hochhaus, and Dr. Charles Bacon) for their assistance, comments and support throughout the project. Many thanks go to my wife Katrena, my parents, my family, and my friends for bolstering my spirits during the seemingly endless effort to complete this project. Lastly, I would like to thank Jacob McCollum for continually putting my priorities in perspective.

TABLE OF CONTENTS

	Page
COGNITIVE DIMENSIONS OF HUMAN PROBLEM SOLVING INVENTION, AND CREATIVITY FROM CONVENTIONAL CONNECTIONIST AND INTEGRATED PERSPECTIVES.	1
Abstract.	2
Introduction.	3
Method.	15
Subjects	15
Materials and Apparatus.	16
Design and Procedure.	19
Results	22
Scoring.	22
Analysis	24
Discussion of Empirical Findings.	27
Theoretical Discussion.	36
Two Perspectives of Cognition.	36
The Conventional Information Processing Perspective	41
Problem Solving Information Processing Systems	50
The Problem Solving Process.	61
A Summary of Newell and Simons' Human Problem Solving Theory	71
The Neural Network Perspective	76
The Development of a Neural Network.	85
Two Representative Neural Networks	88
Phase I: Engineering.	90
Multi-layered Perceptrons	90
The Hopfield Net.	93
Phase II: Training.	96
Multi-Layered Perceptron.	96
The Hopfield Net.	98
Phases III and IV: Testing and Operation. . .	99
Multi-layered Perceptron.	99
The Hopfield Net.	100
Conclusion	102
Problem Solving Neural Nets.	105
Synthesis: Integrating Neural Network and Traditional Problem Solving Perspectives . .	117
A Transitional Model of a Hybrid Problem Solving System that Integrates Neural and Symbolic Processes	122

Sequential Neural Processing,
Consciousness, Mental Models,
and Creativity 134
Summary and Conclusions. 170

REFERENCES 183

LIST OF FIGURES

Figure		Page
1.	Silhouette of the Mechanical Object	17
2.	Silhouette of the Nonsense Object	18
3.	Laboratory Arrangement.	18
4.	Average Number of Correct Connections During Non-Zero Trials.	25
5.	Average Number of Correct Connections During Both Zero and Non-Zero Trials	25
6.	Average Elapsed Construction Times.	26
7.	Average Number of Disconnects (Errors).	27
8.	Average Number of Zero Trials	27
9.	Average Number of Looks	27
10.	Schematic of an Information Processing System	42
11.	Thermostat Program.	50
12.	Tic-Tac-Toe Game State.	58
13.	Hypothetical Problem Spaces for Tic-Tac-Toe	59

Cognitive Dimensions of Human Problem Solving, Invention, and Creativity from Conventional, Connectionist, and Integrated Perspectives

Tim P. McCollum

Oklahoma State University

Running Head: PERSPECTIVES ON INVENTION AND CREATIVITY

Abstract

Experimentation and a integrated perspective of current theories were used to investigate human inventiveness. During the experiment, two expert inventors and two novices replicated a mechanical object and a nonsense object. In accordance with Chase and Simon (1973), experts were expected to outperform novices in the mechanical condition and perform equivalently to novices in the nonsense condition. No statistical difference, however, was found between experts and novices. Interpretation is complicated by a possible ceiling effect and the confounding of expertise with subject age due to difficulties obtaining expert inventors as subjects. An extended discussion section examines conventional models of problem solving, the fundamentals of parallel distributed processing models, and how the two might be integrated to produce a new perspective on inventiveness. Edison's invention of the electric light is examined from an integrated perspective to illustrate the insights that might be gained from unifying conventional and neural processing models.

Cognitive Dimensions of Human Problem Solving, Invention, and Creativity from Conventional, Connectionist, and Integrated Perspectives

Areas of expertise which are generally described as creative (e.g. musical composition, creative writing, invention, sculpture, painting) are intrinsically interesting and important because creative activity best exemplifies the human cognitive abilities which distinguish us from even our nearest cousins in the animal kingdom. Indeed, humanity's propensity for inventive and creative actions is largely responsible for our success as a species. Studying creativity may therefore provide insight into the human mind's unique capacities and facilitate the development of techniques which can improve human problem solving performance. Unfortunately, due to their unconstrained, intense, and often lengthy nature, creative behaviors are some of the most difficult to systematically study.

Newell, Shaw, and Simon (1962) described four characteristics which distinguish creative problem solving from other forms of problem solving behavior. According to their view, creative problem solving is problem solving which produces a novel end-product by

using unconventional strategies, persistent, highly intense thought, and a loosely constrained problem formulation. In this view, the product of a creative act must be unique, or novel, and have value to the solver or others. The thinking must also be unconventional in that it requires modification or rejection of traditional approaches to similar problems. Creative problem solving further requires persistence and high motivation over a long period of time, and periods of intense concentration are necessary to arrive at a successful end product. This may be due to the fact that traditional approaches are wrong for the solution of a problem and the solver must work through those incorrect solutions in order to "create" a correct solution. Lastly, a creative problem must in large part be defined by the solver. That is, the problem is poorly defined and the constraints upon the problem are loose, at best. Thus, a large part of creative problem solving is discovering and formulating the specific problem, and sub-problems, to be solved.

The traditional approach to studying problem solving has often been directed at examining the behaviors of experts performing in relatively well defined problem domains (e.g. chess playing, computer

programming, medical diagnosis, complex arithmetic calculations). A chess match or the writing of a computer program can be observed from start to finish in a few hours. On the other hand, a more creative activity, such as invention, may take weeks, months, or years to complete. One may also easily determine whether a computer program or a chess strategy was successful. An invention or a novel is not so easily judged as successful or unsuccessful and may well depend upon its placement in time and space (eg., works that are largely ignored until years, decades, or centuries after their creation; conversely, the creation of an incandescent light bulb would be seen as creative in the 1800's but rather conventional in the 1990's). It is therefore, relatively simple to observe and manipulate initial states, intermediate states, and objectively-defined end states in problem solving domains such as chess playing or computer programming. Furthermore, the goals in traditional problem solving domains are usually well defined, and the solutions often result from recombinations of standard approaches (e.g. using the same I/O routines or similar algorithms in different programs, using the same opening or middle game strategies in different chess matches, etc.). Additionally, the constraints involved in a chess match

or the writing of a computer program are clearly defined (e.g. a king can move one square in any direction, each PASCAL statement must end with a semicolon, etc.). A substantial portion of the processes which allow for the successful solution to problems in these areas can therefore, be classified as conventional problem solving skills.

The distinction between conventional and creative problem solving does not imply that one class of activity is completely devoid of creativity and that the other is a wholly creative enterprise. Generally speaking, when one begins to write a computer program, the programmer has a fairly clear idea of the function that the program is to perform, the computer language in which it will be written, many of the operational constraints (e.g. syntax, memory capacity, value ranges, limiting cases), the user audience, and may even know which previously written blocks of code will be reused in the new program. When an author begins a new work however, she may have only a few ideas about the characters, the actions they will take, the ultimate outcome, the purpose of the work, or who will read the text. Classifying a given problem solving activity as creative or conventional is determined by the degree of creative activity required to develop a

satisfactory result, and not the domain in which the problem solving is performed. From this perspective, computer programming is a problem solving domain that requires relatively fewer creative behaviors than invention or novel writing, but any specific problem solving activity within the programming domain cannot be judged as conventional, or creative, based solely upon the domain in which it occurs. The distinction between creative and conventional problem solving domains is the number and centrality of problems within a domain which require a novel solution by means of a difficult, poorly constrained, unconventional problem formation. The judgement of creative or conventional is therefore a relative distinction. That is, domains such as medical diagnoses and computer programming may be judged as conventional when compared to invention, but programming is certainly a creative domain when compared to solving two digit multiplication problems.

Areas of expertise in which creativity plays a large, central role are difficult to study in a controlled laboratory environment because creative problem solving involves poorly defined starting states, subjectively-defined end products, poor, or loose, constraints on the variety of useable techniques, and may take place over long time spans.

Traditional approaches to the study of expertise, sometimes known as knowledge engineering, have involved four general steps (Hayes-Roth, Waterman, & Lenat; 1983). (a) Interview and re-interview experts. (b) Observe experts performing skills of interest, and obtain protocols based upon observations and interviews. (c) Propose possible processes, characteristics, knowledge structures, and heuristics which would account for the superior performance of experts. (d) Build a system based upon those constructs, and verify that the system operates effectively in the domain of interest.

Some researchers have questioned the reliability of findings based on interviews with experts (Nisbett & Wilson, 1977). Several other researchers, however, have indicated that verbal protocols can be quite reliable if subjects are asked to describe their actions and not explain or make inferences about their actions (Ericsson & Simon 1984, 1980; Kellog, 1982; Kellog & Holley, 1983). Furthermore, the utility of the knowledge engineering approach has also been successfully demonstrated several times by the creation of effective expert systems (e.g. Dendral, Mycin, Prospector, to name just a few).

Even with the issue of verbal report reliability

aside, creative activities, such as invention, take place over a long span of time and it is therefore difficult to develop meaningful, complete descriptions of invention which are based solely upon the verbal reports of experts. In addition, interview methodology is a labor intensive, tedious process which generally requires months to complete, even when used in rather restricted areas of conventional domains. The product of such efforts is usually large quantities of detailed information about highly specific realms of expertise. Consequently, using results from interview data to make meaningful comparisons across knowledge domains becomes an arduous, if not impossible, task.

To further complicate matters, obtaining protocols can also be expensive in terms of materials due to the largely unconstrained nature of creative activity. An expert inventor, for example, may develop many prototypes before a satisfactory result is obtained. It would therefore be advantageous to develop short-term laboratory methods, which use simple neutral stimuli, to collect more manageable data about the characteristics of expert creative problem solvers.

In a related project, another researcher from our lab conducted preparatory interviews with expert inventors and proposed several processes that might be

important to invention (Weber, Moder, & Solie, 1990). One of the most promising constructs identified in the interview data was the ability to parse an object into its components. In short, parsing delineates the loci of variations within a particular object and yields a knowledge structure which is similar to the framework devised by Minsky (1975). In Minsky's framework (sometimes referred to as frames and slots), knowledge categories (e.g. fasteners) are defined by their various instances which contain at least one common characteristic. For example, safety pins, straight pins, buttons, snaps, screws, and velcro may range in their physical similarities, but they all share the common feature of being able to fasten two surfaces. Each device, therefore, is an instance of the category of fasteners. The instances of categories may be referred to as frames, and are in turn composed of various characteristics or slots (e.g. the slots of a safety pin might be spring, clasp, brace, pin).

Parsing an object utilizes a similar type of knowledge structure. For example, if we wished to develop a new type of safety pin (frame), we might start by breaking down the safety pin into its components of a spring, clasp, brace, and pin (i.e., parsing the safety pin into its slots). We might then

alter those components in several ways: we may combine them with components of other objects, we may change their orientation to each other, we may replace the component with an analogous component, we may enlarge one or all of the components, etc. Furthermore, parsing may take place on several levels. Just as we can parse an object into components, we can also parse components into sub-components. For example, a clasp is a component of a safety pin. The material, size, shape, operation, or function of the clasp can also be parsed. Parsing simply provides the basic data structure upon which other transformational processes can operate and may therefore be a fundamental inventive process.

We know by definition that expert inventors are better inventors than are novice inventors, and if parsing is an important inventive procedure, then experts may well have different parsing patterns than do novices. Furthermore, parsing patterns generated by experts should be "better" than those generated by novices. Experts may be faster at generating parses, or they may generate more parses than do novices. de Groot (1965, 1966), however, found that expert chess players did not generate more moves in a smaller amount of time, but instead, generated qualitatively better

moves. That is, expert chess players generated moves with a greater chance of success or moves which more efficiently accomplished a goal.

Chase and Simon (1973) used an ingenious technique (sometimes known as the "quick glimpse" method) for accessing differences between expert and novice chess players. Both novice and expert players were presented with several chess board configurations and were asked to accurately reassemble the pattern onto a second chess board. The boards, however, were positioned so that the subjects could not look at both boards simultaneously. Therefore, the subjects were forced to hold the test pattern (or part of the pattern) in memory while they reconstructed the pattern on the answer board. In analogous terms, they were forced to "parse" the chess board configuration because the number of chess pieces exceeded short-term memory capacity. Furthermore, two types of patterns were used as test items: true board configurations, and nonsense configurations. True board configurations were arrangements which could actually occur during a chess match. Nonsense configurations were illegal and random patterns of chess pieces that would have a zero probability of occurring during a chess match. The experts performed no better in the nonsense condition

than did novice chess players. In the true board condition, however, experts performed markedly better than did novices.

The Chase and Simon (1973) study illustrates some important distinctions between novice and expert chess players. First, it indicates that chess experts do not possess a superior short-term memory relative to novices. Otherwise, the experts would have out performed the novices in both the true board and nonsense conditions. Second, the study indicates that expert chess players are better at remembering actual chess configurations than are novices. This is most likely because experts have seen the patterns many times and through practice, the configurations have been "chunked" (Miller, 1956) into meaningful units. This common result of practice and elaboration can be demonstrated easily by trying to remember and repeat the letter string: e, d, s, l, u, e, c, h. This is a difficult task for most people, but the recall task becomes much easier when the same letters are combined to form a meaningful unit (e.g. s,c,h,e,d,u,l,e,). Thus, in a manner of speaking, chess masters have learned to read the chess board like a book, while the novices are still learning their ABC's. (Similar results have been obtained by de Groot 1965, 1966;

Charness, 1976; in computer programming by McKeithen, Reitman, Rueter, & Hirtle, 1981; and Shneiderman, 1980; in physics by Chi, Feltovich, & Glaser, 1981).

Analogously, parsing may yield the memory chunks of expert inventors. The relationships among a specified pattern's components determine which components belong to which chunks, and the nature of the relationships, both within and between chunks, is determined by the subject's previous experience within the knowledge domain. Parsing identifies/generates groups of related components (chunks) present in a given pattern and makes the members of that parse (chunk) available for other transformational processing (in this case, replication).

If the above findings in conventional problem solving domains can be generalized to the more creative problem solving realms such as invention then expert inventors should be able to divide objects into larger units which would allow them to remember the structure of those machines better than would novice inventors. That is, if the results of the Chase and Simon (1973) study are applicable to invention, and parsing is an important inventive process, then experts at machine invention should produce larger, and thereby fewer, parses than novices in reconstructing the same machine.

Furthermore, the experts and novices should require an equivalent number of parses to reconstruct a non-functioning nonsense object. The current study was conducted to 1) further the understanding of cognitive processes used in the inventive process, 2) determine whether classic research findings from more conventional problem solving domains could be effectively demonstrated in a more creative area of human problem solving, and 3) use a short-term investigative strategy, which utilizes simple neutral stimulus materials, to yield data that would permit relatively straight forward comparisons with research findings in other problem solving domains.

Method

Subjects

The sample consisted of two expert inventors (each had designed one patented mechanical device), and two novice inventors (senior level, university undergraduates who did not have a patent). All four subjects were males. The ages of the inventors were 47 and 49. The ages of the novices were 19 and 22. The experts volunteered to participate and received no reimbursement for their participation. The novices were volunteers and received two extra credit points in their experimental psychology course for participating.

All four volunteers stated that they had worked with Tinker Toys as children and that they spent an average of less than 30 minutes a day working with tools or machines. On a scale of one to seven with seven being the highest degree of enjoyment, both experts rated their enjoyment of working with tools as higher than average (5 & 6). Both novices rated their degree of enjoyment when working with tools as lower than average (1 & 3). The novices estimated their mechanical aptitude as near average (3 & 4; on a scale of 1 to 7 with 7 being highest). The experts also rated their mechanical aptitude as near average, although slightly higher than the novices' ratings (4 & 5). Both novices and one expert denied any formal training in mechanical design. The other expert had earned a Masters degree in mechanical engineering.

Materials and Apparatus

A Pentax color video camera (model# PC-K1500A) and a Pentax VHS format portable video cassette recorder (model# PZ-R1100A) with time stamping (accurate to .1 sec.) were used to record the experiment. A Magnavox VCR (Model# VR9750AT01) with remote control for super slow motion and individual frame advance, connected to a KMC television (model# KMC-1921G) were used to score the tapes.

The two stimuli (mechanical and nonsense objects) were constructed from Tinker Toys (a building set sold by CBS Toys). The mechanical object was constructed using 79 Tinker Toys and was an operational model of a windmill. The mechanical object was operational in the sense that it was composed of a large fan which spun in response to certain forms of external forces (i.e. a breeze, spinning it with one's finger etc.). The energy of the fan was transferred, via two gears, to a vertical shaft which would spin as a result of the external energy applied to the fan blades.

Insert Figure 1 about here

The nonsense object was constructed of a set of 79 Tinker Toys which was identical to the set used to build the mechanical object. The nonsense object resembled no discernable machine, had no moving parts, and did not model any obvious utilitarian activity. The nonsense object was designed by way of a pseudo-random construction procedure. First, the set of 79 pieces were placed in a box. The box was then shaken, opened, a piece drawn, and connected to whatever structure existed. If there was no place for a piece to be connected, it was set aside until an opportunity

to be connected arose. The process continued until all pieces had been removed from the box and connected to the structure.

Insert Figure 2 about here

The stimulus items (mechanical and nonsense models) were individually placed in a blind constructed of three white poster boards (side = 16"w x 34"h, front = 24"w x 34"h, top = 24"l x 18"w). The front and side of the blind had slits (8"w x 6"h) which were covered by flaps of white poster board (10"w x 8"h). The blind was set atop a standard typing table which had one end leaf down. This placed the two slits at approximately eye level (46" from the floor) for an adult sitting in a standard four-caster office chair.

Insert Figure 3 about here

A wooden conference table (10'l x 32"w) was placed length wise in the lab room (12'l x 5'w). Two sets of Tinker Toys (building set #550; approx. 115 various pieces per set) were placed at one end of the table. The subject sat in a four-caster, padded office chair at the same end of the conference table where the

Tinker Toys had been placed (Figure 3 is a sketch of the lab configuration). A line was drawn on the table in order to define the work area. None of the construction materials or the subjects' construction activities were to take place beyond the line because they would be outside of camera range. The blind was stationed 1' to the left of the table and 1.5' beyond the end of the table. The positioning of the table, work area, and blind allowed the subjects to easily move back and forth between either blind slit and the work area. The camera, VCR, and an Amdek Color-I monitor were placed at the opposite end of the testing room. The camera was placed on a tripod and was adjusted so that the subject, the work area, and the blind flaps could be simultaneously video taped. The monitor was placed so that only the experimenter could see the screen.

Design and Procedure

All subjects were tested individually, and replicated both the mechanical and nonsense objects. The order of presentation was balanced across both groups (i.e. half of the experts saw the nonsense object first as did half of the novices).

Upon entering the laboratory, subjects were asked to read and sign a volunteer's consent form. After the

subject signed the consent form, the video camera and recorder were activated. The experimenter then asked several demographic questions. (The results of which are stated in the subjects sub-section.) After gathering this information, the experimenter read the task instructions to the subject and answered any questions. The instructions informed the subject that their task was to build a replica of the object which was in the box beside them. They were further informed that they could look in the box as often as they liked, but they could only look at the object through the two flaps. Participants were also instructed to work as quickly as possible, and not to remove the Tinker Toys from the work area of the table. This was done to prevent subjects from grabbing several materials, turning to the box, and assembling the parts in their hands while directly viewing the test object. Not allowing subjects to simultaneously view the replica and original, forced them to depend upon memory in order to complete the task. After subjects stated that they understood the instructions, subjects were given three minutes to familiarize themselves with the construction materials in the work area. Following the three minute period, the experimenter removed the top of the blind to illuminate the object within and asked

the subject to look through the side flap at the object. Both stimuli were viewed for 60 seconds through the side flap, but in the mechanical condition the experimenter turned the fan blades in order to demonstrate all the moving parts. At the end of the initial viewing, the subject was asked to lower the flap, return to the work area, and look toward the camera. The experimenter then responded to any remaining questions. When all questions were answered, the experimenter asked the subjects to begin and started the camera's internal clock.

After the subjects had accurately replicated the object, they were asked to improve the replica in as many ways as they wished. Subjects were also asked to "think aloud" while they performed this portion of the task. After answering subjects' questions, the experimenter asked them to begin and started the clock.

Upon completing their improvements, subjects were asked to completely "break down" the improved replica so that no Tinker Toys were connected. This was followed by a 10 minute rest period. During the rest period, the subject left the testing room, and the experimenter exchanged the object in the blind for the remaining object. After the 10 minute rest period was over, the subject returned to the testing room, and the

procedures used for the first object were repeated using the second object. Upon completion of the tasks involving the second object, subjects were notified of the specific purposes of the study, the experimenter answered any remaining questions, and the subjects were discharged.

Results

Scoring

The video tapes were scored for number of looks, the elapsed time during each look, the number of connections made during each "look", the number of disconnections (errors), and the total time required to assemble each object. A "look" trial began when the subject's eyes were directed towards one of the two blind slits, the flap covering the slit was raised past the subjects eye level, and the subject's eyes were open. The look trial ended when the subject's eyes looked away from the slit, or the flap was lowered past eye level. The total assembly time was scored as the elapsed time between the experimenter saying "begin" and the subject completing his final connection.

While on the surface the above definition of a look may seem complicated, it yielded very reliable scores. Two separate scorers scored randomly selected, three minute, segments of tape for each subject in each

condition. Both scorers generated identical scores for each tape segment.

The mechanical object required 83 connections and the nonsense object required 80 connections. Extrapolating directly from the Chase and Simon (1973) methodology, the number of Tinker Toy pieces assembled during each trial seemed to be a reasonable dependent measure. When scoring began, however, it became apparent that some pieces were assembled when connected to one other piece while others required as many as nine connections to be considered assembled. For example, a short Tinker Toy rod which served as a gear tooth was fully assembled when one end was connected to the gear hub. The gear hub, however was not assembled until all eight gear teeth and an axle were inserted in the hub's available openings. That is, each gear tooth required only one connection while other materials had to be connected to as many 9 other pieces (gear hub) before they could be recorded as assembled. Thus, a subject might work on assembling parts of several pieces in the same look but not fully assemble any one piece. If number of pieces assembled was used as the dependent measure then these activities would generate a score of 0 for that look. It was also possible that a subject might only make three connections during a

look which would result in three fully assembled pieces and a score of three for a given trial. Thus, even though the subject was more productive in the previous situations, the dependent variable would indicate no subject activity for those trials.

Number of pieces assembled, therefore, was not an accurate representation of the subjects' performance in each trial. The number of connections made, rather than the number of pieces assembled, was therefore scored as the primary dependent measure. The number of disconnections was also recorded and treated as error data.

Analysis

The first analysis used a 2 (object) x 2 (order) analysis of variance for a mixed design to investigate order effects (mechanical-nonsense v. nonsense-mechanical). No significant difference due to the order in which stimuli were presented to subjects was detected ($F(1,2) = 5.76$, n.s.).

For the remaining analyses, a 2 (object) x 2 (expertise) analysis of variances for a mixed design was computed for the average number of correct connections made during non-zero trials. On several of the look trials, subjects would look at the original and then look at their reproduction without connecting

any pieces. These were considered zero-trials and were excluded from the first analysis. The average number of correct connections was computed by dividing the number of connections required (83 for the mechanical object and 80 for the nonsense object) by the number of productive looks for that condition. In the nonsense condition, novices made an average of 1.84 correct connections per look while experts made 1.68 connections. In the mechanical condition novices also outperformed experts, although the differences due to expertise were not statistically significant, $F(1,2) = 1.60$. The nonsense object was significantly more difficult to replicate than the mechanical object, $F(1,2) = 32.62$, $p<.05$.

Insert Figure 4 about here

Figure 5 represents the outcomes when zero trials were included in the analysis. The variance due to object type increased ($F(1,2) = 70.25$, $p < .025$), but

---

Insert Figure 5 about here

---

novices still performed better than experts ($F(1,2) = .33$, n.s.), although to a lesser degree. In other

words, novices took more non-productive looks at the nonsense objects than did experts (see Figure 8). The interaction between expertise and object type were nonsignificant in both analyses ($F(1,2) = 3.42$) when zero trials were excluded; $F(1,2) = 2.63$ when zero trials were included).

Experts took longer to complete both replication tasks than did novices (See Figure 6). The nonsense object required more time to complete even though it required fewer connections than the mechanical object. The effects due to expertise ($F(1,2) = .1$) and the interaction ($F(1,2) = .03$) were statistically non-significant when elapsed time was used as the dependent measure. The only statistically significant difference in elapsed time was due to the object being replicated, $F(1,2) = 41.05$, $p<.025$.

Insert Figure 6 about here

As is depicted in Figure 7, experts made more disconnections than did novices, and more disconnects were performed in the nonsense condition than in the mechanical condition. However, neither main effect was significant ($F(1,2) = 1.18$; $F(1,2) = 2.95$, respectively), nor was the interaction, $F(1,2) = 1.2$.

Insert Figure 7 about here

The only condition in which experts outperformed subjects was the number of looks required to assemble the nonsense objects (Figure 9). Experts required an average of 131.5 looks to assemble the nonsense objects whereas novices required an average of 177.5. In the mechanical condition, however, experts averaged more looks than did novices. Once again the differences due

---

Insert Figures 8 & 9 about here

---

to expertise were non-significant, $F(1,2) = .15$. The differences due to object type ($F(1,2) = 8.55$) and the interaction between object type and expertise ($F(1,2) = .5$) were also statistically non-significant.

