PROBLEM SOLVING AND COGNITIVE SKILL AQUISITION

c

Technical Report AIP - 32

Kurt VanLehn

Departments of Psychology and Computer Science
Carnegie Mellon University
Pittsburgh, PA. 15232

The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

2

PROBLEM SOLVING AND COGNITIVE SKILL AQUISITION

Technical Report AIP - 32

Kurt VanLehn

Departments of Psychology and Computer Science
Carnegie Mellon University
Pittsburgh, PA. 15232

22 February 1988

I would like to thank John Anderson, Dirk Ruiz, Herbert Simon and especially Micki Chi for their comments. This work was supported by the Personnel and Training Research Program, Psychological Sciences Division, Office of Naval Research, under contract N00014-86-K-0349 and by the Information Sciences Division, DARPA, Office of Naval Research, under contract N00014-86-K-0678; Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) A I P -- 32	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, Pennsylvania 15213	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, Virginia 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS p4000ub201/7-4-86 PROGRAM ELEMENT NO: N/A; PROJECT NO.: N/A; TASK NO.: N/A; WORK UNIT ACCESSION NO: N/A

11 TITLE (Include Security Classification)
PROBLEM SOLVING AND COGNITIVE SKILL AQUISITION

12 PERSONAL AUTHOR(S) Kurt VanLehn

13a. TYPE OF REPORT	13b. TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 86Sept15 TO 91Sept14	1988 February 22	62

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Problem solving, Human skill acquisition, Cognitive modelling,

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

There has been a great deal of work in human problem solving since the landmark publication of Newell and Simon's Human Problem Solving in 1972. After reviewing the 1972 theory and the subsequent additions to it that model practice effects and expert-level problem solving, this tutorial presents 19 basic findings that seem to capture much of the recent experimental work in the field.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b TELEPHONE (Include Area Code) (202) 696-4302 / 22c. OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Table of Contents

List of Tables

Abstract

There has been a great deal of work in human problem solving since the landmark publication of Newell and Simon's *Human Problem Solving* in 1972. After reviewing the 1972 theory and the subsequent additions to it that model practice effects and expert-level problem solving, this tutorial presents 19 basic findings that seem to capture much of the recent experimental work in the field.

A brief history of the study of problem solving

Although virtually any human activity can be viewed as the solving of a problem, throughout the history of the field, most research has concerned tasks that take minutes or hours to perform. Typically, subjects make many observable actions during this period, and these actions are interpreted as the externally visible part of the solution process. Even if subjects are required to solve problems in their heads (e.g., to mentally multiply 135×76), they are usually asked to talk aloud as they work, and the resulting verbal protocol is interpreted as a sequence of actions (see chapter 1, this book). Thus, the tasks studied are not only long tasks, but also *multi-step* tasks.

The earliest experimental work on human problem solving was done by Gestalt psychologists, notably Kohler, Selz, Duncker, Luchins, Maier, and Katona. They concentrated on multi-step tasks where only a few of the steps to be taken were crucial and difficult. Such problems are called *insight problems* because the solution follows rapidly once the crucial steps have been made.[1] An example of such a task is construction of a wall-mounted candle-holder from an odd assortment of materials, including a candle and a box of tacks. The materials are chosen in such a way that the only solution involves using the box as a support for the candle by tacking it to the wall. To find this solution, subjects must change their belief that the box is only a container for the tacks and instead view the box as a construction material. This belief change is the crucial, insightful step. Once it is made, the solution is soon reached.

In contrast, most problem solving research in the last three decades has concerned multi-step tasks where no single step is the key. Rather, finding of a solution depends on making a number of correct steps. An example of such a task is solving an algebra equation. The solution is a sequence of proper algebraic transformations, correctly applied. The difficulty in the problem lies in deciding which transformations to apply, remembering them accurately, and applying them correctly. Thus, the responsiblity for the solution is spread over the whole solution process rather than falling on the discovery of one or two key steps. This choice of tasks caused research to focus on how people organize the solution process, how they decide what steps to make in what circumstances, and how their knowledge of the task domain determines their view of the problem and their discovery of its solution. These topics are the ones emphasized in this chapter.