Discussion of Empirical Findings

The pattern of data resulting from the current study did not produce interaction and main effects consistent with the findings of Chase and Simon (1973). In the present study, novices consistently outperformed experts as measured by any of the various dependent measures. The mechanical object, however, was easier to replicate than the nonsense object, and the

differences due to object type were the only effects that reached statistical significance.

Although sampling error is the most likely explanation for the statistically non-significant differences due to expertise, a few other extraneous influences may have also been at work. First, due to difficulties obtaining a sample of expert inventors, the average age of the novices (20.5) was substantially less than the experts' mean age (48). The novices' younger eyes and hands may have provided a performance advantage during the replication task. Also, the younger novices may have been slightly more motivated because they were receiving extra credit in their psychology class, and the experts were volunteering their time with no form of reimbursement. During the experiment, however, all subjects appeared enthusiastic, cooperative, and equally motivated to perform the tasks.

Even though the above mentioned factors were present, I believe that they had little, or no, effect on the experimental outcome. I think the lack of any statistical differences between experts and novices in this task is legitimate, and the task, therefore, was not able to distinguish between expert and novice inventors. Experts and novices performed at

approximately the same level in all object conditions, and the primary factor accounting for the lack of differences is most likely a ceiling effect. While the nonsense object appears to have been more difficult than the mechanical object, both objects may have been too simple and did not require the use of specialized knowledge to perform the replication task. Had the objects been more complex structures with a more sophisticated design then the task may have required the experts to utilize more unique knowledge structures. Likewise, a more sophisticated design may have pushed the novices to the limit of their mechanical knowledge. The novices, lacking any specialized inventive or mechanical abilities to cope with such complex stimuli, would demonstrate a performance deficit relative to the experts. Slight support for this view comes from the number of looks required to replicate each object (see Figure 8). The experts required fewer looks to replicate the nonsense item than did novices. One implication of this result is that the increased complexity of the nonsense object may have affected the experts less than the novices. While such a singular result is interesting it is still highly suspect. Again, one must remember that the result was statistically non-significant and the

observed differences are most likely due to common sampling error.

One of the goals of this study was an attempt to develop a relatively simple, short term method to study the nature of human inventiveness. Thus, the choice of a simple, neutral machine versus a intricate specialized machine was made. There was no a priori means to determine how simple, or complex, that machine had to be, and the costs of the design complications introduced by the use of a more complex stimulus were much too high. Even now, it would be difficult to determine how sophisticated the machine should be. For example, most inventors from mechanical fields probably possess a general mechanical aptitude and a high degree of specialized knowledge within the field in which they have received patents. In the current study, one expert received a patent for an improvement in air conditioning design, and the other received a patent for a modification to a piece of agricultural equipment. While it is true that both of these subjects would probably have little trouble changing a faucet washer, it is also very difficult to say that the air conditioning expert could have as easily developed the agricultural invention and vice versa. The two experts, while possessing similar general

aptitudes, and perhaps similar inventive aptitudes, simply do not possess the same specialized knowledge that allowed them to create their respective innovations. Therefore, if a fair comparison is to be made, the mechanical object should be of substantial complexity to access any unique knowledge structures of the expert inventor. If the mechanical device is of such complexity then comparing the expert's performance to completely naive subjects would be an unfair comparison. It would be impossible to determine whether any observed performance advantage for the expert should be attributed to special inventive abilities or simply to greater familiarity with the mechanics of the test item. Therefore, the subject population would also need to be changed. For example, if one could assemble a sample of experts who had a patent in air conditioning design then an appropriate novice population might be a sample of air conditioning service technicians who possessed no patents.

Obviously, using such procedures would add a great deal of time and expense to locate and reimburse such professionals for participating in the study. At the beginning of this study, it was my hope that such complicated studies could be avoided, and I therefore attempted to determine whether the use of a neutral,

simple machine could be used to demonstrate any cognitive abilities unique to inventors. Unfortunately, no concrete conclusions about the nature of human inventiveness have been uncovered as a result of the current study.

The fact that experts and novices performed at an equivalent level when given neutral, relatively simple objects to replicate is, however, consistent with the idea that the specialized cognitive structures which make invention possible are most likely tightly bound to the specialized knowledge of the domain in which the inventor works. The possible exception to this rule may be those rare cases of particularly gifted inventors (e.g. Edison) who seem able to translate their inventive capacities to a variety of fields. Even in these rare cases, however, it seems clear that such inventors must exert great effort to become competent in a new domain, and only after a obtaining a relatively high level of understanding of a new domain can they develop truly unique inventions. (I will return to this point and attempt to describe why this may be the case in the last section of the paper.)

While the primary goal of the research was to identify processes and abilities that distinguish inventors from the general population, a secondary goal

was to develop a methodology which would allow for relatively straightforward comparisons between this study and other studies of expertise in diverse domains. Furthermore, I wanted to develop a methodology which eliminated the complexities involved with protocol analyses of experts' verbal reports. The project was only partially successful in achieving these two secondary goals. Conceptually replicating the Chase and Simon (1973) study did provide data that allow reasonable comparison between this study and similar studies in other domains. Furthermore, I was able to investigate inventive expertise without the use of verbal reports.

The present methodology unfortunately introduced an entirely new set of analytical difficulties. First, scoring the tapes was a time consuming process to put it mildly. Every minute of tape required approximately two hours of effort to score and verify. The present data represents approximately 160 hours of scoring not including analysis, time required to set up the scoring equipment, or breaks required to maintain accurate scoring. When all the other "real-time" factors were included, more than one semester of daily scoring activity was required to score and verify the data from this study. Therefore, while the data generated from

the current methodology are somewhat more objective than the data commonly generated from experts' introspections about their actions, the scoring effort remains comparable to the effort required by protocol analyses.

The major cause of the difficulty surrounding the scoring was the vigilance required to score the onset and offset of looks. Scoring the number of looks was originally planned to be a simple matter of counting flap openings and closings. Unfortunately, the scoring quickly became a process in which the scorer had to determine when the flap was raised past the eyes, the eyes were directed toward the flap, and the eyes were open. Simply counting the number of flap openings and recording the length of time that the flap is open would have resulted in inaccurate results. Subjects tended to open the flap and glance at the stimulus (look beginning), then glance at their replication (look ends), and then back at the stimulus (a new look begins). They would often look back and forth several times before closing the flap once and returning to the work table. Although determining the point at which a look begins, and ends, involves monitoring three conditions, and consumed much of the scoring effort, the use of a super slow motion VCR with frame advance

made the determination very accurate and reliable.

As is generally the case in science, the result of the present study is a mixed bag. The research did not produce a methodology which accesses invention processes without requiring labor intensive scoring efforts. The study did, however, produce objective data that could be compared to studies from other knowledge domains. Unfortunately, the data did not produce results consistent with the Chase and Simon (1973) findings, but the results do hint at one potentially important characteristic of inventors. Before an interested individual can become an inventor, it may be necessary to acquire extensive knowledge of a domain before she can create a unique, patent quality, invention within that domain. While inventive ability and domain knowledge are hardly synonymous, they may be highly inter-related. That is, inventive processes are most likely high level processes that can only utilize knowledge structures which have a compatible level of sophistication. In other words, the inventor may have to obtain a fairly high level of comprehension within a domain before any useful, unique, inventive procedures can be effectively applied. Therefore, the development of experimental techniques, which can disentangle the inventive procedures from the domain specific

knowledge, may hold the greatest promise for unlocking the nature of human inventiveness.

Theoretical Discussion

Two Perspectives on Cognition

Following World War II, a new technology, the computer, was beginning to proliferate throughout the scientific and technical community. The von Neumann architecture was, and still remains, the most widely applied computer architecture. A von Neumann machine contains two primary components: a central processing unit (CPU) and a central memory array. The CPU contains registers which are small memory elements that can contain one "chunk" of information (usually an address of a central memory location, or a numeric value read from an address in the central memory). The machine operates by executing a pre-programmed sequence of instructions which is stored in the central memory array. The CPU begins by fetching the first instruction in the sequence from the central memory array. Next, the CPU performs the operation indicated by the instruction and may write the result of the operation back out to a location in the central memory. The CPU then obtains the next instruction in the sequence and the process repeats itself until the final instruction in the sequence is executed. In the von

Neumann architecture, the CPU and memory are separate components and information is represented explicitly in individual memory locations. The processor of a von Neumann machine can, at any one time, perform only a single operation from a sequence of operations, and von Neumann computers are, therefore, best described as serial machines.

The serial computer provided scientists and engineers with a tool which was flexible enough to model complex systems and thereby allowed them to build, control, and attain a higher understanding of such systems. The human brain is certainly one of the most complex systems known to science, and even though human cognition is probably far more intricate than the most sophisticated post-war computing process, it did not take long before researchers began drawing analogies between human thought and computer processing (Turing, 1950; von Neumann, 1958). By the mid 1960's the computer-mind metaphor was the dominate metaphor driving psychological investigations and the von Neumann machine was firmly entrenched as the dominate computing device. It should therefore come as no surprise, that human cognition had become characterized in terms of serial information processing stages. Viewing human cognition in terms of sequential symbolic

processes, resulted in substantial advances in computer language designs, human factors, linguistics, artificial intelligence, human learning and memory, cognition, and human problem solving. Almost from the beginning of the information processing revolution however, there were detractors who warned against taking the serial computer-mind metaphor too literally.

A major criticism of the computer-mind metaphor arose from the differences between the actual neurological organization of the human brain and the dualistic organization of process and memory in the von Neumann computer. How could scientists, who were supposed to be concerned with truth and accuracy, so willingly embrace serial models of cognition when there existed such an obvious discrepancy between the models and the supposed underlying physiological mechanisms? The answer was, of course, that the information processing theories operated at a level of explanation which was higher than the level of explanation utilized by neurological theorists. The information processing camp affirmed the symbol as the fundamental component of human cognition and thus, their explanations supposedly did not extend to processes which operated below the symbolic level. The information processing investigators undertook the task of creating symbols

for objects, relations, operations, and any other cognitive component that they deemed pertinent. The human mind was viewed as a symbol processor and any process which operated below the level of the symbol was seen as a topic more suited to neuroscience.

The last two decades of advances in AI/expert systems, human problem solving, cognitive psychology, and the information sciences serves as a testament to the power of explanation at the symbolic level. However, as all worthwhile paradigms must do eventually, the conventional information processing approach illuminated human cognitive capabilities which could not be adequately modeled with a sequential symbolic processor (e.g., continuous speech recognition, dynamic pattern recognition, visual scene interpretation, content addressable memory, and autonomous vehicles). The processing models of the neural realists had always seemed to possess the potential for dealing with such problems, and by the 1980's the technology and computational models had developed sufficiently to use them successively in both applied and theoretical realms.

The modern work on artificial neural models actually began more than 40 years ago with work done by Hebb (1949), McCulloch and Pitts (1943), and Rosenblatt

(1959) but, their efforts were stymied by a lack of technology and the early successes with symbolic processing systems. The recent resurgence in neural theories began in earnest during the mid 1980's with work by Hopfield (1982, 1984, 1986), Kohonen (1984), Grossberg (1986a, 1986b), Feldman & Ballard (1982), Hillis (1986), and Rumelhart and McClelland (1986a, 1986b) and represents an exciting possibility for cognitive, neurologic, and computing research. Undoubtedly, neural net theories will be a great boon to those who are working on "monster" AI problems such as those mentioned above, and to neural scientists who wish to model human neural systems on a computer. One of the most exciting prospects for neural computing, however, lies in coupling the past successes of the symbolic serialists with the power provided by the new neural networks. For the first time, we may possess the theoretical rudiments necessary to begin developing a complete model of human cognition which encompasses the higher level, apparently sequential processes of the human mind and the highly parallel mechanisms which underlie those processes. The purpose of this portion of the dissertation is to explore how neural and traditional models may be integrated so that new insights might be gained about one aspect of high-level

cognitive processing, namely, human problem solving.

The first section will provide an overview of Newell and Simon's (1972) influential work on human problem solving by discussing the underlying assumptions of traditional human problem solving theory, defining some basic terms, and describing the general process of problem solving. The following section will provide an overview of neural networks by discussing the underlying assumptions of neural computing, defining fundamental terminology, describing two representative neural nets, and lastly, highlighting the tasks which neural nets most easily lend themselves. The next section represents a first attempt at uniting the two views of cognition by describing a common data structure, formulating an integrated processing model for applied expert systems problems, and discussing a general, unified model of creativity and invention. The final section will summarize the previous sections and make some rather modest predictions for the future of neural computing and problem solving.

The Conventional Information Processing Perspective

In 1972 Allen Newell and Herbert Simon authored the highly influential book, Human Problem Solving,

which, as the title implies, was an effort to advance the understanding of human problem solving. The fundamental postulate of the Newell and Simon theory is that humans operate as an information processing system or IPS (p.19). According to Newell and Simon, an IPS consists of three primary components: I/O mechanisms, a processor, and a memory (see Figure 10). Furthermore, an IPS resides in an external world that contains the task information and all the physical entities with which the IPS must deal.

Insert Figure 10 about here

The three components of the IPS, (the I/O mechanisms, the processor, and the memory), comprise any IPS's general architecture. The I/O mechanisms are divided into two general categories: receptors and effectors. The receptors receive information from the external world and pass it to the processor. If the IPS of interest is a human then the receptors are analogous to eyes, ears, and the other senses. If the processor selects an output which must be manifested in the external world, the action is performed by the effectors. Again, if the IPS of interest is a human then the effectors are abstractions of hands, feet,

mouth, or any other physical component which can directly affect the external world. Newell and Simon divorce their problem solving theory from any detailed concern with the sensory mechanisms and it should be made clear that the primary components of an IPS operate at a level that is more abstract than the sensory or motor processes. The processor consists of three components: a fixed set of elementary information processes (eips); a small, limited short term memory (STM) that can hold only a few symbol structures at any one time; and an interpreter which determines the sequence of operations to be executed by the IPS. The last component of the IPS is the large, virtually unlimited long term memory which holds the symbol structures until the processor requires them in its short term memory.

The IPS operates by accumulating information about the external world via the receptors. The input from the receptors is passed to the STM of the processor. The processor then locates, by invoking eips, symbol structures in memory that represent the external objects and events. The symbol structures are then passed to the processor and based upon the pattern of activated symbol structures (i.e. the context), the interpreter composes sequences of operations from the

set of available elementary information processes. The resulting operations may animate the effectors and/or cause the activation, modification, and/or creation of symbol structures in memory.

In specifying an IPS capable of performing a desired task, one must first define a set of fundamental symbol manipulation procedures known as elementary information processes (eips). The eips combine with the symbol structures to define an IPS's total range of capabilities. The IPS's entire behavior is produced by executing sequences of the eips. These simple processes can be combined to build more and more complex procedures until an integrated problem solving behavior emerges. Just what makes a process an elementary process depends upon the purposes of the particular application. The eips must be general and powerful enough to generate the full range of behaviors necessary to solve a specified problem. Furthermore, the eips must be realizable by known mechanisms. For example, there is no reason to take problem solving as an eip for it would tell us nothing about how problem solving is accomplished. The set of possible eips is not unlimited, but a unique set of eips capable of resolving all problems does not currently exist. In fact, the quest for a limited set of eips capable of

resolving all symbolically representable problems is one of the computing sciences' cornerstones.

Perhaps Newell and Simon's most ubiquitous theoretical element is the symbol, and a primitive symbol is the most basic form of a symbol. Like the eips, the definition of a primitive symbol depends upon the current application. For example, if one wished to define an IPS capable of understanding speech, then one might select the phonemes as the atomic structures of speech and assign a primitive symbol to each phoneme. By designating phonemes with primitive symbols, the resulting model of speech processing would disregard any process or knowledge which occurs below the phonemic level (e.g. sound wave forms, the physiological processes by which phonemes are detected, and the perceptual processes by which phonemes are recognized). In general, the definition of eips and primitive symbols is determined by the degree of specificity with which one wishes to define an IPS.

Primitive symbols perform three primary functions. First, primitive symbols designate specific events or structures in the external world. Such primitive symbols may be evoked when their referent occurs in the external world of the IPS, or their presence within the IPS may cause the IPS to create such events or

structures externally. Second, primitive symbols may designate eips or sequences of eips so that the referent can be activated by the interpreter. Lastly, primitive symbols can be connected via relations to produce more complex symbol structures.

As an example, consider the action of turning on a light switch. First, one must define the eips for the IPS which is to perform the action. For present purposes, the IPS has four eips available to it (locate-switch, touch-switch, move-finger, return-to-previous-state). It should be clear that the words/symbols which designate eips are just labels. That is, the symbols have no inherent meaning to the IPS other than being a unique designation for the action(s) to be performed. I could have labeled them A1, A2, A3, and A4, but I have instead chosen to designate them with symbols which more adequately describe their actions to us as external observers of the IPS. We can now combine these four actions (eips) into a sequence of actions and designate the sequence with the symbol "flip-switch". Now if the IPS encounters a situation in which the context requires the flipping of a light switch, it need only activate the symbol structure "flip-switch" and the four step sequence of "locate-switch", "touch-switch", "move-

finger", and "return-to-previous-state" will be executed.

The advantage of viewing a problem solver as an information processing system lies in the fact that an IPS uses an interpreter which only requires a small, finite amount of mechanism (i.e. the eips). However, symbols, which can be used to designate eips, can be arranged in very complex ways. Therefore, the complexity of an IPS's behavior (sequences of eips) is limited only by the complexity of the symbol structures that can be built up in memory.

The power of symbols lies in their ability to designate (i.e. to have a referent). The primary designatory relationship is between a symbol and a symbol structure . Thus, the symbol X2 may designate the symbol structure (C A T). The ability to designate means an information process can take a symbol as input and gain access to the referenced object, or action, in order to affect it or be affected by it in some way. For example, an information process is given the symbol TABBY which refers to the symbol structure (own cat black male). Likewise, if the information process is given the symbol structure (own cat black male) the symbol TABBY can be produced. Given the symbol or symbol structure an information process can then

operate on that symbol, the object in the external world to which it refers, or use that symbol structure to affect another symbol structure. In short, a symbol can be used to encode information about any conceivable thing and can hence operate as a surrogate for that thing within the IPS.

In general, an IPS operates by locating symbols and performing sequences of actions (eips) based upon the context provided by the symbol patterns held in the processor's STM. The component of the processor which determines the sequence of operations is the interpreter. The fundamental assumption of the Newell and Simon theory is that an interpreter will operate in a lawful fashion. That is, given the exact situational context (same pattern of activated symbols), an interpreter will generate a functionally similar or exact sequence of operations. The behavior of an IPS is therefore predictable if one can determine the conditions under which certain sequences of behaviors are generated by its interpreter. In accordance with this assumption, a fundamental task for an inductive scientist, who wishes to study human problem solving, is to observe an IPS in order to hypothesize a <u>program</u> for the IPS's behavior. A program, in Newell and Simon's terms, is a set of rules and regularities that

describe the sequences of eips which the interpreter executes as a function of its current informational context.

The second general obligation of problem solving theorists, according to Newell and Simon, is determining the extent to which the IPS actually runs according to the program which has been specified for it. It should be made clear that a program is purely external to the IPS. A program is our way, as external observers, of describing the system. A program should be understood as a theory which describes the operation of a system in information processing terms. If the interpreter of an IPS is a true interpreter (i.e. generates sequences of operations based upon symbolic structures in predictable ways), then we can also describe the internal structure of the interpreter with information processing terms. However, there may be nothing inside the system itself that corresponds directly to the program, but only a mechanism that behaves in the manner described by the program. For example, the behavior of a thermostat may be adequately described by the following program taken from Newell and Simon (1972, pp 31).

Insert Figure 11 about here

The thermostat however, has no interpreter which processes symbolic structures. The thermostat contains only a simple mercury switch that behaves in the manner described by the program. At some level then, an interpreter is just a mechanism which directly accomplishes the actions described by the program. If the interpreter of an IPS is actually a mechanism that simply produces a sequence of behaviors (e.g. mercury switch) then we cannot describe its actual internal structure in information processing terms. Only examination of the microstructure of the system in question will determine the actual mechanisms at work within an IPS, but even in this case, a program will remain an adequate description and predictor of the IPS's behavior.

Problem Solving Information Processing Systems

A problem solving situation consists of two general constructs; a problem solving system and a task environment. A problem solving system is simply any IPS capable of selecting and executing actions in order to achieve specified goals. Although Newell and Simon never speak of a problem solving system in this way, it

is apparent that the problem solving system represents the lower limit of their theory and equates roughly to any physical platform capable of problem solving activity (i.e. selecting and executing actions to achieve goals). As they indicate several times, the physiology or the true underlying structure of the IPS is not important for describing the behavior of the system. The problem solving system, therefore, is left largely undefined and can be any system capable of meeting the generic description of an IPS (human, chimp, computer, thermostat, etc.).

The task environment can be generally defined as an environment coupled with a task, goal, or problem for which a problem solver is adequately motivated to complete. The task environment contains all the information and objects necessary for the problem solver to produce a correct solution, but the task environment may also contain objects and information which may interfere with, disrupt, limit, and/or block some solution paths. It is natural, albeit incorrect, to think of the task environment as being entirely external to the subject. In fact, the inherit capabilities of the subject such as generalized intelligence and level of experience (master v. novice chess players) are also components of the task

environment. The task environment includes, but is not strictly limited to, the goal of the problem, the conditions under which the goal can be obtained, the legal tools and operations that can be used to obtain the goal, the inherit problem solving capacities of the solver, and the starting state of the problem.

A problem exists when a goal is desired by a system that does not possess an immediately available method to obtain the goal. Even though all the tools and information necessary to solve the problem may be immediately available in the task environment of the problem solver, a solution path to the goal may not be forthcoming because the TRUE problem does not exist in the external world, but is created within the problem solving system. To illustrate this point, consider what might happen if two people are presented with an identical problem in an identical external environment. It is quite possible, even likely, that the two individuals will perform the same task in different ways. In fact, a single individual is likely to perform different behavioral sequences during separate exposures to the same problem. If the problems are the same and the external environment of the subject is the same then the performance differences must be attributed to some variation within the problem solving

system. In Newell and Simon's framework, a problem solver is characterized as an IPS which builds an internal representation of the task environment in order to produce a solution to a given problem. The IPS's internal representation contains all the information relevant to the problem including tools, operations, capacities of the problem solver, problem states, goals, and any other useful concepts which the problem solving system may have available to describe the problem situation (e.g. knowledge gained from experience in other task domains).

The internal representation of all the information relevant to the problem is labeled the problem space (see Newell & Simon, 1972; p. 56 - 86). The problem space is not a physical space which can be pointed to, but is instead the essence of the problem which exists within the problem solver's cognitive machinery. Although a problem space can be represented externally with game trees, productions, magic squares, and other symbolic structures, these structures are not the true problem space. In linguistic terms, the game tree and other symbolic structures are analogous to the surface structure of language while the true problem space is analogous to the deep structure of language. The composition of the internal problem space determines

the nature of the TRUE problem, the behaviors that the system will perform in attempting to solve the problem, and whether or not the problem can be solved by a system operating within the given task environment.

Attainment of a goal becomes a problem when the number of plausible routes to the goal is large or immense, the correct solution paths are widely dispersed throughout that huge set of plausible solutions, and the cost of obtaining and testing each possible solution is high. A problem solver therefore constructs an internal representation of all factors deemed pertinent (i.e. the problem space) in order to manage the immense amount of potentially relevant information. The task environment's relevant components (e.g. the goals, legal tools and operations, and the initial state of the problem) are all represented in the problem space. The problem space also provides various imagined intermediate steps towards a solution. The problem space can therefore be used to internally generate and test solution paths (sequences of mental and motor activity which may or may not lead to a goal) without actually having to perform the associated behaviors. The problem space thus represents the set of possible solution paths which are available to the problem solver. Learning is

therefore the process of composing, editing, and retaining an internal problem space which allows the problem solver to most effectively derive the correct solution path for any problem selected from a specified class of problems (e.g., 4*3 is a problem selected from the class of multiplication problems). Accordingly, problem solving is the process of deriving the correct solution path from the internal problem space, and is therefore internal to the problem solver. Consequently, the true problem (i.e. deriving the correct solution path(s) from the internal problem space) and the resultant problem solving behaviors are determined by the structure of the internal problem space. Furthermore, because the problem space is internal to the problem solver and thereby unknowable to an external observer, the true problem and the behavior of the problem solving system are likewise unknowable to an external observer prior to the execution of those behaviors.

We can know, however, the external stimuli and some of the problem solver's intrinsic capacities (i.e. portions of the task environment), and based upon the demands of the task environment, one can fabricate a hypothetical problem space. Task demands are constraints on the behavior of the problem solver which

must be satisfied in order for the goal to be obtained (e.g., you must have three-in-a-row to win in tic-tac-toe, sufficient IQ, level of experience, short-term memory capacity). An analysis of the task environment produces a description of the task demands which, in turn, establish possible solution paths while rendering other solution paths inaccessible to the problem solver. The inaccessible paths are not, therefore, components of the subject's problem space.

For example, a subject may be placed in an environment where he must obtain $2000. The rules of the subject's society, and personal moral code, may provide task demands by disallowing homicide or theft as viable solution paths. The capacities of the problem solver may also provide task demands. If he cannot read or write, or if he has no collateral of value equal to $2000, he may not be able to obtain a loan from an accredited lending institution. Each of these constraints, and many more, may disallow certain solution paths while defining other paths. By carefully analyzing the task environment, which involves identifying capacities of the environment and the problem solver, one can build a hypothetical problem space devoid of irrelevant solution paths and generate a set of prototypic path features which could

lead a problem solver to a correct solution.

A hypothetical problem space is a representation of the behavior demanded by the task environment given a perfectly rational problem solver operating at specified level of adaptivity. A problem solving system is considered perfectly rational if the behavior exhibited by the system in a specified problem solving situation is appropriate given the demands of the task environment. If a task environment demands certain behaviors and a problem solver exhibits those behaviors then those behaviors tell us more about the task environment than about the subject other than he/she is adequately motivated and equipped to perform the task. It is when actual human behavior deviates from the behavior predicted by the perfectly rational model that we begin to discover something about human rationality.

The demands of the task environment and the psychology of the subject, are the components which Newell and Simon regard as the two most important aspects of human problem solving. Therefore, the first step of the general methodology proposed by Newell and Simon is analysis of the task environment in order to determine the task demands. Second, based upon the task demands, one constructs a hypothetical problem space from a specified set of eips and symbol

structures. Subsequently, a perfectly rational problem solving system is created by allowing an IPS to operate with the hypothetical problem space in the specified task environment. Next, one observes humans performing the same task in the same environment and identifies the differences between the behaviors of the humans and the perfectly rational IPS. Lastly, the differences between actual human behavior and the perfectly rational behavior of the IPS are used to develop a model of the human's actual problem space.

As an example, given the following tic-tac-toe board configuration and model of a perfectly rational

---

Insert Figure 12 about here

---

problem solver, X's most rational next move should be in the lower center square. This is the move that is demanded by the task environment to achieve three-in-a-row. It is when behavior deviates from this rational model of behavior that we learn something about the psychology of the subject. If the subject is motivated to win the game and has the cognitive capabilities to achieve that goal, then why would she play in the right hand center square? Perhaps she was looking for two contiguous squares that contained her pieces and when

she did not find that configuration, she employed a rule that stated: if your opponent has marks on two contiguous squares and an open square in line with those two squares then place your mark in the open square. The subject's deviation from the perfectly rational model has given us a clue about her internal representation of the problem and allows us to modify our perfectly rational model.

Insert Figure 13 about here

In the above case, the subject's representation is slightly flawed because it effectively omits the possibility that two non-contiguous pieces can represent a winning move. The study of problem solving need not, and should not, be limited only to those cases where we can build a superior rational model to that of the human subject. In fact, the study of human problem solving is most interesting and beneficial when our scrutiny turns to the task environments of experts who have obtained rare and superior abilities. For example, an investigator may be interested in determining the most successful chess strategies, and a sound investigative method would be to examine the play of a senior chess master. In most cases, an

investigator could not create a hypothetical problem space for chess which is superior to a senior chess master's internal representation. When the term "perfectly rational" is applied to a problem solving system, it refers only to the fact that the system's problem space is a reasonable description of the task environment and task demands, but implies nothing about the efficacy of the problem space. That is, a problem solving system can be perfectly rational without being perfectly successful.

For example, a perfectly rational chess playing computer might operate by examining all possible board configurations following each of the opponent's moves in order to determine which of its available moves has the highest probability of success. The computer may be able to win, but due to the sheer number of possible combinations, a single match might take years to complete. Furthermore, if the computer were to compete in another chess playing domain, namely lightening chess, it would be soundly defeated. The hypothetical problem space and the perfectly rational behavior of an IPS are merely starting points, or straw-man constructs, to which human behavior can be compared. The behavior of the human experts will almost surely deviate from the behaviors predicted by the perfectly

rational models. One can then modify or create new and better models of human problem solving as one increases the understanding of exactly how the experts' behaviors differ from the predicted behaviors. After modifications based upon the experts' performance, the hypothetical problem spaces can be represented in a problem solving grammar (e.g. production systems, game trees) and made available to non-experts in order to increase their understanding of a particular task environment. In fact, the major benefit of Newell & Simon's approach to the study of human problem solving is not that it allows one to build automatons capable of performing tasks which have heretofore been considered uniquely human, but that the knowledge which results from their approach is communicable to other human beings and can therefore be used to enlighten, educate, and improve human performance.

The Problem Solving Process

The essential components of the problem solving situation, the task environment, the IPS, and the internal representation of the problem (problem space), have now been defined, and a general methodology for the study of human problem solving has been sketched. The only remaining preliminary construct that needs to be outlined is the process by which an IPS can actually

solve a problem within a specified task environment.

The process of solving a problem begins with the act of input translation. During this phase of the process, the IPS creates an internal representation of the problem (i.e. the problem space) which may render solutions obvious, obscure, or even unattainable. After the problem is represented internally and based upon the formulation of the internal problem space, the IPS selects a problem solving method. A method can be viewed as a general strategy or a specified sequence of elementary information processes that, when executed, will achieve, or attempt to achieve, a desired goal. Once begun, a method controls both internal and external behaviors of the IPS, but its control is not absolute. That is, a method can be halted and once it is stopped, the system has four options. First, another method may be selected and tried. Second, another internal representation may be selected and the entire problem reformulated. Third, all attempts to solve the problem may be discontinued (i.e. the problem solver gives up), and fourth, the method reaches a successful conclusion.

As stated earlier, the IPS is fundamentally serial in nature. That is, an IPS is capable of selecting and executing only one method at a time. It should be made

clear, however, that at lower operational levels (e.g. perception), an IPS may have parallel capabilities. That is to say, an IPS may recognize many things at once, but responds to them with only a single action at a time. After settling on an internal problem space, the general behavior of a problem solving IPS can be described as iterative: select a goal; select a method; execute the method; evaluate results; select new goal (or subgoal). The behavior of an IPS, however, can also be characterized as recursive. During the course of its operation a method may produce more than one sequence of potentially successful behaviors. Therefore, the system may continue on one branch while retaining a stack of indexes to other pending branches so that control can return to a given point upon failure of the attempted subgoal.

The process of problem solving, as posited by Newell and Simon, is controlled by two constructs; the problem formulation as determined by the structure of the problem space and the methods which, in turn, are selected upon the basis of the problem formulation. The subjects which Newell and Simon studied tended to be well rehearsed in their task domains and therefore, did not often change their internal representations of the problems. Consequently, their original theory

stated little about creating internal representations or shifting from one internal problem space to another. Most of their theoretical work therefore, did not focus on the specifics of the internal representation but focused instead on problem solving methods.

In order for a method to be useful, it must be general enough to be applied to any problem selected from a specified class of problems. Generalizable methods are created by allowing the methods to contain variables which are instantiated with specific relevant information from the current problem space. The problem space, in most cases, represents more information than is required by the method. The problem formulation serves as the interface between the problem space and the methods by designating a specific method and the information contained in the problem space that is to be used by the method. In effect, a problem formulation and its associated method serve to reduce the portion of the problem space which must be considered in order to solve the problem.

For example, a problem solver is given the goal of finding all the words which can be found in the word "xylophone". If the problem is formulated as "find the words which already exist in "xylophone"" then the method labeled read_words(x) may be activated. The

method read_words(x) would instantiate x to "xylophone" and, by means of several elementary information processes (eips), would read the words which exist in xylophone (i.e. lop, hone, phone, one, on). On the other hand, if the problem were formulated as "rearrange the letters of "xylophone" to form all possible letter combinations that are words", then the problem solver may use a different method (e.g. find_all_legal_combinations(x)). Both of the above cases require lexical searches, but the former problem formulation requires fewer lexical searches (9+8+7+6+5+4+3+2+1 or 45 possible combinations versus 9!/2! (total permutations divided by permutations of duplicate letters) or 181,440 possible combinations). Consequently, the former problem formulation and method reduce the problem space by a greater degree and thus, define a problem which is easier to solve.

The problem solving process can now be summarized as follows. A problem solver is placed in a task environment and forms an internal problem space which represents, but is not limited to, the initial state, the goal, and plausible intermediate steps towards that goal. Next, the problem solver formulates the problem based upon the information represented in the problem space (including relevant information gained from other

experiences). The problem formulation and its associated method may reduce the size of the problem space and thereby, the number of elements which need be considered in locating a solution. The problem solver then applies the method to all the elements in the remaining subspace, and if the problem is formulated correctly (i.e. the proper method is applied to the proper subset of elements), then the goal will be attained.

Newell and Simon describe three general types of problem formulations and concomitant methods: recognition, set-predicate/generate-and-test, and heuristic search. Recognition is used in familiar task environments where a solution can be obtained immediately from memory by simply examining the knowledge gained from previous, often well rehearsed problem solving situations (e.g. 9*4=?). In general, problem solving proceeds by reducing the problem into more manageable tasks. Eventually, however, the reductionism must be stopped and sub-tasks performed. Recognition is a special case of problem solving and is important because it is often the method employed in the most elementary sub-tasks. That is, problem solving usually, if not always, proceeds by reducing the problem to tasks for which the problem solver

already possesses answers. Thus, reducing the problem to sub-tasks stops when a solution can be found by simple pattern matching (i.e. recognition).

The second and most general type of problem statement is the set-predicate formulation. In this problem formulation, the problem solver is given a set of elements (E), and the goal of locating an element that has specified properties. As stated earlier, a method can use only that information which is designated by the problem formulation. The set-predicate formulation only provides a method with the elements contained in set E and the properties which an element must possess in order to be judged as a member of the goal set (G).

The method which makes the most obvious use of the information provided by the set-predicate formulation is known as the generate-and-test method. The method operates simply by generating the elements of E and testing each element for the properties required for membership in G. The generate-and-test method and the set-predicate problem formulation are quite general and will always be successful provided that the properties of the goal set are formulated correctly, the set of possible solutions is finite, and the generator is allowed to operate long enough. The amount of time

required to locate the solution(s) will depend upon the amount of time required to generate each element in the problem space, the time required to test each generated element, the size of the problem space, and the relative position of the solution(s) within the problem space.

The third problem formulation type is the heuristic search formulation and is characterized as a search for a path through the problem space which will lead from an initial state to a goal state. In one sense, the set-predicate and heuristic search formulations are quite similar. Each view the problem space as a set of elements which must be tested in order to determine if that element is a member of the goal set. In the set-predicate formulation, however, the generate and test operations are binary and completely independent. That is, each element is simply judged as to whether it belongs to the goal set or not. No judgement is made or retained as to its similarity to the goal set or its adequacy as an intermediate step toward the goal. The power of the heuristic search formulation is that it makes use of information from previous generations and tests, and knowledge gained from other environments, in order to determine which element in the problem space to

generate and test next.

To illustrate the distinction between the set-predicate and heuristic search formulations, consider the problem of determining the combination of a safe (Newell & Simon, 1972 pp. 97-98). Given a safe with 10 dials, each having 100 different settings, the problem space (E) would contain 100^{10} possible combinations. The generate-and-test method would require that each possible combination be generated and tested, thus placing an unrealistic time burden upon the problem solver (assuming each combination could be generated and tested in one second, it would take over 3 trillion years to try all possible combinations). If each dial, however, generated a faint click when its correct setting was selected, then it would not take a very sophisticated heuristic generator to "crack" the safe. In fact, a problem solver using a heuristic search formulation would require, on average, only 500 attempts to arrive at the correct combination. It should be noted, however, that unlike the generate-and-test method, a heuristic search method does not guarantee a correct solution. For example, if the dial clicks were not indicative of the correct solution then the heuristic method described above might never generate the correct combination.

The basic cycle of the heuristic search method begins by selecting an element in the problem space. Next, the element is tested in order to determine whether it represents the solution. If not, the system may apply other operators to the current element which may suggest whether the element is a correct step towards the desired goal, and should thus be remembered for later use, or whether it should be rejected. The last cycle step is a three pronged decision of whether to apply other operators to the current element, advance the search by replacing the current element with a new element, or go back to an untried path. A heuristic search terminates when a solution is found, resulting in reconstruction of the solution path, or the set of untried paths is exhausted. If the set of untried paths is exhausted then a new problem formulation must be created and a different method applied. For any one problem, there may exist several instances of heuristic search formulations each containing different sets of operators that could be applied to the problem space. For example, one may try meaningful number sequences to open the fore mentioned safe (e.g. birthdays, street addresses, or any other personally significant numbers). It should be clear that the heuristic search formulation and method,

described above, represent a class of problem formulations that can be tailored to any particular problem, but it is not the only heuristic search formulation possible. (e.g. Working-backward, working forward, means-ends, planning)

Summary of Newell and Simons' Human Problem Solving Theory

Newell and Simon's (1972) theory of human problem solving was a largely successful attempt to specify a science of adaptive organisms. Adaptive organisms are adaptive because they are flexible enough to modify their internal processing and external behavior in order to fit a variety of task environments. Consequently, adaptive organisms may vary along any of several dimensions. Hence, Newell and Simon chose to restrict the scope of their study of adaptive systems to a subset of human problem solving activities and described the scope of their study as follows (1972, pp. 3-4 & 790):

> The present study is concerned with the performance of intelligent adults in our own culture. The tasks discussed are short (half hour), moderately difficult problems of a symbolic nature. The three main tasks we use- chess, symbolic logic, and algebra-like

> puzzles (called cryptarithmetic puzzles)- typify this class of problems. The study is concerned with the integrated activities that constitute problem solving. It is not centrally concerned with perception, motor skill, or what are called personality variables. The study is concerned primarily with performance, only a little with learning, not at all with development, or differences related to age. Finally, it is concerned with integrated activities, hence de-emphasizes the details of processing on the time scale of elementary reactions (that is, half a second or less). Similarly, long-term integrated activities extending over periods of days or years receive no attention.

Even though Newell and Simon based their theory on a rather restricted domain of study, the theory was, and remains, useful on a much broader scale.

The theory addresses five fundamental assertions which are supported by evidence from the problem solving case studies discussed throughout the book. Newell and Simon's most fundamental assertion is that human problem solvers can be adequately represented as information processing systems. In this view, all

adaptive systems (e.g. humans, monkeys, computers, cats, thermostats) are members of the IPS family and differ from one another only in respect to their memory organization, available elementary processes, and program organization. The theory put forth in the book, however, specifies an IPS in terms appropriate to the study of human behavior. Second, the IPS representation of a human problem solver can be carried to great detail with fidelity for any given person in any specific problem solving situation. This assumption implies that a complete and accurate theory of a particular problem solver in a particular environment can be constructed so that details as specific as memory retrieval processes, memory capacities, perceptual processes, and minute environmental details can be accounted for by using only symbolic information processing constructs. Third, substantial subject differences exist which are not simply parametric variations but involve differences of program structure, method, and content. Similarly, substantial task differences exist which are not simply parametric variations but also involve differences of structure and content. A theory must, however, contain some invariants and the two previous assertions indicate that there are only a few gross

characteristics of a human IPS which are invariant across subjects and tasks. The most notable invariants are the existence of symbol structures, a processor, eips, a short-term memory, a long-term memory, an external memory, goals, the serial nature of processing, and the rate at which eips operate (less than .5 seconds). Lastly, the largest determinant of a subject's behavior is the task environment which includes the intellectual and physical abilities of the problem solver. Thus, in Newell and Simons' view, the internal microstructure of a human IPS (e.g. operational details of sensory processes, perceptual processes, physiologic processes) are largely irrelevant to the study of human problem solving. They do acknowledge, however, that even though the microstructure of the IPS is not a central issue, a truly complete theory of human problem solving must also account for these processes.

The general outline of Newell and Simon's theory of human problem solving begins with the assumption that human problem solvers can be adequately represented as information processing systems (IPS) which have a few, and only a few, gross characteristic that are invariant across tasks and subjects. The nature of the few invariant characteristics allows the

IPS to internally represent the task environment as a problem space and all problem solving takes place in that internal problem space. Furthermore, because the problem space is an internal representation of the task environment, the structure of the task environment determines the structure of the problem space. Likewise, the structure of the problem space determines, via the problem formulation, the possible programs that can be used in a specified task environment. The problem space invokes a problem formulation which in turn determines what methods (sequences of eips) can be utilized to solve the problem at hand. Lastly, the processing of an adaptive IPS can be adequately described by a program.

We cannot know the structure of an individual's true problem space prior to their performance in a task environment. We can, however, hypothesize a reasonable program structure based upon the demands of the task environment. The behaviors which are predicted by this hypothetical problem space can then be modified in light of actual human behavior. The resulting program represents a theory of the actual internal problem space of the individual and can then be used to educate, enlighten, and improve human understanding and performance in a specified problem solving domain.

The Neural Network Perspective

Neural Networks, as the term implies, are general purpose algorithms which possess operational properties that are analogous to the neural functioning of the human brain. That is not to say, however, that neural nets are wired, nor operate, exactly like the brain. Neural Nets are "neural" in so far as they are structured more like the brain than are traditional von Neumann computers. Advocates of neural computing claim to favor a brain-mind metaphor over the traditional computer-mind metaphor. Neural computers, however, are still computers and even the most zealous neural realist must therefore, see current neural models of cognition as neurally inspired rather than veridical models of neural processing. In a very real sense, neural realists are still using a computer-mind metaphor, but the computer portion of the metaphor has simply changed from a von Neumann computer to a fine-grained massively parallel computer. Still, there are some very reasonable and very compelling neurological principles which speak for the development of cognitive models based upon a parallel distributed processing paradigm. According to Rumelhart and McClelland (1986, p. 130-136), some of the neural characteristics which provide the fundamental impetus for parallel

distributed processing models are:

The brain is composed of 10^10 to 10^11 neurons.

This is both a source of computing power and a constraint upon the possible models.

Neurons are slow.

Modern serial computers can operate in nanoseconds whereas, a neural cell operates in milliseconds or 10's of milliseconds. Neurons operate at 10 to the sixth times slower than do modern serial computers. Imagine slowing down modern AI software by a factor of 10^6. Much of human perceptual processing, intuitive reasoning and other processes occur in a few hundred milliseconds. This means that most of these processes must be accomplished in approximately 100 serial steps.

Neurons are very simple processors.

It seems unlikely that neurons compute functions that are much more complicated than a single digital computer instruction. Again, imagine trying to write an interesting program (e.g. one that recognizes visually displayed letters) with only 100 or even 1000 machine instructions. The mechanisms of mind are best

understood as resulting from the cooperative activity of a large number of relatively simple processing units operating in parallel. Neurons communicate by sending activation or inhibition through the synaptic connections. Neurons receive large number of inputs from other neurons and can send outputs to large numbers of other neurons.

Each neuron can receive (fan-in) from 1,000 to 100,000 signals and can likewise send (fan-out) 1,000 to 100,000 signals to other neurons. This means that even if every neuron is connected to only 1000 other neurons, each neuron is no more than four synapses away from any other neuron in the system.
One or a small number of incoming action potentials is rarely enough to cause an individual neuron to output an action potential.

This suggests that human computation does not involve the kind of logic circuits out of which we make digital computers but, cognitive processing is the result of the cooperative action of many somewhat independent processing units.

Graceful degradation

There seems to be no single neuron whose loss contributes significantly to the overall performance of the brain. In fact, large numbers of neurons can be lost without appreciable differences in processing capabilities. Even in cases where there is sufficient damage to cause a performance deficient, there exists enough redundancy in the brain to allow the system to recover and achieve performance comparable to pre-injury performance. Such capacities are natural characterıstics ınherent in Neural Nets.

Relaxation is the dominate role of computation.

Computation in the brain is best understood as an iterative process in which the brain seeks to satisfy a large number of weak constraints. Neurons should not be thought of as wires in an logic circuit but should be seen as units which serve as constraints for one another. The brain is seen more as settling on a solution as opposed to calculating a solution.

Learning involves modifying neural

connections.

> Knowledge is assumed NOT to be explicitly stored in given physical location, but is represented by the connections among units. Therefore, the attainment of any new knowledge requires modification of the existing connections between units.

Three primary assumptions arise from these fundamental neurological characteristics. First, human cognition arises from the interaction of a large set of simple processing elements rather than the state of any single component of the system. Second, the simple processing elements function by mutually constraining one another and thus contribute in their own way to the overall performance of the system. The essential character of mental processes is thus viewed as a constraint satisfaction procedure where a very large number of constraints act simultaneously to produce a behavior rather than select a behavior from a pool of stored procedures. Lastly, all knowledge is stored in the connections between processing units. Only very short term storage can occur in the individual state of the processing elements and all long term storage is a result of the connections among the units. In other words, knowledge is implicit in the structure of the

processing system rather than explicitly stored in any particular processing element.

These three fundamental assumptions intimate that there are two primary elements of any neural model; simple processing elements and the interconnections between the processing elements. Analogously to the human brain, neural computers have no central processor or central memory, but are instead composed of many, highly interconnected neurodes or nodes. Each neurode (or node) consists of a simple processor and a small amount of dedicated memory. Each neurode's memory holds an array of values which represent the strength of each incoming signal and an array of values which represent the relative importance (weight) of each incoming signal. Furthermore, like neurons, no neurode has access to the specific contents of any other neurode's memory and must communicate with one another by outputting signals which are indicative of their individual activation level. A neurode can therefore be characterized as a matrix operator which does no more than compute a weighted sum of the incoming signals and outputs a value based upon that weighted sum. Thus, the general behavior of any neurode can be described by the following equation:

$$O_j = f(\Sigma(W_i * S_i))$$

where O_j is the output of neurode j. W_i is the relative importance (weight) of the signal being received from node i. S_i is the strength of the signal being received from node i. f is a nonlinear function which compares the weighted sum to a preset threshold and determines the actual output of the node based upon that comparison.

For example, if a neural net was composed of three layers, and the first layer contained 25 neurodes, then each neurode on the second layer would receive one signal from each first layer neurode (25 signals) and store them in an array. Furthermore, each neurode on the second level would contain 25 connection strengths (one per incoming signal) in a second array. Each neurode would then multiply each value in the signal array by the appropriate value from the connection strength array. Next, the neurode would obtain a grand sum of the products of it's calculations. Lastly, the weighted sum would be passed through a function which compares the sum to some preset threshold and generates an output based upon that comparison. Each layer of the neural net would operate in parallel, thereby improving overall system performance by spreading the

huge number of calculations over many processors. After an input pattern is applied to the system, the neural net will generate the proper output based solely upon these simple calculations and the pattern of connections between neurodes. It should be made clear that a neural network does not execute a series of instructions as does a von Neumann machine, and information is not stored in a specific memory location. Instead, as it may be in the human neural system, knowledge is represented by the pattern of neural connectivity and the overall state of the system after it has settled into a temporary equilibrium condition.

At this point, it is necessary to distinguish neural nets from the underlying hardware upon which they may actually operate. As stated in the introduction, one of the reasons for the resurgence in neural theories is due to recent of advances in computer technology namely, fine-grained, massively parallel or connectionist architectures (Hillis, 1986). Fine-grained massively-parallel architectures are computers that, like neural networks, have many small processors, each of which have a small dedicated array of memory. Most often, the processing elements (processors plus their memory arrays) of fine-grained

massively parallel machines are arranged in a two- or three-dimensional array so that each processor is connected to it's four surrounding neighbors (the so called North-South-East-West connection scheme). In some machines which use a three-dimensional array, the processors may also be connected to the processor immediately above and below them. Fine-grained massively parallel machines, however, are NOT neural networks.

Neural networks are defined by the pattern of interconnection between neurodes, the rules that determine whether or not a neurode will fire (transfer function), and the rules governing changes in the relative importance of individual connections among processing elements (learning rules). Fine-grained massively parallel architectures provide a general purpose hardware platform upon which neural nets can operate efficiently, but the defining characteristics of a specific neural network are usually soft-coded. It would be impractical to rewire a fine-grained massively parallel computer every time one wished to change the interconnection scheme of a neural network. In most cases, it is the software (netware) which defines the neural network and the architecture of the neural net is independent of the underlying hardware

architecture. It is very possible to simulate neural networks on von Neumann machines or even with pencil and paper but, speed may become a serious limitation in both cases. For the purposes of this paper, neural net(works)s, neural computing, and parallel distributed processing (PDP) will refer to information processing models that are controlled by one or more general purpose algorithms which define the interconnection schemes between neurodes, the transfer function, and the learning rule that allows the system to learn, correctly classify, and properly respond to inputs without the benefit of predefined, explicitly coded, task knowledge. Furthermore, the present discussion will focus on the netware details and not the particulars of the hardware upon which the netware is implemented. This means that the algorithms described in this paper can be implemented on any system capable of handling matrices including fine-grained massively parallel machines, serial computers, and pencil and paper (although the latter two's appropriateness may be questioned on the basis of time constraints).

The Development of a Neural Network

The development of a neural net can be characterized by four stages; engineering, training, testing, and operation. In the engineering stage of

development, the overall design of the neural network is determined. That is, the pattern of the connectedness among neurodes is defined, each neurode's set of connection weights are initialized, the transfer function is set, and a learning rule is selected.

During the second phase of development, the neural network is trained with inputs which are representative of separate conceptual categories. Neural nets do not arrive at solutions by locating and executing an explicit set of task instructions, but are, instead, pre-programmed only with very general computational algorithms (transfer function & learning rule). Neural Nets must therefore, generate their own, internal set of transformations by learning through trial and error. The pre-programmed computational algorithms are general in that the same neural net which can classify pixel patterns as numerals could also learn to classify any concept (e.g. alphabetic characters) which could be represented in the same pixel grid. In order for the neural network to recognize the new patterns, the neural net would simply require additional training and neither the programming or design of the neural net would be changed.

After a neural net is trained, it begins the testing stage. A neural net is most often tested with

the training patterns, as well as, new patterns to which it has never been exposed. One of the powers of neural nets is that they are capable of automatic generalization. That is, they do not need previous experience with a particular input or a huge memory of possible feature combinations in order to classify a particular input. Sequential pattern matching algorithms must often rely on explicit descriptions of the features and relationships between features in order to recognize a particular input. Neural nets operate by using the constraints inherent in the input and the connection weights, which are determined during training, to converge on a particular representation. If the input does not allow the system to converge on any of its known classes then the neural net will respond by creating a new class or responding "other".