In the 1950s and 1960s, most research concerned tasks that require no special training or background knowledge. Everything that the subject needs to know to perform the tasks is presented in the instructions. A classic example of such a task is the Tower of Hanoi. The subject is shown a row of three pegs. On the leftmost peg are three disks: a large one on the bottom, then a medium sized one, and a small disk on top. The subject is told that the goal of the puzzle is to move all three disks to the rightmost peg, but only one disk may be moved at a time and a larger disk may never be placed on top of a smaller one. There are many variations of this basic puzzle. For instance, there can be more disks than three, and the starting and finishing states can be arbitary configurations of disks. All these variants of the puzzle are called a *task domain* and each specific version is called a *task*, that is, one element of the task domain. "Task" and "problem" are virtually synonomous. The Tower of Hanoi, and other simple, puzzle-like task domains are called *knowledge-lean*, because it takes very little knowledge (i.e., just what one reads in the instructions) in order to solve problems in the task domain. Of course, some subjects may have a great deal of knowledge about the task domain -- puzzle fanantics, for instance. However, possesion of such knowledge is not essential for obtaining a solution. Someone with very little knowledge can blunder through to a solution.

The study of knowledge-lean tasks led to the formulation of Newell and Simon's landmark theory. Their 1972 book, *Human Problem Solving*, is still required reading for anyone seriously interested in the field. This theory became the foundation for many detailed models of problem solving in specific task domains. The models are able to explain not only the steps taken by the subjects, but also their verbal comments (e.g., Newell and Simon, 1972, chapters 6, 9 and 12), the latencies between steps (e.g., Karat, 1982), and even their eye movements (e.g., Newell and Simon, 1972, chapter 7). The early seventies marked a high point for theoretical work in the field of knowledge lean problem solving.

In the late 1970's, attention shifted to studying *knowledge-rich* task domains, which are task domains where many pages of instructions are required for presenting even the minimal knowledge necessary for solving the problem. Knowledge-rich task domains that have been studied include algebra, physics, thermodynamics, chess, bridge, geometry, medical diagnosis, public policy formation and computer programming.

Much early empirical research into knowledge-rich tasks concerned the differences between experts and novices. Varying the level of expertise while holding the task domain constant helped investigators separate the effects of expertise from the influence of the task domain. The typical study gave the same set of problems to experts and novices and used protocol analysis (see chapter 1, this book) to examine differences in the performance of the two groups. Of course, the novices found the problems quite hard and the experts found them quite easy. If one assumes that the experts had encounter the same or similar problems many times in the past, one would expect them to simply recognize the problem as an instance of a familiar problem type, retrieve the solution template from memory, and generate the problem's solution directly. Novices, on the other hand, might have no such knowledge, so they would have to blunder about, searching for a solution, just as the subjects in the knowledge-lean task domains do. To put it briefly, the hypothesis is that expertise allows one to substitute recognition for search.

Although the mid-70s saw development of computer programs that could model the steps, latencies and even eye movements of subjects solving puzzles, no such models have been developed for experts solving problems in knowledge rich task domains. Partly, this is because it has proved difficult to build computer programs that contain a great deal of knowledge, and only recently has the technology for building such expert systems begun to bear fruit. There is a small but increasing number of programs that can competently solve problems in knowledge-rich task domains, although they often resort to methods that human experts do not seem to use (e.g., extensive combinatorial searches).

However, there are also scientific reasons for *not* just building an expert system as a model of expert problem solving. Expert behavior, whether generated by people or programs, is a product of their knowledge, so any explanation of that behavior must rest on postulating a certain base of knowlege. But what explains that knowledge? Although it could be measured or formally constrained in various ways, the ultimate explanation for the form and content of the human experts' knowledge is the learning processes that they went through in obtaining it. Thus, the best theory of expert problem solving is a theory of learning. Indeed, learning theories may be the only scientifically adequate theories of expert problem solving.

Thus, the focus of attention in the 1980's has been on the acquisition of expertise. There has been a revival of interest in traditional topics in skill acquisition, such as practice effects and transfer, and many of the regularities demonstrated with perceptual-motor skills have been found to govern cognitive skills as well. There are also novel experimental paradigms. For instance, much has been learned from taking protocols of students as they learn. A number of learning mechanisms have been developed, and will be discussed later, but we still have incomplete knowledge about their respective roles in the total picture of cognitive skill acquisition.

The organization of this chapter

Because this is a time of transition, a coherent theory of problem solving and skill acquisition cannot be presented here, so the ingredients for developing such a theory are presented instead. First, the 15-year old theory of Newell and Simon is presented using as illustrations the knowledge-lean task domains that are its forte. Second, the idea of a *schema* is introduced because it has played an important role in explaining the long-term memory structures of experts. Third, a list of major empirical findings is presented.

Knowledge-lean problem solving

This section discusses a theory of problem solving that was introduced by Newell and Simon in *Human Problem Solving* and has come to dominate the field. It forms a framework or set of terms that have proved useful for constructing specific analyses and models of human cognition.

The theory begins by making idealizations that distinguish between types of cognition. These distinctions are often difficult to define in objectively measurable terms. For instance, the first idealization is to distinguish between problem solving that involves learning, and problem solving that does not. "Learning", in this context, means resilient changes in the subject's knowledge about the task domain that are potentially useful in solving further problems (see Simon, 1983, for a discussion of this definition of learning). Early work (e.g., Newell and Simon, 1972) assumed that there is little or no learning during problem solving. This idealization allowed formulation of an theory that still useful today. Moreover, it provided the foundation for accounts of problem solving *with* learning. As usual in science, oversimplification is not necessarily a bad thing.

The first several subsections of this section will present a discussion of problem solving under the idealization that learning is not taking place. In the last subsection, the oversimplification will be ammended and learning mechanisms will be discussed.

Problem solving = understanding + searching

A second important idealization is that the overall problem solving process can be analyzed as two cooperating subprocesses, called *understanding* and *search*. The understanding process is responsible for assimilating the stimulus that poses the problem and for producing mental information structures that constitute the person's understanding of the problem. The search process is driven by these products of the understanding process, rather than the problem stimulus itself. The search process is responsible for finding or calculating the solution to the problem. To put it differently, the understanding process generates the person's internal representation of the *problem*, while the search process generates the person's *solution*.

It is tempting to think that the understanding process runs first, produces its product, and then the search process begins. However, the two processes often alternate or even blend together (Hayes & Simon, 1974; Chi, Glaser & Rees, 1982). If the problem is presented as text, then one may see the solver read the problem (the understand process), make a few moves toward the solution (the search process), then reread the problem (understanding again). Although some understanding is logically necessary before search can begin, and indeed most understanding does seem to occur towards the begining of the problem solving session, it is not safe to assume that understanding always runs to completion before search begins.

The first subsection will discuss the understanding process, and the second will discuss the search process. A third, brief subsection discusses a common type of problem solving that has some of the characteristics of both understanding and search.

The understanding process in knowledge-lean task domains

The understanding process converts the problem stimuli into the initial information needed by the search process. The early stages of the understanding process depend strongly on the media in which the problem is presented: text or speech, diagrams or pictures, physical situations or imaginary ones. Presumably, a variety of perceptual processes can be involved in the early stages of understanding. Because perceptual processes are studied in other fields of cognitive science, problem solving research has concentrated on describing the later stages of understanding, and in particular, on specifying what the output of the understanding process is.

In knowledge-lean task domains, there is wide-spread agreement on what the product of understanding is. It follows, almostly logically, from constraints on the type of material being understood. By definition, the instuctions for a problem in a knowledge-lean task domain contain all the information needed for solving the problem, although it may have to be supplemented and interpreted by common sense knowledge. When this definition is combined with the fact that problem solving tasks are, almost by definition, multi-step tasks, then it follows that the minimal information that the subject needs to obtain from the problem instructions consists of three components: (1) the initial problem state, (2) some operators that can change a problem state into a new state, and (3) some efficient test for whether a problem state constitutes a solution. These three components, along with some others that can be derived from them, are collectively called a *problem space*. Thus, a major assumption about the understanding process for knowledge-lean task domains is that it yields a problem space.