Lastly, if a neural net performs adequately during the testing phase it can be put into operation as a classifier, associative memory, or whatever task the system was designed to perform. In most cases, if the neural network does not perform adequately in the testing phase then it is simply given additional training. In some cases, however, the learning rules and/or transfer functions are altered and the system is completely retrained.

Two Representative Neural Networks

Neural nets are distinguishable from one another upon the basis of three primary characteristics; the architecture of the interconnections among neurodes, the transfer function, and the learning rule. In this section, I will expand the discussion of these three defining aspects by describing the development of two representative and distinct neural nets; the Hopfield net and the Multi-layered perceptron.

The Hopfield net is named for its creator, John Hopfield who has been instrumental in revitalizing neural net research during the 1980's (Hopfield 1982, 1984, 1986). Hopfield (1982) is generally credited with correctly characterizing neural net behavior as a process of successive approximations in which the difference between the current system state and the desired system state is reduced. The technique is known as gradient descent and can be viewed as a type of "hill climbing" search heuristic.

The now famous perceptron is one of the first systems ever designed which can be classified as a neural net (Rosenblatt, 1959; 1962). The early perceptrons consisted of a single layer of neurodes and were incapable of computing certain functions. Minsky and Papert (1969) wrote an elegant analysis of the

single-layered perceptron which exposed its computational weaknesses. The persuasiveness of the Minsky and Papert argument coupled with the early success of the serial symbolic processing approaches in artificial intelligence all but killed neural computing. The majority of Minsky and Papert's criticisms, however, applied only to the simple single-layered perceptron models. Several researchers have now shown that the most severe enervations of the single-layered perceptron can be overcome by multi-layered perceptrons (Rumelhart & McClelland, 1986a; chapters 5,7,8). Furthermore, many of the neural nets being used today are modified versions of multi-layered perceptrons (Brown, Garber, & Venable, 1988; Caudill 1988, Jones & Hoskins, 1987; Kinoshita & Palevsky).

The task of both networks will be to correctly classify inputs which represent the numerals 0 through 9. For the current task, the numerals 0 - 9 will be represented by 10 separate 3 x 5 pixel grids. Each grid will be divided into 3 columns by 5 rows of pixels. Furthermore, each pixel can have one of two values; 1 (ON) or 0 (OFF). Each of the 10 numerals can therefore be represented by a unique pattern of activated pixels within the 3 x 5 grid. Following the training, the neural nets will be tested with both the

training patterns and a set of novel patterns. The novel patterns will be degraded versions of the training patterns and will be generated by changing the value of three randomly chosen pixels in each training grid. (For a more detailed account of a multi-layered perceptron which recognizes alphabetic characters, see Brown, Garber, & Venable 1988).

Phase I: Engineering

The Multi-layered Perceptron. Multi-layered perceptrons are described as fully-connected feed-forward networks which contain one or more layers of neurodes between the input layer and the output layer. In a fully-connected network, all the neurodes on one level are connected to all the neurodes on the next level. Neurodes on the same level, however, are not connected to one another. Feed-forward networks are simply networks which pass the output from one layer of neurodes to the next higher layer. In contrast, a feedback network would send the output of a layer "back" to the next lower level. Multi-layered perceptrons have one or more hidden layers of neurodes between the input and output neurode layers. It has been shown that a perceptron with two hidden layers can produce arbitrarily complex decision regions (Lippmann, 1987) but, for purposes of simplicity, the network

described here will contain only three layers of nodes (an input layer, an output layer, and one hidden layer). There will be 15 neurodes on the input layer so that each neurode on the input layer corresponds to one of the pixels in the 3 x 5 pixel grid. The hidden layer will consist of 10 neurodes based upon a rule of thumb that the hidden layer should contain 2/3 as many neurodes as the input or output layer which ever is largest (Similar heuristics exist for several types of neural nets (Lippmann, 1987)). The third layer, will contain 11 neurodes, one for each numeral plus one neurode to signify "other".

After training is complete, the neurodes will function by computing a weighted sum of all their individual inputs and then passing that input through a nonlinear sigmoid transfer function. When a pattern is presented to the network, each neurode on the input layer will output either a 1 (fires) or a 0 (doesn't fire) depending upon whether or not its corresponding pixel is on or off. The hidden neurodes and output neurodes can be viewed as matrixes of connection strengths. For the current example, a particular node (k) on the hidden layer (j) would contain 15 weights (connection strengths), one weight for each input node. Initially, the weight matrixes for the output nodes and

the hidden nodes are seeded with small (<.1) random values. The node ($O_{k,j}$) then multiplies the output of a node (i) on the previous layer (j-1) by the weight ($W_{i,k}$). Next, a bias term (ß), which roughly determines the size of steps taken toward the solution, is subtracted from the product of the weight ($W_{i,k}$) and the node ($O_{i,j-1}$) output. The neurode then sums the results for the i (# of nodes on the previous layer) calculations. Lastly, the sums for each neurode are passed through a non-linear sigmoid function ($f(x)=1/(1 + e^{-x})$. In actual practice however, the sums are often passed through a series of conditionals which mimic the nonlinear sigmoid function. For example,

$$f(x) = \begin{cases} .999, & \text{if } x >= +5.0; \\ .001, & \text{if } x <= -5.0; \\ (x+5)/9, & \text{if } +1.0 <= x < +5.0; \\ (x+3)/9, & \text{if } -5.0 < x <= -1.0; \\ (x+2)/3, & \text{otherwise;} \end{cases}$$

Thus, the output of any node (k) on level (j) can be given by the equation:

$$O_{k,j} = f\left(\sum_{i=1}^{n}(W_{i,k} * O_{i,j-1}) - ß\right)$$

where $O_{k,j}$ is the output of node k on level j. $W_{i,k}$ is the relative importance (weight) of the connection

between node k on level j and node i on level j-1. $O_{i,j-1}$ is the value output by node i on level j-1. ß is a predefined bias term. For the hidden layer, n=15, and for the output layer, n=10. For the hidden layer, the maximum value of k is 10, and for the output layer, the maximum value of k is 11.

The hidden layer nodes send their values to the output layer. The output nodes then perform the identical calculations as the hidden nodes but, do not have anywhere to pass their values. Instead, only the neurode with the highest value fires and that neurode should signify the appropriate class for the given input.

The Hopfield Net. The Hopfield net is a single layered network in which all neurodes are connected to all other neurodes. The minimum number of neurodes in a Hopfield network is equal to the number of classes divided by 0.15. For the current task, there are 10 classes (the numerals 0-9) and 10/.15 = 66.67. The number of neurodes is always rounded up. Thus, the minimum number of neurodes needed by a Hopfield net to solve the current problem is 67. The Hopfield net can be viewed as a grid of fully-connected binary nodes which have a direct one-to-one correspondence to the pixels of the input pattern. In other words, the number

of pixels in the input patterns should equal the number of neurodes in the Hopfield net. This means that the resolution of the input patterns should be increased from 15 because we need at least 67 neurodes to recognize 10 classes. For this example, the number of nodes will be increased to 70 in order to make a uniform two-dimensional pixel grid. Each of the 10 numeral classes will thus be represented in a 7 x 10 pixel grid and the network will likewise consist of 70 neurodes.

Following training, the neurodes of the Hopfield net, like the perceptron, will compute a weighted sum of their incoming signals and then pass that weighted sum through a nonlinear transfer function. The transfer function used with the Hopfield net, however, is usually a hard-limiting nonlinearity instead of a sigmoid nonlinearity. That is, the output of any neurode in the Hopfield net will be one of two values as opposed to the perceptron where output values can be continuous and graded. Also, unlike the perceptron, the Hopfield net is a single-layered network and must therefore go through several iterations in order to settle on the proper output pattern. In this sense, the Hopfield net is a feed-back network, because the output of each neurode is sent back into the system as

input for the next iteration. Initially, each neurode in a Hopfield net corresponds to one of the pixels in the input pattern. Each neurode, therefore is either on (+1) or off (-1) depending upon the value of its corresponding pixel. The Hopfield net begins its processing by having each neurode inform all the other neurodes of its initial state. Next, each neurode computes a weighted sum of all the incoming signals and passes the weighted sum through the hard-limiting nonlinearity. The transfer function of the Hopfield net is given by the equation:

$$O_{j(t+1)} = f\left(\sum_{i=0}^{n-1}(W_{i,j} * O_{i(t)})\right)$$

where O_j is the output of node j and t is the current iteration. $W_{i,j}$ is the weight given node i by node j. O_i is the output of node i and n=70.

The neurodes transmit the new value generated by the transfer function to all other neurodes and each neurode then computes a new output. The process continues to iterate until the network converges on a stable output pattern. The Hopfield net is said to have converged on a solution when any two contiguous

iterations generate the same output pattern. Upon convergence, the combined output of the neurodes should represent the pixel pattern of the input pattern's class exemplar.

Phase II: Training

Multi-Layered Perceptron. Perceptrons are referred to as adaptive neural nets because they adapt to inputs via learning rules. The most common learning rules (LMS, Delta, Generalized Delta, and Back Propagation) use a gradient search method in order to reduce the mean squared difference between the current overall output of the network and the desired output of the network.

A multi-layered perceptron is trained by presenting a series of class exemplars to the net. The net then performs the fundamental computations described by the transfer equation. In the training phase, however, a supervisor monitors the output of the system and notifies the system of what the correct output should be. For example, if the input to our perceptron is a "3" then the fourth node on the output layer should be .999 and all the other nodes on the output layer should be close to .001. If the output of node three is not close to .999 and the output of all other nodes is not close to .001 then the supervisor indicates the proper

values to the network. The nodes on the output layer then calculate an error term based upon the mean squared difference between their actual and desired outputs. The nodes on the output layer then adjust the connection strengths (weights) which are held in their local memories and are representative of the connections between the output layer and the hidden layer. Using the same transfer functions described in the previous section, the nodes on the output layer calculate a new value based upon the new weights and propagate the new values back to the nodes on the hidden layer. The nodes on the hidden layer then calculate an error term based upon the difference between their actual output and the new values from the output layer. Based upon the error term, each node on the hidden layer adjusts the connection strength between the hidden layer and the input layer. (For a more mathematically detailed discussion of learning rules, see Caudill 1988; Jones & Hoskins, 1987; Lippmann, 1987; Rumelhart & McClelland, 1986)

All these operations, from the fundamental computations through execution of the learning rule, represent one training iteration. A perceptron is usually given many (100+) exposures to class exemplars before it reaches an acceptable level of performance.

If the system never reaches a desired asymptote or reaches asymptote too slowly then features of the network may need to be changed. For example, the learning rule contains a bias term which determines the size of the steps taken when locating the minimum mean squared error. If the bias is too small, it may take too long to find the ideal set of weights or it may get caught in a local minimum during the search. Conversely, if the bias is too large, the weight vector may begin to jump around excessively as the system approaches the ideal minimum which will significantly slow the settling process. After the necessary "tweaking" has been performed and the system reaches an acceptable performance level with the training patterns, the weights are "frozen" and the system is ready to be tested with novel patterns.

The Hopfield Net. Some types of neural nets, such as the Hopfield net require training periods but do not require supervision and do not learn adaptively. Neural Nets which are not capable of adaptive learning require fixed weights which are calculated during single exposures to class exemplars. Furthermore, they do not generate any output during training and therefore, do not receive any tutoring from an outside monitor.

Static neural nets such as the Hopfield net are

trained using class exemplars (for the current example, the non-degraded numerals 0 - 9 which are represented in a 7 x 10 pixel grid). Each numeral is presented to the network once. After each numeral exemplar is presented, each neurode adjusts its matrix of connection weights based upon the pattern of inputs. The weights are adjusted via the following equation:

$$W_{i,j} = \sum_{i=1}^{n}(X_{i,s} * X_{j,s}), \quad i <> j$$

where $W_{i,j}$ is the connection strength between node i and node j. $X_{i,s}$ can be +1 or -1 and represents the value of the ith element of the exemplar for class s. Likewise, $X_{j,s}$ is the jth element of the exemplar for class s.

It should be pointed out that in this equation, elements and neurodes are functionally equivalent because each neurode corresponds to one and only one element/pixel of the exemplar. After all exemplars have been presented once, the system is ready to be tested with unknown inputs.

Phases III and IV: Testing and Operation

The Multi-layered Perceptron. Testing the multi-layered perceptron is a straight forward procedure in

which novel stimuli are presented to the system. In this case, the system will be tested with a mixed set of patterns consisting of the normal exemplar training patterns and exemplar patterns which were degraded by random noise. Testing patterns, however, can be different versions of the exemplar pattern. For example, the training exemplar for the number "3" might be represented by turning on all three pixels of the top and bottom rows, the five pixels in the third column, and the pixel in the third row of the second column. Serifs can be added to the testing pattern for the numeral "3" by turning on the second and fourth pixels in the first column. From the neural net's perspective, classifying a variation of an exemplar is the same as classifying a degraded version of an exemplar. In any case, the testing patterns are simply presented to the system and the system responds, hopefully, by firing the correct output neurode. If the system passes the testing phase then it can be put into operation. If it does not perform satisfactorily, then it will be returned to the training phase where its performance can be optimized.

The Hopfield Network. The testing and operation phases for the Hopfield net are one in the same because there is no way to "tweak" a Hopfield net. If the

system does not perform satisfactorily once its weights have been set then there is no other option but to generate a new set of training materials and completely retrain the system with the new set of exemplars. The actual steps in testing/operating a Hopfield net also differ from the multi-layered perceptron in terms of its output format. As with the perceptron, a set of materials is selected and presented to the network. The Hopfield net however, cannot fire an individual node which represents the proper category of the input. Instead, the pattern of all the neurodal outputs depicts the proper class exemplar. For example, the input might be a degraded "3". The system would iterate until it had settled on pattern of activated neurodes which corresponded to the exemplar pattern for the numeral "3".

The advantage to non-adaptive neural nets is that they require very few training trials, but they have several drawbacks. First, the number of patterns which can be recognized by such a net is limited and if that limit is surpassed the system will settle on novel spurious patterns different from all exemplar patterns. Second, the number of neurodes and calculations needed to recognize even a small number of classes can be quite large. For example, the Hopfield net can only

recognize a maximum of .15 times as many classes (C) as there are nodes (N) in the system (C = .15N or N = 6.67C). In other words, a Hopfield net capable of recognizing all the letters of the alphabet (24 classes) would require a minimum of 160 nodes and 25,600 connection weights! Lastly, an exemplar pattern will be unstable if it shares too many pixels in common with another exemplar. For example, a degraded "8" may be classified as a "3" by the system unless certain orthogonalization procedures are followed which may further increase the number of neurodes, weights, and calculations needed to solve even relatively simple classification problems.

Conclusion

Neural net research began nearly forty years ago, but due to a lack of technology, sufficient mathematical techniques and the early successes of sequential symbolic processing efforts, the field remained virtually dormant until recently. Neural networks are neurally inspired models of processing which are based loosely upon known fundamental operating characteristics of the human brain. Like the human brain, neural networks consist of many highly interconnected, simple processing units which take a weighted sum of their inputs and transmit an output

based upon that sum. The result of neural net processing is not a particular value stored at a specific memory address but is, instead, represented by the overall state of the system after it has converged on some equilibrium condition.

The development of a neural net involves the four stages of engineering, training, testing, and operation. Many different types of neural nets exist and the structure of a particular neural net is defined by three characteristics; 1) the pattern of interconnection between the processing elements, 2) the rules that determine whether or not a processing element will fire (transfer function), and 3) the rules governing changes in the relative importance of individual connections to a processing element's output (training/learning rules). The multi-layered perceptron is a type of neural net which is characterized as a fully-connected, feed-forward network which requires supervised training. The Hopfield network, on the other, hand is a single layered, feed-back network which does not require supervision. Even though the multi-layered perceptron and the Hopfield net have very different structures and operating characteristics, they can solve equivalent types of problems by virtue of their highly parallel,

fine grained architectures.

Neural networks are computing techniques which may provide solutions to problems that traditional computer systems have, thus far, failed to solve efficiently. Neural networks seem to solve problems that humans can solve easily, even unconsciously. Problems such as identifying an entity given only a partial or degraded description, recognizing faces, recognizing continuous speech and any other task which would be solved most effectively with an associative, content addressable memory. Neural nets do not work by executing a specific set of explicit, sequential steps. Instead, they are trained, learn, self-organize and settle on solutions. All of the problems mentioned above are important problems for the computing and psychological sciences, but do neural models have anything to say about the nature of higher level human problem solving capacities which have been competently portrayed by traditional serial information processing models? In the next chapter, I will discuss the relationship between neural nets and serial problem solving models, how neural nets can be built to solve higher level problems, and what the advent of neural nets may mean for the future of human problem solving research.

Problem Solving Neural Nets

In the previous section I presented an overview of Neural Networks and described how two different types of neural nets could be employed to perform a classification task typical of the class of problems to which neural nets are most readily applied. In the remaining chapters however, I wish to focus on the application of neural net models to higher level cognitive processes. To begin the discussion, and in order to lay some ground work, I will describe the correspondence between neural networks and schemata. (Most of this ground work is essentially a summary of Chapter 14 in McClelland and Rumelhart (1986).)

Whereas the previous chapter drew strongly on the work of many applied researchers in order to provide concrete examples of neural network fundamentals, the work of Rumelhart and McClelland (1986a; 1986b) is a much more theoretical endeavor. Their general goal was to explicate how human cognitive processing can be characterized by coalitions of highly interconnected processing elements that operate in parallel. They did not attempt to specify one type of neural computer which could account for most human thought processes but instead described a class of computing architectures which they refer to as parallel

distributed processing (PDP) systems. In their two volumes, Rumelhart and McClelland (1986a; 1986b) describe a variety of human cognitive and behavioral phenomenon and then fit one or more PDP systems to a given phenomenon in order to demonstrate how PDP systems can account for the specified phenomenon. Some of the PDP systems which they describe have operating characteristics which resemble perceptrons, Hopfield nets, Hamming nets or other specific neural network architectures, but all the systems described by Rumelhart and McClelland have the features commonly associated with neural networks such as high inter-connectivity between processing elements, no explicitly coded instructions, all knowledge available to the system is represented by the weights of the connections between processing elements, and a high degree of parallel operation of processing units. (Rumelhart & McClelland's use of the term "units" is equivalent to the concepts of neurodes, nodes, and processing elements.)

It should be noted that in the PDP framework units can vary in their level of abstraction. That is, units can represent low-level features such as points, lines, arcs, pixels, and edges, or they can represent concepts as complex as voltage, words, phonemes, resistance,

furniture or any other concept relevant to performing the task at hand. Thus, the unit becomes Rumelhart and McClellands' theoretical primitive and the level of explanation at which any PDP model operates is defined by the degree of abstraction employed at the unit level.

The construction of a problem solving PDP model begins by specifying a set of primitives sometimes called knowledge atoms. These primitives are essentially mapped onto the processing units and the weights between the units are set based upon the "real world" relationships among the knowledge atoms. For example, if one were to construct a PDP model capable of classifying rooms based upon the contents of the room, one might select "furnishings" as the level of the primitives (see Rumelhart & McClelland, 1986, p. 22-32, for a complete description of such a model). The unit representing "lounge chair" might give strong weight to the signal from the unit representing "couch" because lounge chairs and couches are often found in the same room (e.g., the living room). Thus, if a lounge chair is present the couch unit obtains a higher state of activation. Likewise, the units representing foot stool, fire-place, and television might also achieve a higher state of activation because they also

often co-occur with lounge chairs in living rooms. A unit representing "kitchen-sink" would have a high negative connection strength with the "lounge-chair" unit because a kitchen-sink rarely occurs in the same room as a lounge chair. Items such as carpet, windows, and drapes would probably have a positive connection strength with the "lounge chair" unit but, because they are likely to occur in several rooms (e.g. bedroom, den, family room, office) the connection strength would be weak.

Essentially, each unit in a PDP model represents a hypothesis about the presence or absence of the concept to which the unit corresponds, and the connection strengths represent constraints among the hypotheses. The processing of a PDP system can therefore be characterized as a process of constraint satisfaction. Thus, if feature B (e.g. couch) is expected to be present when feature A (e.g. lounge-chair) is present then there should be a positive connection between the unit representing the hypothesis that feature A is present and the unit representing the hypothesis that feature B is present. Likewise, if B never occurs with A then there should be a negative connection between units representing hypotheses about A and B. Furthermore, the strength of the connection

should reflect the degree to which the presence of A is a predictor of the presence of B. If B is very often, or very rarely, present when A is present, then the magnitude of the connection strength should be large. If the occurrence of A does not consistently predict the occurrence of B then the weight should have a small magnitude. Inputs to the system also provide constraints to the system. A positive input indicates that there is evidence from outside the system which supports the hypothesis that a particular unit is present and the value of the input signifies the strength of the evidence. Likewise, a high negative input provides strong evidence from outside the system that a particular unit is not present.

If a system, designed in this way, was allowed to run then some units would gain enough evidence for their existence and would fire. After previously inactive units fire, the system possesses a new activation pattern which represents new evidence to the units as to the likelihood of their presence. This new evidence is equivalent to a new set of constraints and is reflected by the overall state of the system (i.e. the activation levels of all the units). On the next processing cycle, the new set of constraints would cause other units to fire, or not, which in turn

creates a new group of constraints with which the system must deal. Eventually, however, the system will settle or "relax" into a state which optimally satisfies as many of the constraints as possible with priority given to the strongest constraints. That is, the system will ultimately reach a point where the satisfaction of all impinging constraints cannot be improved, and, therefore, the activation pattern of the units no longer changes.

In the human nervous system it is very unlikely that individual neurons account for abstract concepts such as couch or sofa. Nor do PDP theorists even remotely suggest such to be the case. In an ideal PDP environment, concepts such as sofas would be defined by other networks which use concepts such as padding, cushions, length, width, height, and upholstery as the level of abstraction. Likewise, these units would represent instantiations of sub-nets which define the elemental concepts (padding, cushions, etc), and so on, until some basic level of interpretation such as perceptual features (edges, points, etc.) is reached. Thus, when activated by input from the external world, the activation patterns of the units representing low-level features will influence what is being interpreted at higher processing levels (bottom-up processing), and

the activation patterns at higher processing levels would, in turn, influence the activation patterns of the lower-level units (top-down processing). When the system finally relaxes (i.e. the constrains are optimally satisfied) an interpretation of the scene has been instantiated and is reflected by the overall pattern of activation across all the units. Therefore, a fundamental assumption inherent to PDP theory is that processing on any level is best characterized by coalitions of processing units whose microstructure is highly parallel, but whose collective actions may be viewed as sequential processes.

While the specific operational details of the above processing scenario are somewhat new, the general data structure which emerges from this type of architecture has been bandied about by cognitive researchers for years and goes by many names (e.g. schema, scripts, and frames). The PDP group tends to use schema as the label for the data structure that emerges from constraint satisfaction networks. The concept of schema can be generally defined as a conceptual structure which represents generalized knowledge about objects, situations, and events. Schemata are sometimes difficult structures to implement because while they are generic

representations of the world, they must also have the capacity to represent specific instantiations of objects, situations, and events.

One characteristic which allows schemata to represent both generic and specific information is that schemata contain variables (sometimes referred to as slots). Schemata can be viewed as a group of characteristics that tend to co-occur with one another. If only partial information is available then a schema is capable of filling in the values of the empty slots based upon the values of the variables which have already been provided.

A second important characteristic of schemata is that they can be embedded. That is, a schema, much like an idealized neural system, is a layered structure in which values in one schema are defined by sub-schemata. For example, one may partially define a schema for a lounge chair by specifying slots such as padding, covering, and position-range with the values of 10 (high), leather, and upright-supine, respectively. The schema for lounge chair however does not exist in a vacuum, but resides in a structure which includes schematic representations for the more specific concepts of padding and covering. Likewise, the schema for lounge-chair may also be used by another

schema which defines a living room. In this case, the lounge chair would in effect become a slot in the living-room schema and would have the value of present. This would in turn activate all the slots of the lounge chair (padding, covering, etc.), which, in turn, would provide activation to the slots within the padding and covering schemas, and so on.

This brings up the third important characteristic: schema represent knowledge at all levels. Whether a structure is a schema, a sub-schema, or a super-schema depends solely upon the viewer's perspective. If the perspective is set at the lounge chair then the living-room is a super-schema and padding is a sub-schema. However, if the perspective is the living room then the lounge chair becomes a sub-schema and padding becomes a sub-sub-schema. There is nothing inherent in the structure of a schema that differentiates their general operating characteristics. In this sense, it is probably best to characterize schemata as a network rather than a hierarchy or tree structure.

The last defining characteristic of a schema is that it should not be viewed as a static structure which is stored at some particular location. A schema is an active structure which seeks to maximize the agreement between data input to the system and data

from previous experience which currently resides in the system as weights. In other words, schema are generative, self-organizing knowledge structures. Most symbolic implementations of schema (e.g. frames and scripts) are essentially static formalism which do not fully capture the generative, self-organizing nature of the theoretical schema. Neural networks do, however, provide the potential for implementing a generative, self-organizing data structure which captures many of the characteristics of the schema ideal.

For illustrative purposes, consider the room classifier described in Rumelhart and McClelland (1986, Ch. 14). They asked subjects to imagine an office and then presented them with a list of 40 descriptors (desk-chair, ceiling, oven, telephone, drapes, etc) and asked whether each descriptor was accurate for an office. Using the same list of descriptors, they asked subjects to repeat the task when imagining a living room, a kitchen, a bathroom, and a bedroom. Each descriptor was represented by one unit in the network. The data obtained from the subjects' judgements were plugged into an equation that determined the weights for each unit based upon the probability of the presence of that unit predicting the presence of any other unit. For example, if the lounge-chair

descriptor was a highly, positive predictor of the couch descriptor then the unit representing the "couch" assigned a high positive value to the input from the "lounge-chair unit". Thus, if the "couch" unit received a strong positive value from the "lounge-chair" unit, then the "couch" unit's activation level would increase.

In the single-layered network described by Rumelhart and McClelland, each unit represented one of the 40 descriptors and each run began by "clamping-on" one of the descriptors (setting its activation value to 1 and never letting it change). Thus, if the weights have been properly set, and oven was clamped on, then one would expect the system to settle on a pattern of activation that included refrigerator, coffee-pot, sink, stove, and toaster having a value of 1 and units representing the descriptors of sofa, bed, toilet, and desk of having a value of 0. In essence, the system settles on a pattern of activation that corresponds to a schematic representation of a kitchen. That is, the only units that will be activated will be the units that represent items commonly found in the kitchen. Likewise, if one clamps on bathtub, the system will settle on a activation pattern that represents a bathroom. In schema terminology, one could say that

given a certain input, the network is able to generate and fill in the slots of the room schema.

It is important to realize that the ability of the system to settle on a given structure is in part determined by the definitional power of the clamped descriptor(s). Bathtub is a highly definitional descriptor of bathrooms, thus it has a high positive weighting for other bathroom descriptors and high negative weightings for non-bathroom descriptors. However, if one clamps on windows, for example, the system may settle on a less coherent set of active descriptors, because windows can occur in almost any room and will thus have moderate or low weights with nearly all other descriptors. Further, it should be made clear that schema are represented by the overall pattern of activation in a neural network and not in the state of any one unit. In a complete neural system the presence of any concept would be defined by the activation pattern of other networks of units. Thus, a complete neural system would consist of networks of networks much like a complete schematic representation would require networks of schemas. For the sake of simplifying the current case, however, the possible activation patterns of the descriptor networks have been collapsed and are represented by the units.

Although Rumelhart and McClelland used a single-layered, fully-connected neural architecture to illustrate neural based schemas, it may also be possible to implement such a system on a multi-layered perceptron. In this case, the descriptors would be mapped onto the units of the input layer and the room types (or descriptors) would be mapped onto the output layer. The behavior of the network would not be as observable as in the case of the single-layered net because most of the processing would take place in the hidden layer(s) of nodes. Also, due to the processing characteristics of the hidden layers, one cannot easily pre-define the connection strengths between all layers. In the perceptron case, the network may use the data obtained from subjects as training and supervisory materials. Lastly, instead of a pattern of activated units being displayed when the system settles on a maxima, perhaps only one node, which represents the class/schema, might be activated on the output layer. While the multi-layer perceptron probably has less illustrative power for describing neural based schema, it remains an important processing model for many researchers.

Synthesis: Integrating Neural Network and Traditional Problem Solving Perspectives

So far I have discussed how an important knowledge structure (schema) can emerge naturally from neural models and how neural networks can be used to classify information, create a "best fit" to the current data and data from past experience, are capable of spontaneous generalization, able to fill in missing data, and in general, characterize memory as a generative process rather than a selection process. While these are certainly important characteristics of neural systems and have a wide range of applicability in the study of cognition and memory processes, they provide little direct information about the nature of human problem solving.

Human problem solving is a process that requires sequences of actions to be created in attempts to attain a desired goal. By its very definition, human problem solving has a prominent serial component, and serial models have already demonstrated their power for capturing the nature of human problem solving. If problem solving has an essential serial component and serial models have been successful in representing problem solving processes, then why should problem solving theorists concern themselves with neural models?

The obvious answer is that even though serial

models are powerful tools, they are not perfect tools. As has already been described, serial models are not good at describing certain processes such as content-addressable memories which are certainly utilized in, and are perhaps central to, the problem solving process. Both PDP and problem solving theorists agree that much of human problem solving involves the general process of generating adaptive sequences of elementary information processes. They also agree that elementary information processes are essentially parallel operations which take less than 250 ms to complete. Both camps further agree that the parallel and serial components must somehow be associated in order to influence one another's processing. Additionally, Newell and Simon recommend that elementary information processes should be defined on the basis of known mechanisms. Therefore, it seems very reasonable that elementary information processes should be defined in PDP terms. Consequently, any complete theory of human problem solving must contain parallel models of some processes.

The second justification for being concerned with problem solving neural models stems from the fact that the human mind does not operate on two different hardware platforms; one for high level rule-based

processing and a second for highly learned parallel processes. Most researchers in the brain and behavioral sciences view the human brain as a fine-grained, massively parallel system and believe that all behavior results from highly parallel interactions of simple processing units (neurons) in the brain. We further assume that the parallel processes are not processing "packets" which are activated by some central, symbol-based executor. The high-level, symbol-based, problem solving processes somehow emerge from the same highly parallel human brain as do the elementary information processes. If we are concerned with increasing the parsimony of modern human problem solving theory then it is important to develop theoretical formalisms which can accurately account for both serial and parallel processes. We can no longer be satisfied with ignoring the importance of parallel processes by rationalizing that the essence of human problem solving is a high-level, symbol-driven, serial process and does not therefore require any parallel formalism to describe it. Therefore, a complete, modern, human problem solving theory should attempt to explicate how sequential, goal oriented, problem solving processes can emerge from a fine-grained, massively parallel system.

The last, and perhaps the most resilient, justification for problem solving theoreticians to concern themselves with neural models is that they represent a potential well-spring of new ways to conceptualize cognitive processes. Neural network architectures have several degrees of freedom which provide a great deal of flexibility in designing a given system. In addition to the Hopfield Net and the multi-layered perceptron which I summarized earlier, there are several other general types of networks and a variety of variations within each type. Therefore, the designer of a neural net has very few constraints on how she wants the neural net to operate. Because of this flexibility, neural nets can be designed to perform nearly any task that a serial model could perform. That is, they can be designed to solve problems which require serial solutions.

A familiar cautionary note is necessary here, however. Like production systems, a danger exists in taking the PDP models too literally. For any one observable, psychological phenomenon, there exist many neural architectures that can reasonably model the process. Thus, one must be careful not to reason backwards. Simply because a neural net model exhibits behavior which is consistent with empirical

observations of people does not mean that the neural net is a valid model of human processing. In fact, most neural net solutions to applied problems will not represent the equivalent human cognitive process any better than a production system. Ironically, this apparent flaw in the neural network position is also the reason why we should pursue neural net solutions. The sheer number of possible ways to perform a given task using PDP techniques, plus the fact that so few of the possible neural models have yet been investigated, or even designed, indicates that the formalism might be rich enough to yield important insights to cognition, and consequently, how we understand the human problem solving process. In short, problem solving theorists should be concerned with developing problem solving neural nets because systematically investigating new, potential sources of insight is a fundamental task in any scientific field of inquiry.

A Transitional Model of a Hybrid Problem Solving System that Integrates Neural and Symbolic Processes

My fundamental theoretical position is that a general goal of modern human problem solving theory is to specify a highly parallel system which can produce ordered, goal-oriented, sequences of behaviors as

described by serial, symbolic models which have been derived from systematic observations of actual human problem solving activity. The goal of understanding serial processes as emergent properties of a highly parallel, distributed system, however, represents a new emphasis for the study of human problem solving, and complete understanding of the process is a long way off. In the meantime, it is important to develop applications and techniques which unite the symbolic and neural processing paradigms in order to maximize the strengths for both processing schemes while minimizing their respective shortcomings. Systems which use serial executors capable of calling massively parallel sub-systems are already available, and The America Association for Artificial Intelligence (AAAI) held its first workshop on integrating symbolic and neural systems in 1990. Such transitional models seem to be a reasonable, and necessary, step on the road to building new theories, and in this section, I will sketch my own transitional model which integrates symbolic and neural processes.

In accordance with the position taken in McClelland and Rumelhart (1986, Ch. 14), traditional schemes for representing problem solving processes, which have concentrated on representing relatively

long-lived (>250 ms), sequential, often conscious activities, still maintain their usefulness for describing higher level processes. However, the traditional formalisms do not capture the essence of the underlying fine-grained, passively parallel, microstructure of human mental activity. Neural models may provide new insights about human problem solving by demonstrating how many of the characteristics commonly associated with high level processes can emerge naturally from a highly parallel system. For example, neural networks have the potential of being more efficient pattern recognition systems, as compared to sequential models. If one understands problem solving to be the general process of dissolving large processes into pattern-matching sub-tasks (as both Newell & Simon and Rumelhart & McClelland do) then the natural way in which neural networks are able to perform pattern-matching operations makes them potentially important theoretical constructs for the study of human problem solving. (I will expand this point in a later section.)

Unfortunately, neural models use only weight matrices and general learning algorithms to describe the behavioral potential of a system. Thus, many, large matrices of data values must be sifted through in order for a human to derive a readily understood

description of the system's behavior. Furthermore, for a neural network to develop the capacity to perform any type of interesting problem solving task would require large amounts of effort to set the connection strengths either by training the system or by "seeding" the system with values obtained from statistical calculations.

For example, if one were to build a checker playing neural network in a purely neural information processing environment, one would first present a game board to the system and allow the system to settle on a move. Next, an expert would have to provide feedback as to the propriety of the move, and allow the system to back-propagate this information. Finally this process would have to be repeated until the neural network had properly adjusted it's weight matrixes so that it would make the proper move given this particular board configuration. Furthermore, this process would need to be repeated for all possible board configurations. One-thousand training trials per pattern is not unusual and would make the endeavor impractical, if not impossible. One could imagine a similar scenario for a single-layered net. For example, one could present all possible board configurations to the network during training. Here

again, however, system constraints would make this impractical. In fact, the time required just to calculate and construct all possible checker board configurations would be quite large.

Even if the training bottleneck was surmounted, the actual learning (changes in weight matrixes) and expertise (weight matrixes) utilized by the system would not be in a form that is readily understood by people. Knowledge-engineering, as inspired by the work of Newell and Simon, has well developed, relatively efficient techniques for deriving domain knowledge from experts. Furthermore, they are able to represent that knowledge in forms (production systems, frames, traditional programming languages, repertory grids) which are relatively easy to communicate to others and require no system training except for coding.

Thus, my fundamental rationale for specifying a system which integrates neural and traditional knowledge engineering techniques can be summarized in six main points. First, traditional rule-bases are very effective at describing behavior and the conditions under which a behavior should occur. Second, it seems more natural to describe complex, sequential, human problem solving behavior with symbolic, serial models than with neural models which

use only weight matrices and general learning algorithms to describe the behavioral potential of a system. Third, modern knowledge engineers are already proficient in the use of symbolic representations for describing expertise and such a system would allow the use of neural processing without necessitating the retraining of the knowledge engineering work force. Fourth, using traditional formalisms means that currently installed rule-bases can be automatically converted to fine-grained, massively parallel platforms without requiring huge numbers of human-hours for the conversion. Fifth, the use of such a system would eliminate, or greatly reduce, the training bottleneck (i.e. setting the weights) required by most PDP systems. Lastly, I wish to determine what performance benefits and costs are realized when using neural models to process high-level, rule-based knowledge as compared to traditional, sequential processing architectures.

As I stated earlier, my fundamental theoretical position is that the ultimate goal of modern human problem solving theory is to specify a highly parallel system which can produce ordered, goal oriented, sequences of behaviors as described by serial, symbolic models derived from studies of actual human problem

solving activity. Thus, there are three major reasons that traditional knowledge engineering techniques and symbolic formalisms are necessary for gathering and describing expertise in a given problem solving domain. First, traditional knowledge engineering techniques which result in some type of symbolic description (frames, productions, repertory grids) of the problem solving behavior have already been shown to be effective tools for gathering and summarizing knowledge. Second, the descriptions which result from traditional techniques can be used to generate a problem solving neural network. Lastly, and perhaps most importantly, the symbolic formalism can be used to test whether the resulting neural system operates in accordance to the description of actual human behavior (i.e. provides a validity verification tool). Therefore, the hybrid system that I wish to outline should be able to use a traditional symbolic description of human problem solving behavior (frames, productions, repertory grids) as input, parse the formalism, determine the inputs, output, number of nodes, and calculate the weight matrixes for each node based upon the information contained within the formalism.

The process would begin with a knowledge engineer

conducting observations and interviews, analyzing the resulting protocols and building a rule base that performs successfully. For the sake of simplicity, I will use the tic-tac-toe rule base in Figure 13 to represent the output of the knowledge acquisition phase. As one can see from the figure, the rule base is composed of rules which are statements of the form IF <conditions> THEN <actions>. The rule base would be processed by an interpreter capable of parsing the rules into units and organizing the resulting units into a network. For example the system might extract units such as player, opp(onent), two marks, on column, on row, on diagonal, intersection, side, corner, etc. After distilling the units from the representation, the system would construct a matrix which represented all possible connections between the units. Next, for some architectures, the system would calculate the weights for each unit based upon the relationship among units in the rule base (how often any two units co-occur in the rules). This would be accomplished by using a formula comparable to Rumelhart and McClelland's probability of co-occurrence formula mentioned previously. In other architectures (e.g. multi-layered perceptrons), the system would construct a training set with some supervisory information.

Unfortunately, rule bases are efficient representations of knowledge and, as such, may not concretely represent all the information necessary to construct a neural network. In the current case, the events of opp=forking pattern and player=forking pattern would have the same activation level because the unit pairs occur equally often in the rule base. Thus, in the case where both conditions are true, the system would not be able to decide whether to block the opponent or to complete its own forking pattern. One solution to this problem would be to employ some type of weighting rule based upon which term appears first in the rule base. A second option would be to allow the user to manually alter the weights. Doing this, however, reduces the interpreter to a "roughing in" role in which it provides a rough outline of the units, the relationships among them, and the general architecture of the system. A user may, for example, examine the units extracted by the interpreter from the rule base and decide that he wishes to use a different general architecture. He would then be able to specify that he wants a multi-layered perceptron with 3 layers, 18 input nodes and 9 outputs. The system would then map the extracted units onto the system specified by the user. In essence, the system would be able to

automate the neural network engineering process, but the human designer would retain the ability to customize any component of the resulting network.

Another, even more promising, knowledge engineering technique known as repertory grids (Boose, 1986) may provide another avenue for uniting neural and symbolic formalisms. Repertory grids were first used by psychologists to determine personality traits and have been used recently by AI workers to automate the knowledge acquisition phase of expert systems construction. Essentially, the knowledge engineer with the aid of an expert would identify the relevant components of a knowledge domain. These components are then organized into all possible pair-wise combinations and given to the expert(s) who simply make judgements about each pair's degree of relatedness. The expert(s) responses are then fed into a processing package which converts the relatedness ratings into rules. If one re-labels the components of the knowledge domain as units and the relatedness ratings as connection strengths among units then the raw data from a repertory grid represents the fundamental information necessary to build a neural network. Further, if one takes the units and weights data from the repertory grid and couples it with some knowledge about neural

net designs, then one has a system capable of automatically generating single-layered neural networks directly from information provided by experts. In the case of multi-layered perceptron architectures, the system may be able to identify inputs and outputs, but the values of the hidden layers would be difficult to determine on an a priori basis. Even in this case, however, the system might be able to automate much of the process by converting the experts ratings to training materials and playing the role of a supervisor during training.

A major added benefit of using a repertory grid to collect expert knowledge is that the system would be capable of simultaneously generating a rule base and neural network. The rule base could then be used by humans to better understand the neural processing, and could also be used to verify the workings of the neural network. Inversely, rather than having to sift through large volumes of weight matrices, one might be able to alter the neural processing by simply changing the rule base. In effect, the system would reason backwards from the rule change and identify the neural data that should be modified in order to implement the rule change.

Obviously, implementing all the capacities of an

integrated knowledge processing system, as outlined above, would require considerable processing power. However, the benefits of having a system which can use symbolic input in order to build, train, and operate a neural network might also be sizeable. If a knowledge base is quite large, for example, neural systems may be able to instantiate solutions faster than a traditional rule base. Generally speaking, if a neural network has been properly trained then it will always move towards a best-fit solution and never away from it. This means it is possible for a neural network to get caught in a local maximum, but, in most cases, a neural net will take a very direct route to a solution. Traditional rule systems which frequently use sequential search processes, however, often must first exhaust processing branches that lead away from the solution before the proper branch comes to the top of the search queue. Neural nets, therefore, are theoretically faster than traditional rule processing.

In conclusion, the potential advantages resulting from a system that integrates symbolic and neural processing include reducing the neural net training bottleneck, enhancing the understandability of a given neural system's processing, facilitating the debugging and modification of a neural net, allowing existing

rule bases to be ported to neural processing platforms, and making the advantages of neural processing available to expert systems developers without requiring extensive re-training of the knowledge engineering work force. In general, neural network and traditional symbolic processing models tend to complement, rather than compete with, one another. Consequently, a primary goal for the next generation of AI technology should be the complementary integration of symbolic and neural processes so that each model's strengths are maximized and its weaknesses minimized.

Sequential Neural Processing, Consciousness, Mental Models, and Creativity

As is pointed out in Chapter 14 of Rumelhart and McClelland, the "distributed" in parallel distributed processing refers to the serial processing component of a highly parallel processing system. Take for example, the act of recognizing a room. Light reflected from the contents of a room enters the eyes and activates certain patterns of photo-receptors in the back of the eye. This pattern of activation is sent through the optic nerve to the occipital lobe in the back of the brain. The patterns of activations are processed, and lines, edges, and basic forms are extracted. These basic forms are then interpreted to indicate the

presence of walls, windows, drapes, furnishings, etc. The pattern of recognized objects are then interpreted as an entire room and a combination of rooms might further be recognized as a particular house. Each one of the steps is a parallel process in that it simultaneously processes a large number of inputs and, based upon the constraints provided by that input and past experience, is able to relax to a stable state which represents the interpretation of the input. However, the room cannot be identified until the furniture is identified, and the furniture cannot be identified until certain basic forms are recognized, and so on. Thus, the system is a highly parallel system, but the parallel processing has to be distributed such that processes which provide constraints for other processes must be completed before the secondary processes can complete their processing. Hence, the distinction between parallel and serial processing becomes a matter of the time frame in which the system is observed. If one looks at the system over a short time frame (< 250 ms) then the processing is best described as highly parallel. If, however, the system is observed over longer time frames, then the parallel processes can be seen operating in sequence. Thus, PDP models possess an

inherent, serial component which operates at every processing level from low-level perceptual processes up to the highest level conscious processes.

So far I have discussed how a neural system might be able to recognize inputs, but not how it can use that information to execute sequences of actions. From the PDP perspective cognitive processing can be summarized in the following way. An input pattern enters the system and the system relaxes to a state which optimally satisfies the constraints provided by the input and past experience. As was indicated in the discussion of schemata, the pattern of activation across units represents the interpretation of the input. Therefore, each network can represent only one interpretation at a time, and the system maintains its pattern of activation until the stimulus conditions change. Once new data enter the system, it begins again to relax to a new stable state.

In the PDP view of cognition, the contents of consciousness are represented by activation patterns which result when a large subset of the mind's total number of processing units relaxes to a stable state. Therefore, thinking operates on a time scale which corresponds to sequences of large-scale, stable states (i.e., when networks of networks of networks best

satisfy all their impinging constraints). Thus, serial thought processes are viewed as sequences of stable states which emerge from the relaxation of large coalitions of parallel architected, constraint satisfaction networks.

One of the supposed problems with such a system is that it requires new input for the interpretation to change. This is not as big a problem as one might first expect. First, the environment is rarely, if ever, static. Thus, new input is continually entering the system. Even here, however, the model may be unsatisfactory because people don't simply sit by and monitor the world. People affect change in the environment based upon their interpretations of the environment. To account for this in PDP models, the environmental changes initiated by an individual are simply fed back into the system in order to provide a new set of constraints for the system to deal with.

Consider the general processing of a game playing neural system described by Rumelhart and McClelland. A game board is presented to the system. The system takes the position of the pieces as constraints and settles to a stable state which represents the system's move. The new position may provide input to a second, opposing, neural system which settles on a move and, in

turn, provides a new set of constraints to the first neural system. The system can thereby generate a sequence of appropriate moves and play an entire game against an opponent. Even in this scenario, however, the system is entirely reactive and, in effect, deals with each move in a conceptual vacuum that has no expectations of future moves.

One of the things that human players are particularly good at is trying to "out-smart" or anticipate what the opposing player is going to do given a particular board configuration. That is, we are good at creating mental models of opposing players and the accuracy of the mental models is a large determinant of our ultimate success in a given problem solving domain. This can be accomplished with neural nets by connecting two neural systems together (Rumelhart & McClelland, 1986b, p.40). The output of the primary neural network would be sent as input to the second, "modeling" neural net. The modeling net would in turn produce an output which represents a guess about what an opponent might do given the system's move. The output of the modeling network can then be fed back into the primary neural network to determine whether the result of a selected move is desirable. In this fashion, the neural system could

look several moves ahead and even "mentally" play an entire game.

The role of mental models is, in fact, central to the PDP view of thinking and reasoning. Both PDP and traditional problem solving theorists assert that problem solving/reasoning proceeds by breaking a problem down into sub-tasks to which we already possess solutions. In effect we attempt to break problems down into pattern-matching operations at which we are very good and that require minimal processing resources. Rumelhart and McClelland further assert that we have three essential abilities which allow us to perform logical tasks; pattern matching, mental modeling, and manipulating our environment. Take as an example task, the process of multiplying two three-digit numbers (343, 822). Most of us do not have the multiplication tables over-learned up to 822 so we must solve the problem by breaking it down into smaller, more manageable sub-tasks. Thus, we may have already learned to represent the problem by putting one number over the top of the other. We can then "see" that below the right-most column we can write a 6. Next we have learned to multiply the second number in the top row by the right most bottom number, so we write an 8 below the second column of numbers. We repeat this

cycle for each number and start a new row when we begin multiplying by a new number on the bottom row. For each cycle the sequence is the same. First, manipulate the environment so that you create a representation of the problem. Next, use the power of our perceptual system to efficiently process the representation. Last, modify the environment to represent the results of the pattern processing and continue processing. In effect, we have reduced the task to a series of more manageable pattern-matching operations.

Many adults, however, do not require physically representing the problem in order to solve it. They can do it "in their heads" because of the human ability to internalize the representations we create (i.e. build a mental model). Thus, we no longer need to physically write down well-learned problems, but can simply imagine manipulating the representation. Of course this does not apply solely to mathematics, but the entire spectrum of human thought. In the PDP view, human rationality is possible because of our ability to internalize or mentally model external events so that we can imagine manipulating the representations in analogous ways to how we might actually deal with the referent in the external world (Shepard's work on mental rotation seems to support this view as well).

In essence, PDP theorists suggest that human rationality is achieved by utilizing models which are represented by the activation patterns of large coalitions of PDP-like networks.

There are some interesting traits which emerge from such a system. First, once a model is internalized, it can be manipulated in several ways. Normal operations can, of course, be performed on the representation because those operations are part of the internal representation of the referent. By normal, I mean operations that are normally done to the referent in the external environment. However, the representation can be combined with other internal representations which, based upon some selection criteria, possess traits which compliment one another. Thus, new, never before experienced, representations can emerge, and new objects which correspond to that representation can be tested mentally, and/or created, and tested, in the external world.

New representations seem to evolve slowly from combinations of existing representations rather than being created anew. The same seems true of invention. That is, inventions slowly evolve from combinations and/or modifications of existing devices. In fact, it seems reasonable that before an invention can be

constructed, an internal representation of the invention must first be created. Once the invention is built in the external world, observations can be made about its true behavior, and the data from the observations can be fed back into the system so that the invention's internal representation can be altered accordingly. Similarly, modifying the internal representation then allows one to perform similar alterations on the external referent, and the cycle continues until some satisfactory result is obtained.

To further clarify this point, consider the general process of problem solving as outlined by Newell and Simon. For a problem to exist, a task environment and an appropriately motivated problem solving system must be present. The problem solving system must also desire another state of affairs than the one in which it currently resides, and that state must NOT be attainable by any complete, immediately executable, series of actions. In order for the system to determine possible action sequences that could lead to the goal state, the problem solving system forms an internal representation of the problem (i.e. the problem space). Based upon the internal problem space, the system activates a problem formulation which, in turn, allows the for the generation of a problem

solving method. The problem solving method is a sequence of elementary information processes that will hopefully lead to solution of the problem. If the method doesn't lead to a successful conclusion then the method may be modified or replaced with another. If enough methods fail then the problem may be reformulated, and if enough problem formulations are unsuccessful, then the problem space may need to be altered.

It is my opinion that problem spaces, formulations, methods and eips are all processes which naturally emerge from coalitions of PDP-like processes. The problem space is a large, pervasive data structure that seems to be "settled on" very quickly. Likewise, the problem formulation and methods can also be generated rather rapidly. In fact, people seem to be able to begin generating possible solutions to problems almost immediately upon being presented with a problem. Considering the large amounts of data that have to be utilized in representing and formulating a problem, it is difficult to imagine that this feat could be accomplished by anything other than parallel distributed processes. In true problem solving conditions, the first attempted methods will probably not lead to a solution and will require modifying a

method, generating a new method, reformulating the problem, or altering the problem space. Furthermore, it seems reasonable to expect that problem difficulty may be directly related to the degree and level (method, problem formulation, problem space) of adjustment necessary to solve the problem.