The name "problem space" comes from the fact that the conjunction of an initial state and a set of operators logically implies a whole space of states (i.e., a state space). Each state can be reached from the initial state by some sequence of operator applications. An incontestable principle of cognition is that people are not necessarily aware of all the deductive consequence of their beliefs, and this principle applies to problem spaces as well. Although the state space is a deductive consequence of the initial state and the operators, people will not be aware of all of it. For instance, a puzzle solver may not be able to accurately estimate the number of states in the state space even after solving the puzzle several times. On the other hand, the size and topology of the state space has played an important role in theoretical analyses where, for instance, the difficulty of a problem is correlated with the topology of the state space (Newell & Simon, 1972).

As an illustration of the concept of a problem space, the problem spaces of two subjects will be compared. Both subjects heard the following instructions:

> Three men want to cross a river. They find a boat, but it is a very small boat. It will only hold 200 pounds. The men are named Large, Medium and Small. Large weights 200 pounds, Medium weights 120 pounds, and Small weights 80 pounds. How can they all get across? They might have to make several trips in the boat.

One subject was a nine-year old girl, who asked me to refer to her as "Cathy" in describing her performance. Upon hearing the instructions, Cathy immediately asked "The boat can hold only 200 pounds?", and the experimenter answered affirmatively. Thereafter, almost all of Cathy's discussion of the puzzle used only "sail" as a main verb and "Large", "Medium", "Small", "the boat" and pronouns as noun phrases. (Cathy's complete protocol will appear later in table 3.) It is apparent from the protocol that Cathy solves this problem by imagining the physical situation and the actions taken in it, as opposed, say, to converting the puzzle to a directed graph then finding a traversal of the graph. Thus, we can formally represent her belief about the current state of the imagined physical world as a set of propositions of the form `(ON X Y)` where `X` is in the set `{L, M, S, B}` and `Y` is in the set `{Source, Destination}`. For instance, the set of propositions,

`(On L Source) (On M Source) (On S Source) (On B Source)`

represents the initial situation, where Large, Medium, Small and the boat are all on the source bank of the river. Notice that Cathy could have a much richer description of the situation in mind, that includes, for instance, propositions describing how much weight is in the boat and on each of the banks. However, such descriptions never appear in her protocol, so it can be assumed (justified by simplicity and parsimony, and subject to refutation by further experiments) that Cathy maintains *only* descriptions of the `(On X Y)` type while she solves the puzzle.

Similarly, we can ask what types of operators Cathy believes are permitted in solving this puzzle. Apparently, she infers it is not permitted that the three men can swim across the river or take some other transportation than the boat. Moreover, she must have inferred that the 200 pound limit implies that only certain combinations of passengers are possible, because she only mentions legal boat rides. Thus, Cathy seems to have just one legal operator, which can be formally represented as `(Sail X Y Z)`, which stands for sailing passenger set `X` from bank `Y` to bank `Z`. The argument `X` is either `{L}`, `{M,S}`, `{M}` or `{S}`, and `Y` and `Z` are either `Source` or `Destination`.

Cathy immediately recognizes when she has reached the desired final state, and moreover, she shows signs throughout the protocol of being aware of it. So we can safely assume that Cathy's understanding of the LMS puzzle contains at least an initial state, the `Sail` operator, and the desired final state.

It is clear that Cathy's problem state is a very coarse representation of the actually physical situation of some men and a boat. Apparently she does not believe that the river's current, the weight of a boatload, and other factors are relevant to solving this puzzle. In order to highlight the subject's beliefs about what aspects of the puzzle's situations are relevant, most definitions of "problem space" (e.g., Newell and Simon, 1972) specify a fourth component, a *state representation language*. Every state in the problem space, including the initial and final states, should be representable as some expression in this formal language. The state representation language in

Cathy's case is simply all possible conjunctions of (On X Y) propositions.