To illustrate, consider Edison's invention of the electric light. It is my assertion that the invention of the electric light, as recounted in Freidel and Israel (1986), represents one of the highest forms of creative problem solving, as well as, intermediate and basic forms of problem solving. In 1876, Thomas Edison became interested in creating a reliable, economical lighting system and visited William Wallace's electric dynamo factory. It is reported that during his visit, Edison exhibited child-like enthusiasm for what he saw there, and 10 days after returning to his Menlo Park lab, he boldly announced that he had the solution to the electric, incandescent light. Unfortunately, his proclamation was quite premature.

By this time in his career, Edison had already been granted an impressive number of patents, many of which were in the field of telegraphy. Furthermore, many of the telegraphy patents involved the use of feedback loops to resolve a variety of problems, and it

was the feedback loop that Edison believed would allow for the development of an efficient incandescent light. The problem with the incandescent light was that it had a very short life. An incandescent light operates by passing a current through a filament which heats to glowing. However, very few materials can heat to glowing without melting or oxidizing. The two most promising materials, carbon and platinum, were resilient to melting. Carbon, however, was initially rejected because of its tendency to flame at lower temperatures. Platinum, on the other hand, had the problem of continuing to heat up past its melting point once its temperature had been raised to the point of incandescence.

Edison, quite reasonably, viewed the problem as one of current control and thus designed several feedback loops to circumvent the problem of over heating. The feedback loop regulated the temperature of the platinum filament by restricting the current when the filament reached a certain temperature. When the temperature of the filament returned to an acceptable level, the current would be allowed to flow freely into the filament. Thus, Edison reasoned, the temperature of the filament would remain within acceptable limits, and never overheat.

While this is very reasonable approach to the problem, it did not work satisfactorily because the platinum elements would still distort and fail after remaining at the point of incandescence for a short period. Edison, and his Menlo Park staff, took the next year and four months to develop a new understanding of the elements required to build an efficient, reliable, incandescent light. That is, the initial problem space was faulty, and they spent the next 16 months creating a new one. During that time Edison gathered the some of the best technology, minds, and technicians for the assault on the light bulb. They performed literally thousands of experiments utilizing different designs and materials in one of the more intense technology development efforts ever undertaken. Finally, in October 1879, the Menlo Park team had developed a new understanding of the requirements for the electric light, had dropped the current regulator from the design completely, and now understood how carbon, thread, coils, and a vacuum could be combined to form a reliable incandescent light.

Let me try to characterize this inventive process by integrating PDP and traditional problem solving perspectives. When presented with the task environment

(the development state of the incandescent bulb, state of technology, available resources, etc.), Edison's cognitive machinery settled on a problem space that activated a problem formation (call it the feedback-loop formulation) which had been successful in the past. In fact, if PDP models are reasonable descriptions of problem solving, then Edison may not have had much choice than to characterize the problem in terms other than the feedback-loop formulation.

A problem formulation is activated based on the interactions of connection strengths, which had been determined through a lifetime of experience, and the new problem components (overheating elements, wires, electric current, etc.). The new inputs possessed similarities to the components that existed in the telegraphy problems that he had solved successfully. Thus, the correspondence between the new data and the existing cognitive structures would tend to activate the highly successful, reliable feedback-loop problem formulation. In essence, the components of the incandescent light problem were mapped onto the pre-existing feed-back loop formulation.

Remember that in PDP theory, one can view cognitive structures as networks of schemas and coalitions of schema networks. Thus, mapping the

components of the incandescent light problem onto the feedback-loop formulation is equivalent to instantiating the slots of an already existing schema to the components of the incandescent light problem. Thus, the values of the slots (processing units) have changed but the relationships among the slots remain the same. The process of taking new domain knowledge and mapping it onto a pre-existing knowledge structure is probably one of the most fundamental problem solving techniques, as well as, the primary process by which we can understand the world around us. Simply stated, understanding does not exist until new information has been reconciled with the old. Thus, the model I have described, so far, represents a process by which initial understanding is achieved by analogy, and suggests that we may have no other choice but to initially attempt to solve novel problems by a form of analogical reasoning.

In PDP terms, the process operates as follows. The weightings between the components that comprise a problem space, or formulation, are established by a lifetime of experience. Thus, when new data enter the system, the system attempts to find a pattern of activation which maximally satisfies all the impending constraints. The new inputs represent only a minority

of the total number of impinging constraints and the majority of constraints are provided by the pre-existing internal connections between cognitive elements. Thus, the system will most likely activate a problem formulation which corresponds to previously experienced and successful activity. Because new information must first be mapped onto existing knowledge structures, new insights can only be achieved by modifying old representations. If the new data have a high degree of concordance with the old so that the existing data structure allows for reliable and accurate predictions about the new data (or in PDP terms the old structure is a good model for the new information), then only slight modifications may be needed in order to understand the new information. If, however, the information processing system does not have a pre-existing data structure which adequately models the new problem, then several possible formulations (or problem spaces) may be activated. This results in a "fuzzy" understanding of the new data in which different components of the new data may correspond to components of several different internal representations. The existing structure which fits best may then be modified and refined until a suitable representation is obtained.

In order to modify an existing structure, one must over-ride the influence of previous experience by having frequent experiences with the new data and concentrating considerable attentive and conscious effort on the new and old data. In effect, the experience and conscious effort will eventually allow new connections to be formed between units which will adequately represent the new information.

Sometimes, however, a problem does not lend itself to any known problem formation (known at least to the person attempting to solve the problem). In this case, applying a previously learned representation to the problem only leads to plausible but ultimately ineffectual solutions. Edison was the victim of this when he settled on a solution that involved a feed-back loop to regulate the current, and it required an effort of historically monumental proportions to over-ride this powerful problem representation. When no existing representation is sufficient, the solver must create a new representation that is more than just a mutation of another previously learned representation. One may need to break down several representations, gather new data, and combine all those bits and pieces into a new representation. This is a very difficult process because the old representations are already

established, or to use PDP terminology, the weights have already been set. Therefore, any time the new information is presented, the system settles on the old, strongly connected, representations and blocks the new, weakly connected, structure. Furthermore, before the new knowledge will be fully assimilated, other related knowledge structures may need adjustment in order to maintain their accuracy and reliability. Thus, creating a new representation and gaining new insights is an intense pervasive process that requires a great deal of mental energy in order to over-ride the automatic inclinations of our cognitive machinery.

At some point in the process of modifying weights (learning), the connection strengths between relevant and irrelevant units will be about equal, and the system will have a very difficult time locating a stable constraint satisfying maxima. The point at which both relevant and irrelevant components receive comparable activation is probably experienced as confusion. The combination of a system unable to settle on a stable activation pattern and the substantial effort required to change weights may explain why learning new, difficult information (i.e., information for which no adequate internal representation exists) sometimes results in discomfort,

agitation, and frustration.

In this view of invention, all of the Menlo Park efforts from roughly September 1878 to August 1879 were fundamentally directed at constructing a more accurate internal representation of the components and relationships necessary to build an incandescent light. In order to fully appreciate the difficulty required to create and disseminate anything new, one must remember that the true problem does not exist in the external world but within the problem solver's representation of the problem. Therefore, before an invention can emerge, the internal representation of that invention must first be created. A prototype can then be created in the external world and observations made about its true behavior which, in turn, feed back in to the problem solving system and allows alteration of the internal representation.

Likewise, altering the internal model allows one to determine what modifications to the external referent may be fruitful. If a manipulation seems to work on the internal representation, then one may similarly modify the external working model and observe the results. If the modifications suggested by the internal representation result in too many failures then perhaps the internal representation needs to be

discarded and a new representation formed (as Edison had to do). It therefore seems feasible to expect the evolution of an internal model to be mirrored in the evolution of an invention.

Likewise, the evolution of a technology over generations may reflect the development of a culture's general technological understanding. New inventions result from new internal representations, and new representations evolve from a highly effortful process of modifying existing representations. The new inventions will therefore reflect the changes in the internal models, and it seems reasonable that inventions would appear to slowly evolve from combinations and modifications of existing technology. To be precise, however, inventions do not evolve from existing technology, but emerge from the ever-evolving mental models which represent the current understanding of existing technology.

For example, lighting systems have been employed since the advent of fire. Over time, lighting systems changed from a center fire, to torches, to candles, to kerosene lanterns, to gas lights, arc lights, and to the modern electric light. Each of these advances in lighting represents a new understanding of how lighting could be achieved. Thus, before any new lighting

technology could have emerged, a new internal representation of the process of lighting had to be developed in the mind of the inventor(s). The new representations were built by modifying previously acquired representations in light of new information. The new inventions were built based upon this new representation and reflected the change in the internal representation. Next, once the new representation and corresponding invention are developed, the new representation has to be distributed to other personnel who build, install, maintain, and use the new lighting technology. In order for these other people to effectively interact with the new technology, they must also modify their internal models of lighting. Therefore, disseminating new representations across a culture requires substantial time and effort by the individuals of that culture, and the degree of change required by the new representation determines the speed, and ease, with which a new technology can be absorbed by a given culture. Thus, in general, new inventions, which provide a user/operator interface that reduces the degree of effort that the user/operator has to expend in understanding the workings of the new technology, should be accepted more readily than comparable technology which does not

attempt to reduce the learning curve of the user/operator.

As already described, modifying internal representations is an effortful process. Further, the amount of effort may be proportional to the level and degree of modification necessary to construct a new, appropriate representation, and may partially explain why most inventions evolve and gain acceptance slowly. However, it certainly seems easier to understand how the electric light works than it was to invent the electric light. If understanding, like invention, requires modification of the internal representation then why is it easier to understand the electric light than it was for Edison to invent it? If it is simply due to the newness of the information or the complexity of the solution then it should be just as difficult to understand someone else's new theory/representation as it is to create your own. This does not appear to be the case. It seems easier to understand someone else's ideas than to have created them myself. As an example, I readily understood most of Edison's work and the operation of the incandescent light. In fact, the electric light seems like a rather simple device. I doubt however, that I could have so easily created the electric light as I understood the writings about it.

Part of the answer surely lies in the fact that I do not exist in the same historical context as Edison. After Edison developed his light, the state of knowledge was forever changed. The people who taught science and technology to me had lived in a world where Edison's light had been around for some time. During the interim between Edison's work and my learning of his work, further clarifications, new works which built upon his, and simpler language had been developed to illustrate, explain, and demonstrate the concepts applied by Edison. My representation of electrical technology was therefore compatible with Edison's inventions because it had been partly shaped by Edison's inventions. Furthermore, devices such as dynamos, electric generators, efficient vacuum pumps, and high quality conductive materials are now common place. In Edison's times these devices were high-tech devices and the ultimate solution for electric lighting depended partly upon improvements made by the Menlo Park staff in these technologies. In other words, I simply do not have to, and perhaps never can, solve the same problem as Edison or his contemporaries. It is difficult to fully appreciate the difficulty of past advances without understanding what was <u>not</u> available to those pioneers.

Overcoming the subtle delusions caused by the current informational context is one of the greatest challenges facing those who wish to fully understand the significant developments of the past. That is, we most often have to view the past through the filter of current knowledge. We can, however, gain understanding about the past by making ourselves aware that certain pieces of information were not available, and, by looking at the form of a solution, we may be able to gain valuable insights about the problem representation and the method which generated the solution. Unfortunately, even when our knowledge of the context allows us to empathize with past problem solvers, our current representation of the contemporary world over-rides our imagery of the past, because our representation of the past is still a part of our current knowledge state. Therefore, we are forced to view past advances through a subtle filter of current understanding, which impedes our ability to fully, and accurately, re-create the true problem spaces of past inventors, and we mistakenly over-simplify the nature of the true problem that had been solved.

While the pitfalls of hindsight are partly responsible, the primary reason understanding is easier than creation, however, is most likely due to the fact

that I did not have to deal with as many details as Edison. As described earlier, identifying, and processing, the salient details of a problem are activities which require large amounts of time and effort in novel, or creative, problem solving. By definition, the initial state of a creative problem is large, relatively unstructured, and contains many extraneous components. The essence of problem solving is utilizing, or developing, a strategy which efficiently separates relevant from irrelevant problem elements. If a proven problem formulation (e.g., feedback-loop), which is an efficient means of distinguishing relevant details, fails on a large scale, then the problem solver is left with only weak methods to identify the relevant factors within the problem space (e.g., generate-and-test, do-it-and-see-what-happens, if-carbon-based-then-try-it). This in fact seems to be the case with Edison's light. When his feedback-loop formulation failed, he involved the Menlo Park workers in an exhaustive series of materials tests in hopes of finding the right combination of materials to solve the oxidation and melting problems. (Edison's staff ran electric currents through materials as diverse as metals, coconut fiber, human hair, fishing line, and broom corn to name just a few.)

Once the problem has been solved and new representation for the problem emerges, however, there is no need to examine promising, but irrelevant, details. In fact, a successful formulation will ignore extraneous factors. Therefore, as I read of Edison's work, I concerned myself only with the most salient details. I simply will not have, and probably do not need, as detailed an understanding of all the problem components as Edison. Not having to attend to irrelevant details removes much of the processing burden from the student of an invention while it may have actually consumed much of the inventor's energy. Consequently, creating an invention differs qualitatively from understanding an invention much as discovering the Cumberland Gap differs from driving a car on the road which now runs through the pass. Although he probably didn't think in these terms, I believe Edison was referring to the severe processing burden imposed by weak methods, and the mechanics of constraint satisfaction processing in the human mind when he observed that, "There is no expedient to which a man will not resort in order to avoid the real labor of thinking."

Much like the distinction between learning about and creating an invention, a similar question can be

asked of the distinction between creative and traditional problem solving. That is, if problem solving is really a matter of modifying and recombining old representations then what is the difference between traditional problem solving and creative problem solving? One of the possible distinctions between creative and traditional problem solving is the degree and manner in which a representation is modified. In the simplest case, problem solving is simply a matter of using a representation. Consider a second grader who is learning to multiply and applies the problem representation for 7x3= to 8x4=. The quantities of the problem change. However, if the problem representation is sound, then the operations should still be successful. There is very little change in this representation, but solving the problem still requires effort for the novice multiplier.

At the other extreme may be the case of Edison, who had to reject a rich, reliable, and deceivingly promising representation, and create a new one by discovering new information, breaking apart pieces of old representations, and combining all the pieces to form a virtually unthought of representation for the physics, chemistry, and architecture of an incandescent electric lighting system. In the middle may lay the

cases where one makes moderately difficult modifications to an existing representation or applies an existing representation to a problem domain which it had never been applied.

Problem difficulty may in large part be determined by the degree and level of modification necessary to resolve a given problem. Creativity, however, is not defined by the difficulty of the processing, but is defined by the rarity of the solution. In this view, traditional problem solving and creative problem solving may be operationally identical. That is, traditional problem solving usually requires modification of a method which is a relatively simple alteration process. However, if that modification results in significant savings in time and/or effort required to accomplish a task and the solution is rare in the given cultural context then the act is said to be creative. The ability for the solver to generate a rare method may be indicative of the uniqueness of that individual's problem space. If the internal representation is unique, then, for the solver, the problem was relatively easy to solve. However, someone, who did not begin with a problem space that allowed for such a unique modification, would have to exert much effort to construct an internal problem

space that would facilitate a culturally rare solution. Conversely, it is possible for someone to exert mammoth efforts to construct a new understanding of a problem, but the ultimate solution may not be judged creative. For example, present a modern automobile to an individual, who has minimal auto maintenance experience. Furthermore, inform him that the car is "running rough" and it is his job to fix it. The naive mechanic may have to completely reformulate the problem of automobile maintenance in order to understand the nature of the malfunction and its remedy. That is, from the novice's perspective, the problem space is relatively large, unstructured, and few formulations exist for traversing the problem space. Thus, from the perspective of the novice's internal processing, learning to repair a car is an effortful, creative act, but from the perspective of our culture, it is not rare and is not, therefore, creative.

Whether we judge the final product of processing as creative depends upon the product's rarity within its cultural context. In general, the highest forms of creativity will correspond to internal processing that requires disassembly of several internal representations, gathering of new information and assimilating all the information into a new internal

representation which generates some product of value to the solver or society. In many cases, however, a person may have to undertake this highly effortful process to solve a traditional problem. Likewise, there may be cases where a relatively simple modification results in a rare, creative solution. Therefore, while there is a degree of correspondence between creativity and problem difficulty, as determined by the magnitude and level of modification required by the internal representations, the correspondence is not perfect.

Problem difficulty and creativity both involve modifying internal representations which are reflected in the activation patterns of large coalitions of neural networks. Neural networks are, in turn, composed of large coalitions of processing units (i. e. neurodes) and the data structure which emerges from this architecture can be described as a network of generative schema. Furthermore, it is my assertion that the judgement of creativity is based upon a cultural context and not upon the nature of the underlying processing. Therefore, the internal processes utilized in a large segment of creative problem solving is operationally equivalent to traditional problem solving processes in that both

utilize processes that modify pre-existing internal representations. A creative act, however, results when these same processes act to form a culturally unique problem representation that is effective at resolving a given problem. It therefore seems reasonable that an appropriate manner to study creativity is to uncover operations (heuristics) which might lead to unique problem representations. Interestingly, alterations which emerge naturally from the processing of constraint satisfaction networks generate heuristic-like modifications of existing schema/internal representations.

In the conceptual framework discussed so far, the task environment consists of everything available to the subject in the external environment plus the cognitive potential of the problem solving system as represented by all the connection strengths in the system. When the system is placed in a problem solving situation, the task environment activates a problem space which is represented by a pattern of activation across a large coalition of networks.

Likewise, the activation pattern of the problem space, via the inputs from the external world and the internal connection strengths, activate a problem formulation. The problem formulation, subsequently

activates a method which, in turn, activates a series of eips. It should be made clear that even though only one problem representation, formulation, and method may fire, several may receive partial activation that does not meet the activation threshold. In a normal, relatively well learned, problem solving domain the system is able to settle on successful patterns quickly. That is, the external input is consistent enough with stored connection strengths, that the system can quickly settle on the appropriate pattern of activation and generate the corresponding behaviors. However, if the external problem constraints and the internal representations do not "fit" one another well enough to activate a known representation (i.e. a tightly bound coalition of networks and units), then the system settles on a spurious pattern of activation.

For example, Rumelhart and McClelland clamped on two descriptors, sofa and bed, which were strongly-predictive, but mutually contradictory, descriptors for the living room and bedroom, respectively. When both predictors were clamped-on, the resulting pattern of activation did not represent a normal living room or bedroom. Instead the pattern of activation defined what could be called a luxurious bedroom which was large, contained a bed, a lounge chair, a dresser, a

fireplace, and a sofa. The spurious patterns that can result, when strongly-predictive, contradictory constraints are activated, may take several forms which correspond, at least roughly, to some of the combinatorial possibilities outlined by Weber & Perkins (1989). The spurious patterns are important because they provide clues as to how the internal representation might be modified in order to resolve a given problem or create a new artifact.

In the simplest case, the system activates a series of behaviors which do not successfully accomplish the goal. The system then receives information that notifies it of the failure and the system begins to search for a better representation. Parts of the representation and corresponding behaviors, may be judged as faulty based upon the external data and data from previous experience. The components which are deemed faulty are inhibited which results in a new pattern of system constraints. The system tries to relax to a new stable state which may result in simple omission of certain methods, in replacement of some methods, and/or reorganization of the methods.

In more complex cases, however, previous experience and external data cannot combine to generate

a well defined, stable state. In this case, as in the Rumelhart and McClelland case, the system forms, after some substantial jumping around, a new coalition that is the raw data for one class of creative thought. In effect, no one representation, formulation, or method are fully instantiated. Therefore, only portions of several representations are activated. In one case, the spurious activation pattern may take the form of an ANDing operation in which two independent schemas are activated as one, or where only a few components of one representation are activated in unison with another, complete, schema. It is important to realize that while this process activates certain related components, it may also eliminate under-supported components and result in operations that are equivalent to ORing and XORing.

The spurious pattern that results from this natural activity of constraint satisfaction networks is the raw data that may be used in the next, and perhaps most laborious, problem solving process. In most truly novel problem solving situations, the patterns of activation, which emerge from this process, will not be wholly complete or accurate. That is, some of the activated components will be unnecessary and even contradictory while other necessary patterns will not

reach threshold and will not, therefore, be part of the overall pattern of activation. The problem solving system must therefore engage in a process whereby it builds up connection strengths between relevant components and eliminates connections between undesirable components.

The process is a two pronged process involving conscious effort and experience. The connection strengths are accumulated through experience with the components and the relationships among the components. As components tend to co-occur with one another, the units which represent hypotheses about their presence become more tightly connected. Thus, experience with the components, in the form of repeated exposures to the units, is necessary for the proper connections to be built.

Simple repetition and rehearsal, however, are probably not enough to account for the ability to form new representations. Due to the system's tendency to settle on a previously stored pattern of activation, simply exposing the system to the factors would probably require huge numbers of exposures, during many different states of activation, in order for the new information to be completely assimilated.

Instead of presenting the system with massive

numbers of external exposures, humans are able to make multiple presentations of the factors by "imagining" the components in concert with other information. This conscious, effortful, activity allows for efficient multiple exposures to the system, so that weights can be changed more quickly than with the brute force required of multiple external exposures. It is also possible that consciousness allows for the adjustment of weights by activating, or in some way involving, a chemical "broadcast" process in the human brain which inhibits previously stored weights and facilitates construction of new connections across the system.

Lastly, non-conscious attention probably plays a large role in resetting connections by allowing the system to operate on other problems while continuing to propagate the new data throughout the system. Such processing may account for insights that purportedly come after a period of incubation.

If creative problem solving proceeds by refining spurious patterns of activation then insight may have several flavors. One type of insight may result when a spurious pattern of activation is activated and new combinations are therefore available to the system for further processing. A second form of insight may be experienced when the system gains enough experience

with a collection of phenomena that a new, stable, pattern of activation emerges. The third flavor of insight may result when a pattern of activation is edited such that it behaves consistently with all, or at least most, related representations (i. e. the representation has been fully assimilated).

Summary and Conclusions

My intent in the previous section was to outline the possible correspondence between neural and traditional problem solving models. My position is that both views have much to offer one another and are, in fact, much more complimentary perspectives than they are adversarial. The two theories mesh nicely in their level of explanation and can be combined to form rich, integrated, processing models that can be applied to real world information processing problems. I have also tried to describe how common, and creative, problem solving processes may emerge naturally from an underlying constraint satisfaction processing model. It has not been my intent to provide proof for the existence of such processes because such proofs may require career-long efforts. Instead, my goal has been to suggest reasonable processes which allow the reader to envision the varied, potentially important, insights that neural models can contribute to the study of human

problem solving and creativity.

When I first became interested in the two general fields of neural networks and human problem solving, there seemed to be a battle raging in the cognitive sciences about the rightful places of traditional sequential and the newer neurally-inspired information processing models. Upon reading and studying two of the landmark works in both fields, I have come to the conclusion that the theories of the four predominate theorists (Newell & Simon and Rumelhart & McClelland) have relatively few points of dispute. In fact, there seems to be a great concordance between the paradigms with respect to their relative positions in the theoretical landscape, and their perspectives on the nature of human problem solving.

First, it should be pointed out that the Newell and Simon theory is mainly concerned with describing behavior in order to deduce precise, abstract models (production systems/programs) of underlying human knowledge structures. It was their view that the programs which resulted from such investigations would accurately predict human behavior, but that the program itself should not be viewed as a specification of the actual processing mechanisms. Newell and Simon took great care in divorcing their theory of human behavior

from any detailed description of underlying physiological and computational processes, and in fact, acknowledged that much of the actual underlying mechanism of human behavior was most likely parallel.

The Newell and Simon theory is concerned mainly with describing the outward behavior of humans in problem solving situations and inferring from that behavior a precise, accurate, parsimonious model of the individual's internal problem representation. Their theory is related to machinery only in that the computer metaphor provided them with a theoretical tool capable of the descriptive and predictive precision which they sought.

The serial von Neumann computer provided them with the proper degree of precision because they viewed the essence of human problem solving to be a highly-integrated, sequential process (a point with which most PDP theorists agree). The serial computer metaphor provided them with a flexible, powerful, "perfectly rational" problem solver to which human behavior could be compared and thereby provide a greater understanding of human rationality. Newell and Simon contend that a major goal of the study of human problem solving is not only to create machines which can mimic the problem solving proficiency of humans, but to describe and

explain human problem solving performance with such precision as to be capable of disseminating the knowledge in a useful way to other humans; be they expert, novice, or theoretician.

The PDP theorists, on the other hand, are more concerned with specifying low-level cognitive mechanisms that are computationally powerful enough to produce the entirety of human behavior. Neural models have gained momentum in recent years due to three occurrences. First, connectionist computer architectures have become available which make highly-parallel computing processes practical. Second, traditional serial architectures have proven too cumbersome, even at high processing rates, to efficiently solve "monster" AI problems such as content-addressable memory, speech recognition, scene interpretation, and any other process that requires pattern-matching processes that contain large quantities of data points. Lastly, learning-algorithms and transfer functions have been developed which overcome weaknesses of the earlier neural networks.

Neurally inspired computing models overcome the weaknesses of serial processes by taking a different approach to data processing. In serial computing, the amount of time required to identify a pattern increases

with the number of data points to be interpreted. On the other hand, human beings, the most advanced PDP system we know about, seem to be able to produce solutions more quickly when given more information. Humans are able to make better use of context effects whereas serial computers must process each element in a virtual vacuum, thereby, increasing the time required to process all relevant information. Neural nets, analogously to humans, are able to make more effective use of the multiple constraints provided by the context instead of being burdened by them as are serial machines.

PDP, or neural network, models are inspired by the architecture of the brain. First, they have no central processor, but are composed of a large number of highly-interconnected simple processors which interact and constrain one another in ways determined by the relative connection strengths that exist between them. Secondly, neural networks are not programmed but are trained. Thus, the essential character of processing is a constraint satisfaction procedure in which a very large number of constraints acts to produce behavior rather than select a behavior from a predefined pool of possible procedures. Lastly, no knowledge is explicitly coded in the system, but instead, exists

within the connection strengths between the processing elements.

According to their books, Rumelhart & McClelland and Newell & Simon, agree on nearly all major topics concerning human problem solving. Both agree that a large portion of human cognition has a sequential nature and that much of human problem solving behavior can be captured in serial models. Further, they agree that human problem solving proceeds by dissolving the problems into sub-tasks for which the solver already possesses solutions. This is tantamount to saying that problem solving proceeds by reducing the problem into pattern matching tasks which are probably highly parallel operations (Newell and Simon refer to these processes as elementary information processes). Thus, the serial nature of problem solving results from executing sequences of these elementary, parallel processes as directed by a successful problem solving strategy.

The two camps also agree that if a process takes less than half a second then it is probably parallel, and if it takes more than 500 ms then it probably has a serial nature. Of course there is a gray area (250 ms to 500 ms) in which both camps claim some dominance and it seems likely that some sequences of operations can

take place in less than 500 msec while some parallel processes may require longer than 250 ms to settle on a solution. They agree on the fact that the contents of consciousness/STM are probably the result of processes done in parallel (content addressable memory retrieval v. settling of large networks to a maxima). The two camps even agree to a large degree on their respective places in the theoretical landscape and is captured nicely by Rumelhart and McClelland. In essence, it is their view that at the low-end of cognitive processing there is a relatively high degree of understanding. That is, we tend to have relatively good models for low level processes such as color recognition, edge detectors and the like. Likewise, we also have a rather good understanding of the highest level, most conscious processes, because if we didn't, we would not be able to communicate with one another. In the middle between these two end points, however, there exists a sizeable hole in our understanding. Serial theorists attempt to illuminate this chasm by climbing DOWN into this pit with their methodological flashlights. PDP theorists attempt to climb UP into the breach with their methodological flashlights in hand. The hope is that the two will eventually meet someplace where their combined lights will illuminate the entirety, or at

least most, of the gulf that exists in our current understanding of human cognition. It is my hope that the current paper has provided some power to both theoretical lanterns and thereby helped to reduce the gulf that exists between the two paradigms' respective areas of illumination.

People are confronted with a problem when they desire a goal and do not posses an immediately available method to obtain the goal. The problem solver must therefore formulate a strategy to obtain the goal. Formulating the strategy involves dissolving the problem into its components and performing a sequence of elementary processes upon those components which result in attainment of the goal. Each elementary processes is readily accessible to the solver, takes less than 500 ms to complete, is probably a pattern-matching process, and is most likely a highly parallel process, or a tightly-bound sequence of parallel processes. If the problem formulation is correct, and the solver is given enough time to complete the strategy, then the desired goal will be obtained.

On the surface, it seems a straight-forward operation to combine these two processing models into one complete system. First, one needs only to specify

a serial executor which is capable of formulating a strategy and planning a sequence of actions. Second, couple the executor with a PDP architected memory and procedures and let the system run. Of course there are many technical issues that would have to be addressed before such a system were operational, but the general concept is sound. In fact, many firms are now looking to create such hybrid systems. Specifying applied systems which use both serial and neural processes is important for applied researchers, but may also provide a transitional model for cognitive theories. Eventually, however, our theoretical models will mostly describe serial processes in terms of parallel processes.

I am not advocating discarding concepts such as problem space, heuristics, and internal representations. Nor am I denying the fact that the fundamental nature of human problem solving appears to sequential. Quite the contrary, these are important concepts and should be integrated with PDP models to develop a cohesive model which encompasses both the parallel microstructure and serial macrostructure of human problem solving.

In fact, I believe that PDP models would be completely insufficient models of human problem solving

and the study of human problem solving would come to a halt if the only methodological tools available were the currently available PDP models. The two theoretical camps need each other if either is to advance.

Newell and Simon have outlined an effective procedure for studying human behavior and have established an accepted and relatively understandable formalism for describing human performance. Unfortunately, that system does not adequately explain how such behaviors can be produced by a highly-parallel system such as the human nervous system. PDP models, on the other hand, have the potential of modeling a great deal of low level mechanism and explaining much of the phenomena which is observed in the cognitive psychology laboratory. However, pure PDP models of higher level processing are difficult to build and test because of the huge training overhead.

Ironically, neural networks may model high-level human processes too accurately to be of direct use. Just as humans may take years to learn a high level skill (chess playing, invention, novel writing, etc) it might take a pure neural net just as long to be trained in the same high-level knowledge domain. Thus, a synthesis of the two approaches seems prudent.

First, much of the data collection techniques used in the study of human problem solving will remain unchanged. The procedures outlined by Newell and Simon, and refined by a myriad of other knowledge-engineers, still seem adequate and useful. Likewise, much of the model building will still use production systems or other symbolic formalisms to describe problem solving behavior. In the short term, these systems will most likely be coupled with neural network sub-systems in order optimize certain pattern recognition operations. A second option for the applied world is to develop systems, such as the one described in the previous chapter, which can translate production systems/symbolic descriptions to neural network platforms and back again. In the pure research realm, similar hybrid model building will take place, but eventually, the theoretical vernacular will probably take on a more PDP-like flavor. While I do not propose tossing out important concepts such as problem space, problem formulation, and heuristics, I do think it is time that we began trying to specify in PDP-like terms just what it means for someone to be using a "means-ends heuristic" and how such a heuristic can be implemented on a highly parallel platform. Likewise, how does a highly parallel system formulate a

problem and plan a strategy for resolving that problem? These are all very complex problems and will not be resolved in the immediate future. They do, however, represent a general goal of cognitive science and the attainment of that goal will represent the merging of the two information processing paradigms into one.

As scientists we are obligated to attempt to explain our findings at the level of greatest specificity available to us. After all, along with the ability to produce adequate explanations, precision and parsimony are two of the most important criteria by which a scientific theory is judged. Again, it is not reasonable to discard serial process models, because human behavior does have a strong serial component, and sequential, symbolic models represent an appropriate level of explanation. With the advent of PDP models however, we should not be content with the serial level of description. While it will take some time for the transition to occur, cognitive scientists, who study human problem solving, now need to attempt to take their models to another level of specificity by postulating how their serial models can be implemented on a highly parallel system.

As I hope I have demonstrated in this paper, PDP models provide us with a potentially important avenue

to new insights and understanding of human problem solving, and I believe we should add one more goal to the goals outlined by Newell and Simon for the study of human problem solving. In order to further the study of human problem solving, we should now attempt to specify how a highly parallel, PDP-like system can produce ordered, goal-oriented, sequences of behavior which are consistent with human performance as described by serial, symbolic models derived from studies of actual human problem solving activity.

References

Anderson, J. A. (1977). Neural models with cognitive implications. In D. Laberge & S. J. Samuels (Eds.), Basic processes in reading perception and comprehension, 27-90, Hillsdale, NJ: Erlbaum.

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. IEEE Transactions on Systems, Man & Cybernetics, 13, 799-815.

Anderson, J. R. (1985). Cognitive psychology and its implications. New York: W. H. Freeman and Company.

Boose, J. (1986). Expertise transfer for expert system design. New York: Elsiver.

Brown, J. R., Garber, M. M., & Venable, S. F. (1988). Artificial neural network on a SIMD architecture. Martin Marietta Electronic Systems technical report.

Caudill, M. (1988). Neural networks primer. a series in AI Expert, December 1987 - November 1988.

Charness, N. (1976). Memory for chess positions: resistance to interference. Journal of Experimental Psychology: Human Learning and Memory, 2, 641-6 53.

Chase, W. G. & Simon, H. A. (1973). The mind's eye in chess. In W. G. Chase (Ed.), Problem solving. New York: Academic Press.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, 121-152.

de Groot, A. D. (1965). Thought and choice in chess: An overview of a study based on Selzean theory. In N. H. Frijda & A. D. de Groot (Eds.), Otto Selz: his contribution to psychology. New York: Mouton Publishers The Hague.

de Groot, A. D. (1966). Perception and memory versus thought. In B. Kleinmuntz (Ed.), Problem solving. New York: Wiley.

Ericsson, K. A. & Simon, H. A. (1980). Verbal reports as data. Psychological Review, 87(3), 215-251.

Ericsson, K. A. & Simon H. A. (1984). Protocol analysis. Cambridge, MA: MIT Press.

Feldman, J. A. & Ballard, D. H. (1982). Connectionist models and their properties. Cognitive Science, 6, 205-254.

Feldman, J. A. (1985). Connectionist models and their applications: Introduction. Cognitive Science, 9, 1-2.

Freidel, R. & Israel, P. (1987). Edison's electric light: Biography of an invention. New Brunswick, New Jersey: Rutgers University Press.

Grossberg, S. (1986a). The adaptive brain I: Cognition, learning, reinforcement, and rhythm. Amsterdam: Elsevier-North Holland.

Grossberg, S. (1986b). The adaptive brain II: Vision, speech, language, and motor control. Amsterdam: Elsevier-North Holland.

Hayes-Roth, F., Waterman, D., & Lenat, D. (1983). (Eds.) Building expert systems. Reading, Mass.: Addison-Wesley.

Hebb, D. O. (1949). The organization of behavior. New York: Wiley and sons.

Hillis, D. (1985). The connection machine. Cambridge, Mass.: MIT Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Science USA, 79, 2554-2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Science USA, 81, 3088-3092.

Hopfield, J. J. & Tank D. W. (1986). Computing with neural circuits: A model. Science, 233, 625-633.

Jones, W. P. & Hoskins, J. (1987). Back Propagation: A generalized learning rule. Byte, October, 155-161.

Kellog, R. T. (1982). When can we introspect accurately about mental processes? Memory and Cognition, 10, 141-144.

Kellog, R. T. & Holley, C. S. (1983). Interference of introspection with thinking in concept identification. Perceptual and Motor Skills, 56, 641-642.

Kohonen, T. (1984). Self organization and associative memory. New York: Springer-Verlag.

Lippmann, R. P. (1987). An introduction to computing with neural nets. IEEE ASSP Magazine, April, 4-22.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.

Mckeithen, K. B., Reitman, J. S., Reuter, H. H., & Hirtle, S. C. (1981). Knowledge organization and skill differences in computer programmers. Cognitive Psychology, 13, 307-325.

Miller, G. A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. Psychological Review, 63, 81-97.

Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), The psychology of computer vision, 211-277, New York: McGraw-Hill.

Minsky, M. & Papert, S. (1968). Perceptrons. Cambridge, Mass.: MIT Press.

Newell, A., Shaw, J. C. & Simon, H. A. (1958). Elements of a theory of human problem solving. Psychological Review, 65, 151-166.

Newell, A., Shaw, J. C., & Simon, H. A. (1962). The process of creative thinking. In H. E. Gruber, G. Terrell, & M. Wertheimer (Eds.), Contemporary approaches to creative thinking, 63-119, Englewood Cliffs, New Jersey: Prentice-Hall.

Newell, A. (1973). Production Systems: Models of control structures. In W. G. Chase (Ed.), Visual information processing, New York: Academic Press.

Newell, A. & Simon H. A. (1972). Human problem solving. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Nisbett, R. E. & Wilson, T. D. (1977). Telling more than we can know and verbal reports on mental processes. Psychological Review, 84, 231-259.

Posner, M. I. & Synder, C. R. R. (1975). Attention and cognitive control. In R. L. Solso (Ed.), Information processing and cognition: The Loyola Symposium, 55-85, Hillsdale, NJ: Erlbaum.

Rosenblatt, F. (1959). Two theorems of statistical separability in the perceptron. In Mechanisation of thought processes: Proceedings of a symposium held

at the National Physical laboratory, November 1958. 421-456. London: HM Stationary Office.

Rosenblatt, F. (1962). Principles of neurodynamics. New York: Spartan.

Rumelhart D. & McClelland, J. (1986a). Parallel distributed processing, explorations in the microstructures of cognition volume 1: Foundations. Cambridge, Mass.: The MIT Press.

Rumelhart D. & McClelland, J. (1986b). Parallel distributed processing, explorations in the microstructures of cognition volume 2: psychological and biological models. Cambridge, Mass.: The MIT Press.

Shneiderman, B. (1980). Software psychology. Cambridge, MA: Winthrop.

Turing, A. M. (1950). Computing machinery and intelligence. Mind, 59, 433-450.

Von Neumann, J. (1958). The computer and the brain. New Haven, Conn.: The Yale University Press.

Weber, R. J., Moder, C. L., & Solie, J. B. (1990). Invention heuristics and mental processes underlie the development of a patent for the application of herbicides. New Ideas in Psychology, 8, 321-336.

Weber, R. J. & Perkins D. N. (1989). How to invent artifacts. New Ideas in Psychology, 7, 49-72.

Figure Caption

Figure 1. The silhouette of the mechanical object used in the experiment. Neither this drawing, or the drawing of the nonsense object, are complete technical drawings of the objects actually used. Both drawings, however, reasonably depict the relative visual complexity of the two stimulus items.

Figure Caption

Figure 2. Silhouette of the nonsense object which is constructed of materials identical to the materials used in the mechanical object.

Figure Caption

Figure 3. Experimental lab arrangement.

Figure Caption

Figure 4. Average number of correct connections made during non-zero trials for experts and novices in the mechanical and nonsense conditions.

Figure Caption

Figure 5. Average number of correct connections made when both zero and non-zero trials were included for experts and novices in the mechanical and nonsense conditions.

Figure Caption

Figure 6. Average elapsed time necessary to construct both objects by experts and novices.

Figure Caption

Figure 7. Average number of disconnects (errors) made by experts and novices in the mechanical and nonsense conditions.

Figure Caption

Figure 8. Average number of zero trials for expert and novices in both conditions. The effects of expertise, object type, and the interaction were all non-significant ($F(1,2) = .35$; $F(1,2) = 5.89$; $F(1,2) = .51$; respectively).

Figure Caption

Figure 9. Average number of looks taken by experts and novices during the construction of both objects.

Figure Caption

Figure 10. Newell and Simon's (1972) schematic of an Information Processing System (IPS).

Figure Caption

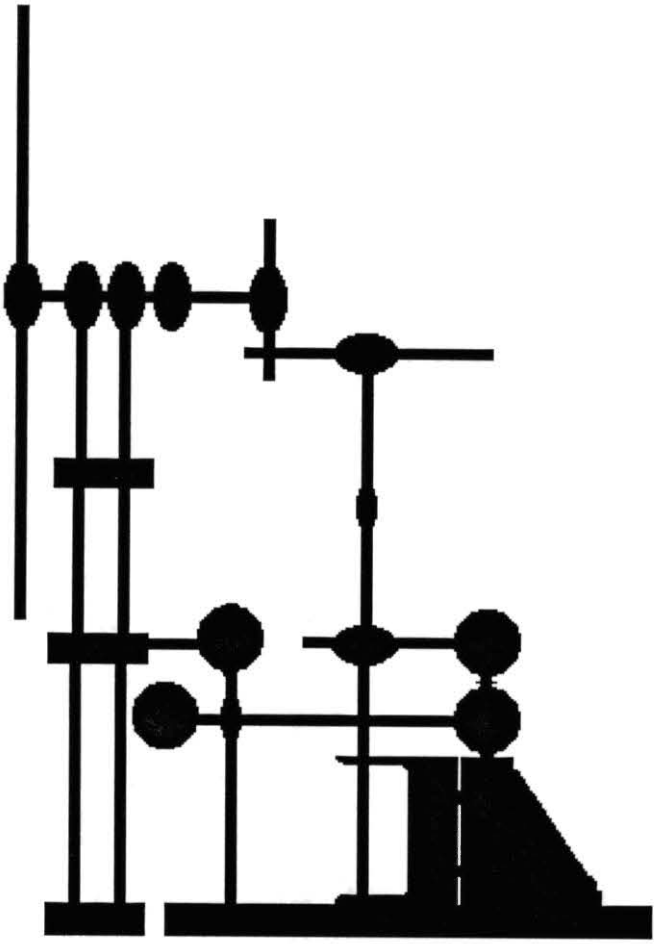
Figure 11. Program which describes the behavior of a thermostat.

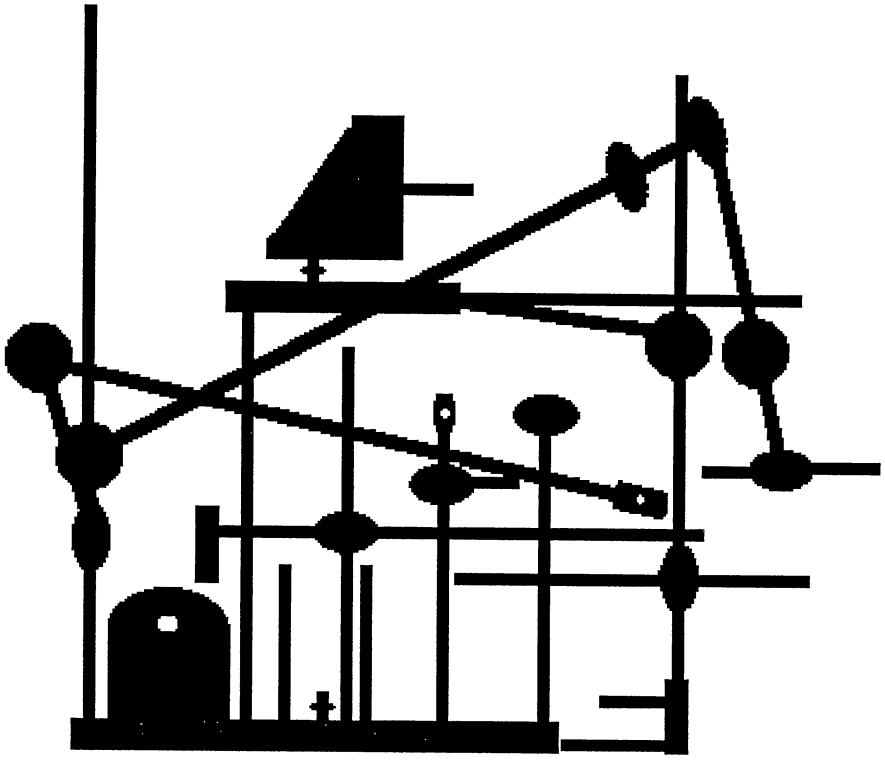
Figure Caption

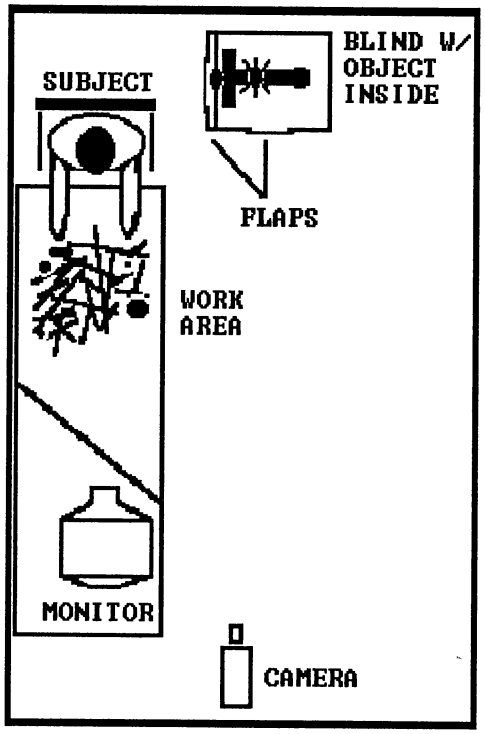
Figure 12. Game state of a tic-tac-toe game in which the most successful move for X is the lower center.

Figure Caption

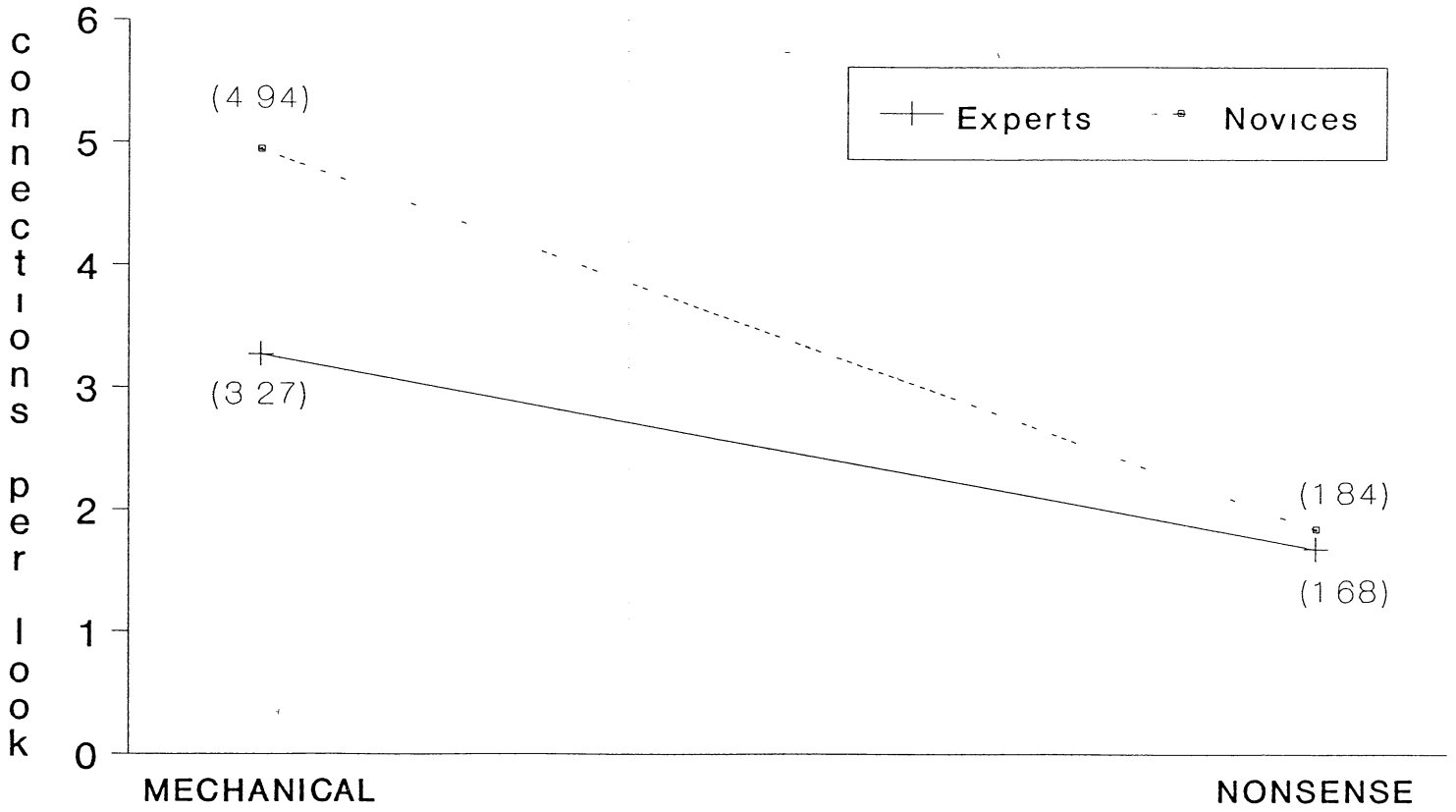
Figure 13. The rule-base on the left represents what we, as external observers, may propose as a rational problem solver given the task demands of a tic-tac-toe game. The rule base on the right represents the player's problem space which is inferred from her actual game playing behavior.