It is important to note that not all subjects derive the same problem space from the instructions. For instance, another subject, a 60-year old adult male, first understood the instructions given to Cathy as an arithmetic problem. After hearing the instructions, the subject immediately answered that it would take two trips, because only 200 hundred pounds could be moved per trip, and there were 400 pounds of men to move. He generated a different problem space from Cathy's, even though he received the same instructions. He was asked to described exactly what those two trips were. He indicated that first Large could row across, then Medium and Small. The experimenter asked him how the boat was gotten back across. The subject replied that there must be a system of ropes or something. The experimenter asked him to assume instead that someone would have to row the boat back. This added instruction caused the subject to change his problem space. His new problem space was similar to Cathy's. This second subject's behavior shows that the understanding of simple knowledge-lean puzzles can interact with common-sense knowledge in interesting and non-obvious ways, and can proceed differently with different subjects.

The above example also shows that subjects can change their problem space to accomodate added information from the experimenter. Sometimes, information garnered by the subjects themselves in the course of problem solving will also cause them to change their problem space. Some investigators (Duncker, 1945; Ohlsson, 1984) hypothesize that the "insights" of subjects solving insight problems are often changes of problem spaces.

There are problems that do not fit neatly into the problem space mold, mostly because the solution states are not well defined. For instance, one can ask a subject to draw a pretty picture. Although minimal competence in this task requires no special knowledge, and therefore the task domain qualifies as a knowledge-lean one, it is difficult to characterize the subject's test for the final state. Indeed, it is likely that some subjects themselves may not know what the final state will be until the picture is half-drawn. In these cases, finding a set of constraints that qualify a problem state as a solution is just as imporant as generating a solution state. For knowledge-lean task domains, a *well-defined* problem is defined to be one where the subject's understanding of the problem produces a problem space, that is, an initial state, a set of operators, and a solution state description. Problems whose understanding is not readily represented as a problem space are called *ill-defined* problems. Sketching pretty pictures is an ill-defined problem. A definition of "well-defined" for knowledge-rich task domains would be equivalent in spirit, but is not so easily stated because the understanding process for knowledge-rich task domains is considerably more complicated. There have been only a few studies of ill-defined problem solving. Reitman (1965) studied a composer writing a fugue for piano. Akin (1980) studied architectural design. Voss and his colleagues have studied agricultural policy formulation (Voss, Greene, Post & Penner, 1983; Voss, Tyler & Yengo, 1983). Simon (1973) provides a general discussion of ill-defined problem solving. This chapter concentrates exclusively on well-defined problems, since that is where most of the research has been focused.

Although the output of the understanding process in knowledge-lean task domains is well-understood (albeit, by fiat), less is known about the process itself. In part, this is because the understanding process for typical puzzles takes very little time. Cathy's protocol was two minutes long, but the understanding process seems to have run to completion during the first 20 seconds. The only behavior to observe during that brief time was Cathy's posing a question to the experimenter. In order to magnify the understanding process, Hayes and Simon (1974) studied a puzzle, called the tea ceremony, whose instructions are quite difficult to understand:

> In the inns of certain Himalayan villages is practiced a most civilized and refined tea ceremony. The ceremony involves a host and exactly two guests, neither more nor less. When his guests have arrived and have seated themselves at his table, the host performs five services for them. These services are listed in the order of the nobility which the Himalayans attibute to them: (1) Stoking the Fire, (2) Fanning the Flames, (3) Passing the Rice Cakes, (4) Pouring the Tea, and (5) Reciting Poetry. During the ceremony, any of those present may ask another, "Honored Sir, may I perform this onerous task for you?" However, a person may request of another only the least noble of the tasks which the other is performing. Further, if a person is performing any tasks, then he may not request a task which is nobler than the least noble task he is already performing. Custom requires that by the time the tea ceremony is over, all the tasks will have been transfered from the host to the most senior of the guests. How may this be accomplished?

Hayes and Simon took a protocol of a subject interpreting these instructions and solving the puzzle. The subject read the text many times before he began to solve the puzzle. From the protocol, it appears that the subject first built up an understanding of the objects in the initial state, then of the relationships between the objects, and finally of the legal operators. The subject proceeded statement by statement, trying to reconcile each statement with his current understanding.