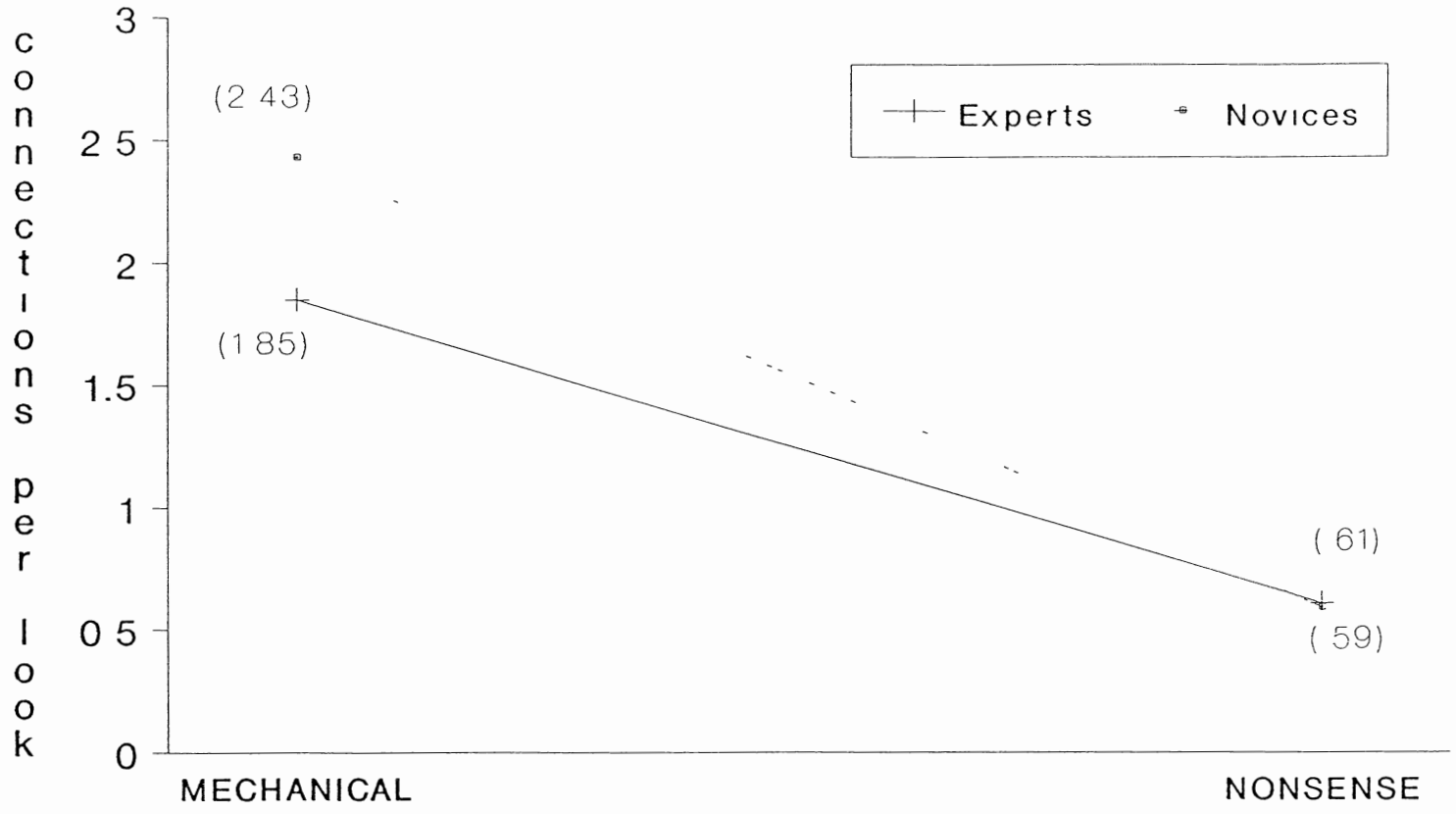

SUBJECT
BLIND W/
OBJECT
INSIDE
FLAPS
WORK
AREA
MONITOR
CAMERA

AVG # OF CORRECT CONNECTIONS
(DURING NON-ZERO TRIALS)

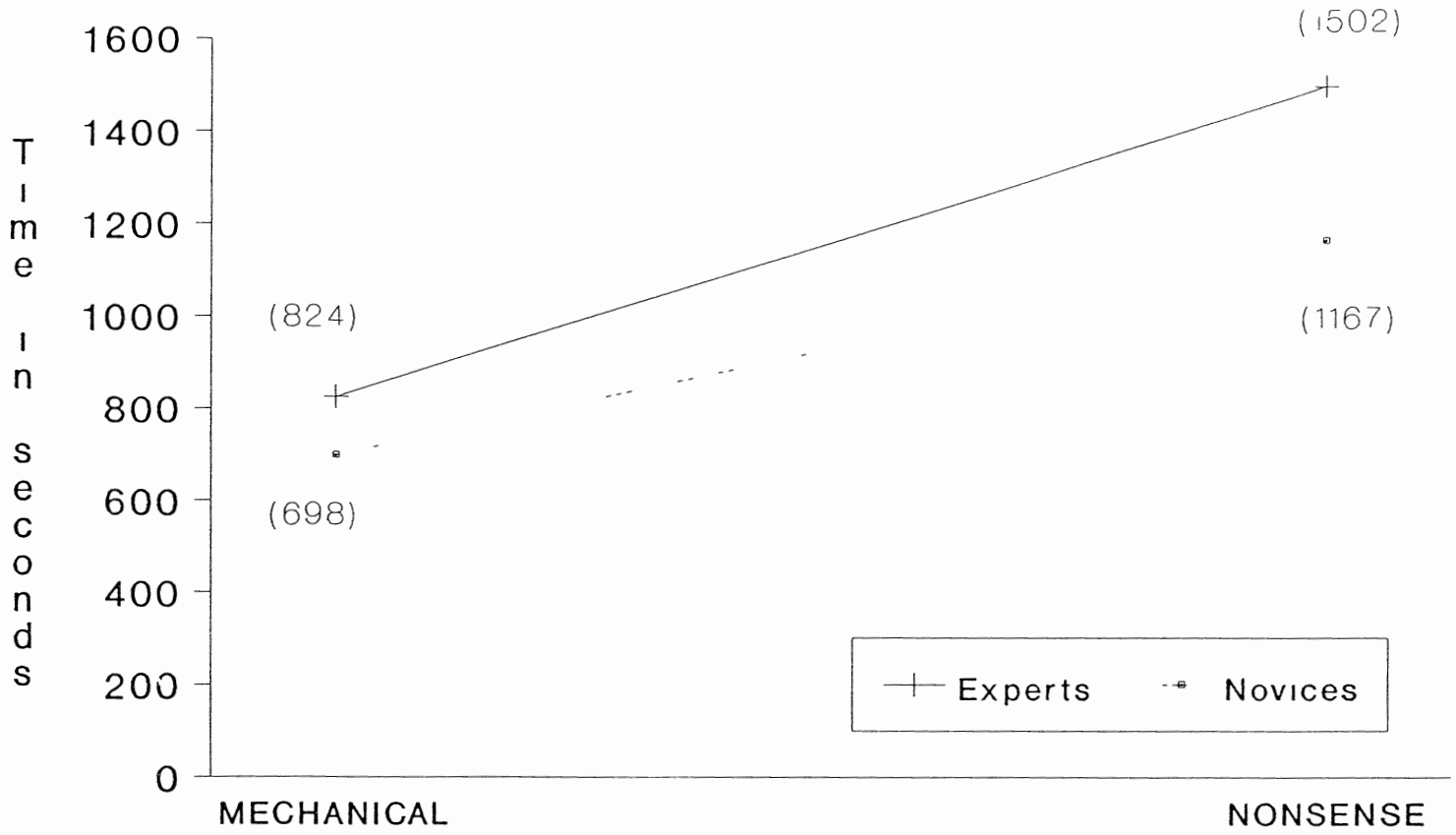


OBJECT TYPE

AVG # OF CORRECT CONNECTIONS
(INCLUDING ZERO & NON-ZERO TRIALS)

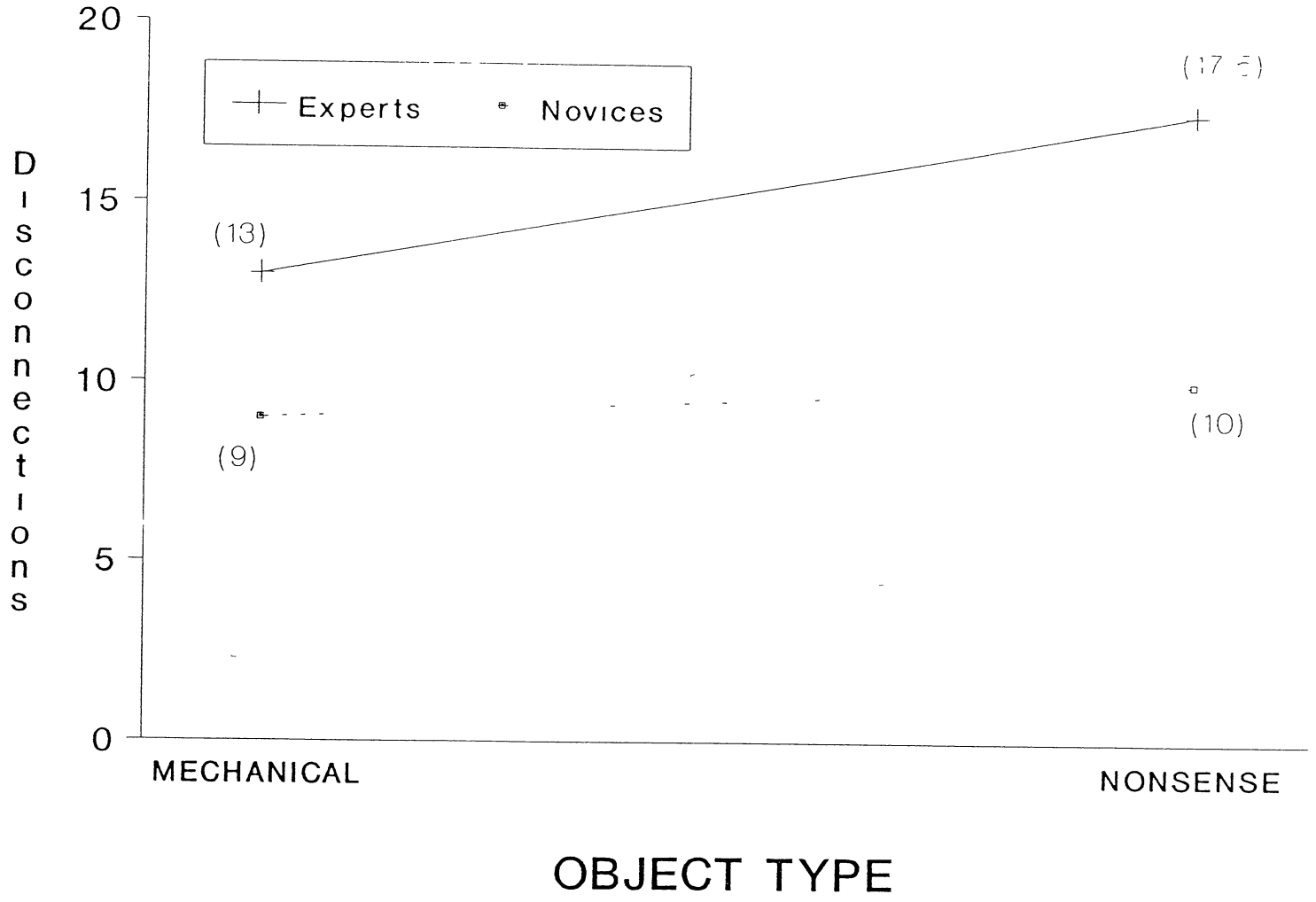


Average Elapsed Time
(thru last connection)

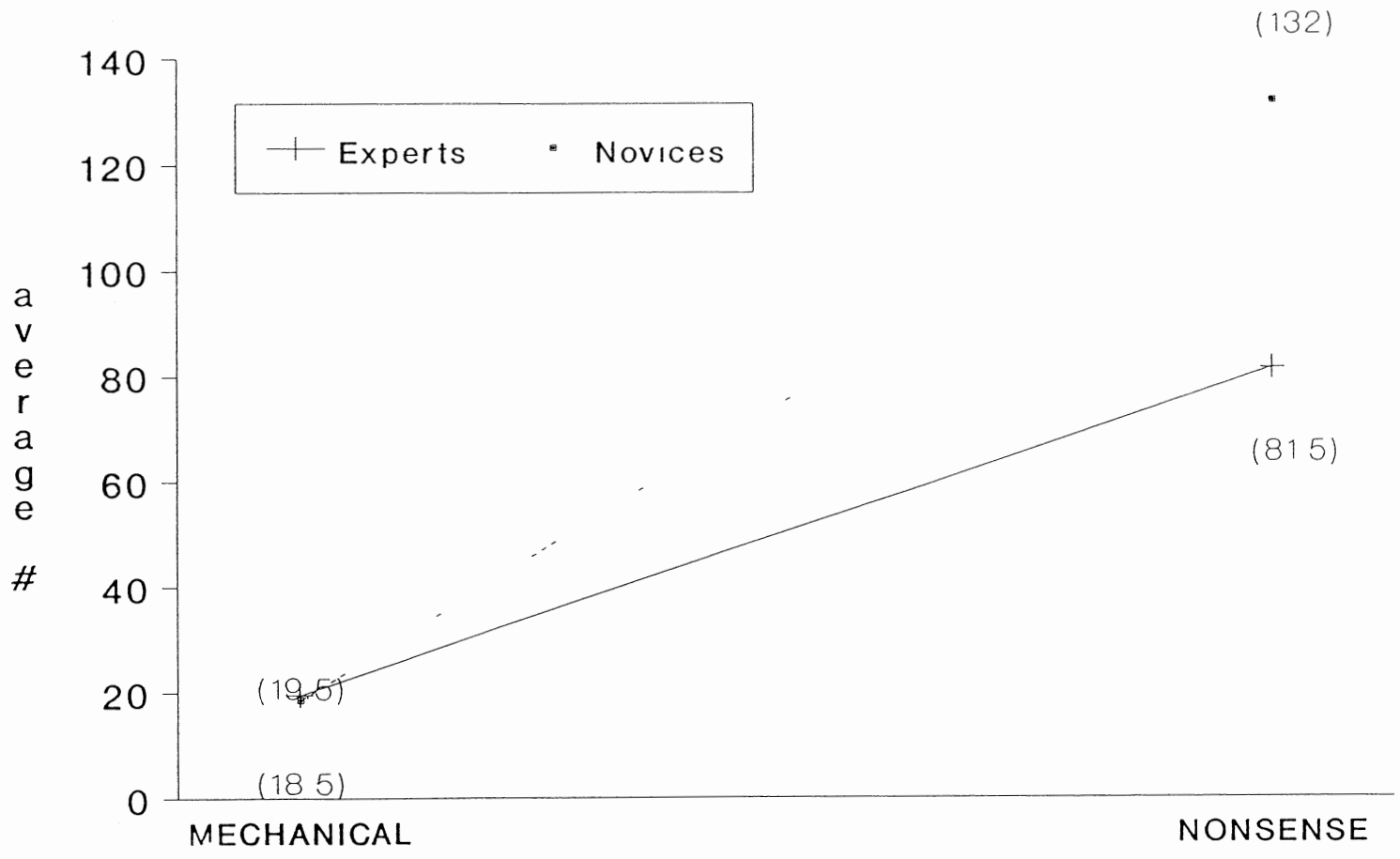


OBJECT TYPE

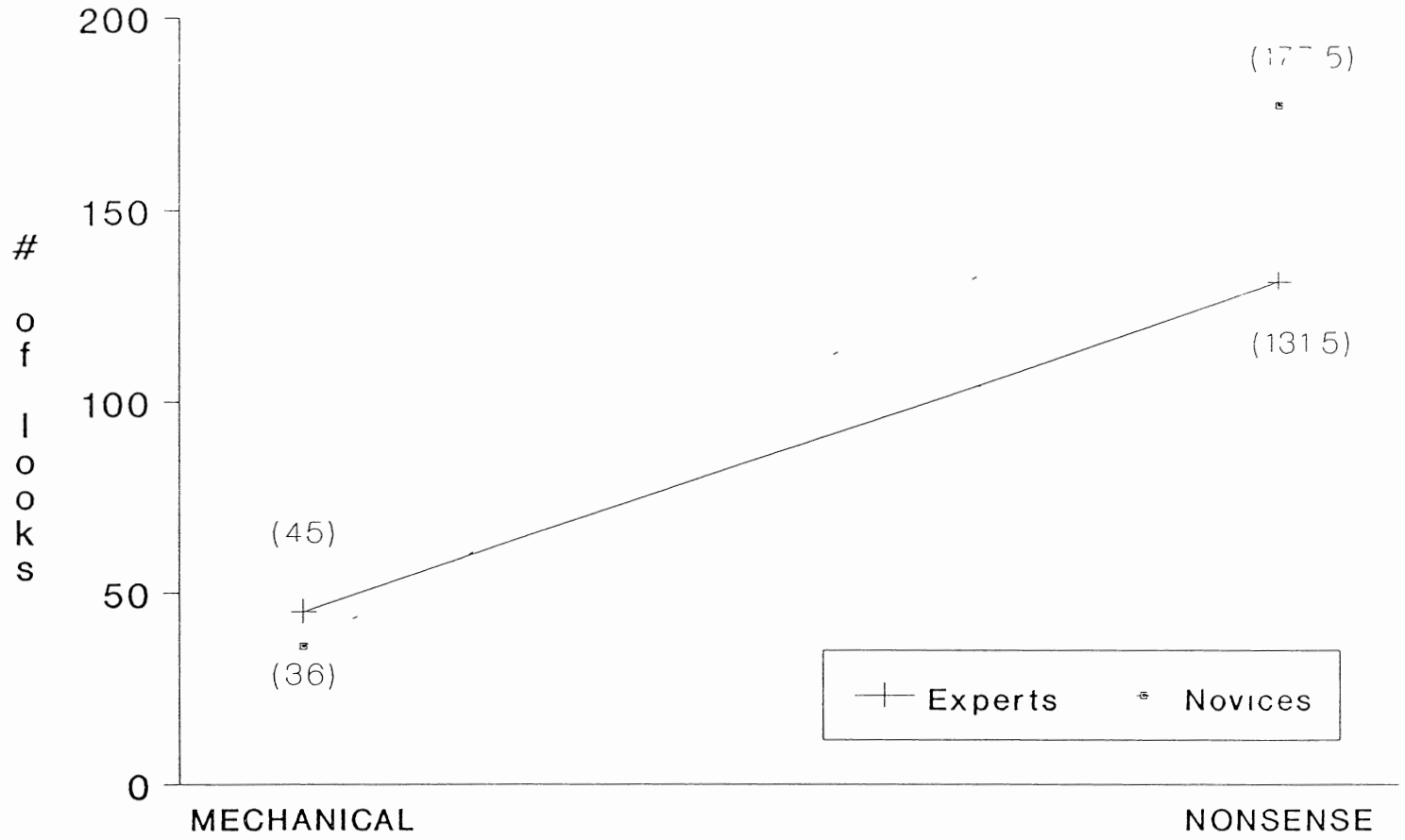
Average # of Disconnections

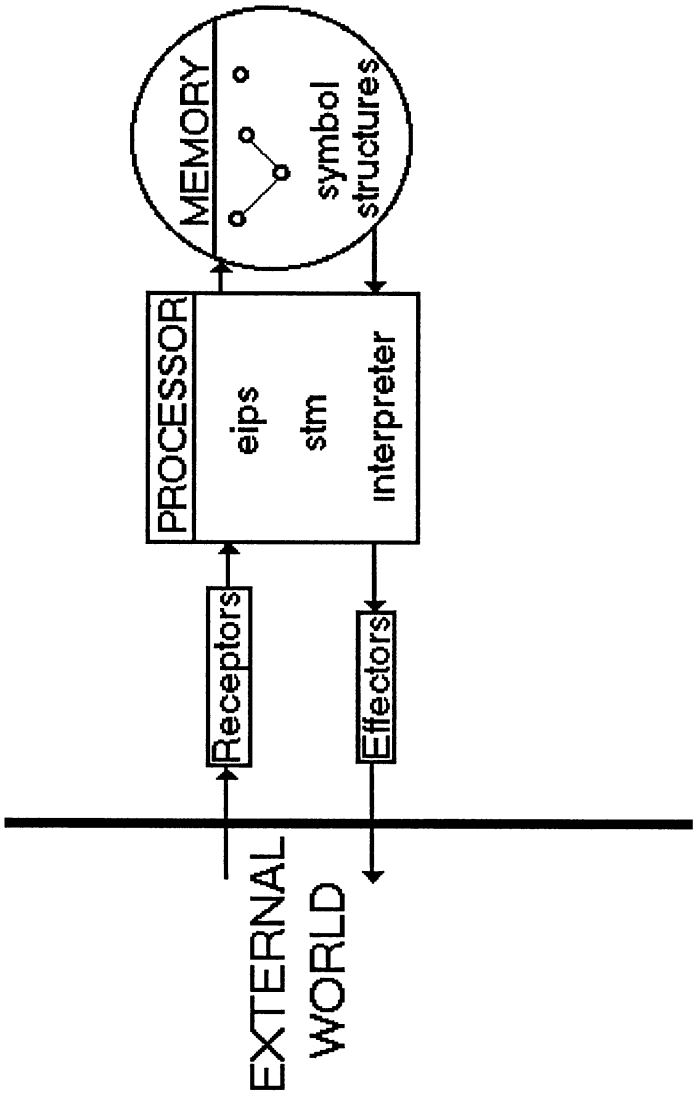


NUMBER OF ZERO TRIALS



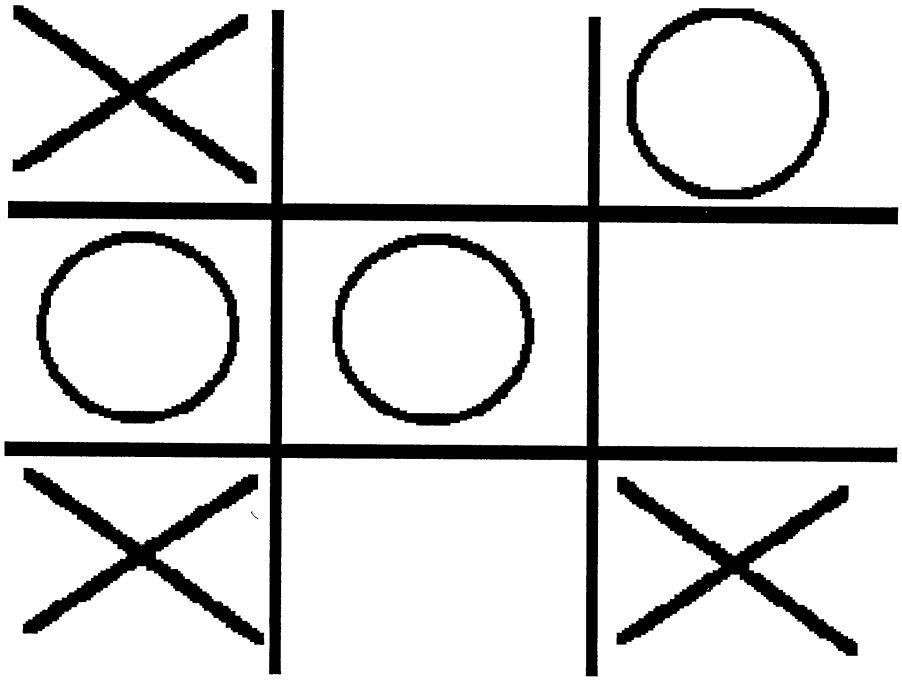
AVG # OF LOOKS


200
150
100
50
0
# of looks
(45)
(36)
(131 5)
Experts
Novices
MECHANICAL
NONSENSE
OBJECT TYPE


MEMORY
symbol structures
PROCESSOR
eips
stm
interpreter
Receptors
Effectors
EXTERNAL WORLD

```
1. observe-temperature,
    if < 70 degrees go to 2
    if > 72 degrees go to 4
   go to 1;

2. test if furnace-on
    if true go to 1;

3. turn-furnace-on
   go to 1;

4. test if furnace-on
    if false go to 1;

5. turn-furnace-off
   go to 1;
```

top



RATIONAL PROBLEM SOLVERS HYPOTHETICAL PROBLEM SPACE

```
IF (move=opponent)
THEN
   stop.

IF (own=two marks on a row)
   &(blank square on horizon)
THEN
   play blank square.

IF (own=two marks on column)
   &(blank square on column)
THEN
   play blank square.

IF (own=two marks on diag.)
   &(blank square on diag.)
THEN
   play blank square.

IF (opp=two marks on a row)
   &(blank square on horizon)
THEN
   play blank square.

IF (opp=two marks on column)
   &(blank square on column)
THEN
   play blank square.

IF (opp=two marks on diag.)
   &(blank square on diag.)
THEN
   play blank square.

IF (own=forking pattern)
   &(intersection blank)
THEN
   play intersection.

IF (opp=forking pattern)
   &(intersection blank)
THEN
   play intersection.

IF (center is blank)
THEN
   play center.

IF (opp=side square)
THEN
   play corner.

IF (opp=corner)
THEN
   play opposite corner.
```

PLAYER'S HYPOTHETICAL PROBLEM SPACE INFERRED FROM BEHAVIOR

```
IF (move=opponent)
THEN
   stop.

IF (opp=two marks on a row)
   &(blank square on horizon)
THEN
   play blank square.

IF (opp=two marks on column)
   &(blank square on column)
THEN
   play blank square.

IF (opp=two marks on diag.)
   &(blank square on diag.)
THEN
   play blank square.

IF (own=two marks on a row)
   &(blank square on horizon)
THEN
   play blank square.

IF (own=two marks on column)
   &(blank square on column)
THEN
   play blank square.

IF (own=two marks on diag.)
   &(blank square on diag.)
THEN
   play blank square.

IF (own=forking pattern)
   &(intersection blank)
THEN
   play intersection.

IF (opp=forking pattern)
   &(intersection blank)
THEN
   play intersection.

IF (center is blank)
THEN
   play center.

IF (opp=side square)
THEN
   play corner.

IF (opp=corner)
THEN
   play opposite corner.
```

VITA

Tim P. McCollum

Candidate for the Degree of

Doctor of Philosophy

Dissertation: COGNITIVE DIMENSIONS OF HUMAN PROBLEM SOLVING, INVENTION, AND CREATIVITY FROM CONVENTIONAL, CONNECTIONIST, AND INTEGRATED PERSPECTIVES

Major Field: Psychology

Biographical:

Personal Data: Born in Ponca City, OK, March 24, the son of O.L. and Eletha McCollum.

Education: Graduated from Shidler Senior High School, Shidler Oklahoma, in May 1977; received Bachelor of Science Degree in Psychology from Oklahoma State University in May 1982; completed Master of Science degree at Oklahoma State University in December 1983. Finished degree requirements for Doctor of Philosophy in May 1992 at Oklahoma State University.

Professional Experience: Peer Instructor, College of Arts and Sciences, Oklahoma State University, August 1978 - May 1982; Teaching Assistant, Department of Psychology, Oklahoma State University, August 1982 - December 1983; Teaching Assistant, Department of Psychology, Oklahoma State University, August 1985 - May 1986; Human Factors Scientist, IBM Santa Teresa Laboratories, San Jose, California, August 1986 - September 1987; Faculty, Department of Psychology, University of Texas - Pan American, Edinburg TX, August 1988 - May 1992.