The subject's major problem lay in interpreting the sentence, "During the ceremony, any of those present may ask of another, 'Honored Sir, may I perform this onerous task for you?'" The correct interpretation of this sentence, which the subject eventually discovered, is that the responsibility and ownership of the onerous task is transfered from one person to another. However, the subject's initial interpretation of the sentence was that one person is asking to do the task for the benefit of the other without actually relieving the other of the responsibility and ownership of the task. This benefactive reading is arguably the default interpretation for the English "perform for" construction, so it is no surprise that the subject's initial interpretation was benefactive. He only changed his interpetation when he noticed that the desired solution state requires that ownership of the onerous tasks have been transfered, and yet he has no operator that will effect such transfers. In order to make the problem solvable, he re-examines his interpretation of the "perform for" sentence, and discovers its other reading.

This study and others (Hayes & Simon, 1976; Kotovsky, Hayes & Simon, 1985) convinced Hayes and Simon that understanding of well-defined problems in knowledge-lean task domains is a rather direct translation process whose character is determined mostly by the type of stimulus used and the need for an internally consistent initial problem space. As will be seen later, this is not an apt characterization of the understanding process in knowledge-rich domains, nor does it explain why different subjects sometimes generate different problem spaces from the same instructions.

The search process

Suppose that problem spaces had not yet been invented, and we set out to formally describe the process of searching for problem solutions. We would soon discover that it is often quite easy to represent the subjects' current assumptions, postulations or beliefs about the problem as a small set of assertions. For example, in the midst of trying to extrapolate the sequence "ABMCDM," the subject might have the beliefs that the sequence has a period of three and the third element of the period is always "M." Thus, the subject's current beliefs could be notated formally as including the assertions `Period = 3` and `For all p, Third(p)= "M"`, where `p` indexes periods. The search process consists of small, incremental changes in the subject's beliefs that can be modelled as small changes to the set of assertions. For instance, the next step in the search for the pattern of ABMCDM might produce just one new assertion about the problem, say, that the first and second elements in a period are consecutive letters in the alphabet. (Put formally, the new assertion is `For all p, Second(p) = Next(First(p))`.) This formal description of the problem solving process, as a sequence of incremental changes to a set of assertions, is exactly the same as the problem space notation. A state in a problem space corresponds to a set of assertions. The applicatation of an operator to a state corresponds to the incremental changes in the subject's set of assertions. The operators themselves correspond to the heuristic rules that the subject uses to modify assertions (e.g., "if the same letter occupies both positions `i` and `i+x`, then assert that `Period = x`"). This demonstrates the naturalness of problem spaces as a formal notation for the behavior that subjects exhibit while problem solving.[2]

The assertions that populate a problem state can represent beliefs that arise directly from perception. For instance, if the subject sees that the leftmost peg of the Tower of Hanoi puzzle has no disks on it at this time, then one could include the assertion `Disks(leftmost-peg)={}` in the set that represents the subject's beliefs. Similarly, moving a disk can be represented as an incremental change in the set of assertions. Thus, the problem space framework serves both to represent changes of the subject's internal state was well as changes in the physical state of the world.[3]

For most problem spaces, there are usually several operators that can be applied to any given state. For instance, instead of infering that `Second(p) = Next(First(p))`, which relates A with B and C with D in ABMCDM, it could be infered that `First(p+1)=Next(Second(p))`, which relates B with C. In this case, it does not matter which operator is chosen. However, some operator applications lead to dead ends. For instance, if it is decided that the period of the sequence "defgefghfghi" is 3, then a correct solution cannot be found by adding more assertions to the resulting state, because the correct period is actually 4. These facts -- that multiple operators apply at most states, and that some sequences of operator applications lead to dead ends -- follow logically from the definition of the problem space. Any intelligence, human or artificial, must cope with these facts in order to find a solution path.

Suppose it is assumed that only one operator can be applied at a time and that an operator can only be applied to an *available* state, where a state is available only if (1) it is mentioned in statement of the problem or (2) it has been generated by application of an operator to an available state.[4] These assumptions logically imply that any solution process must be a special case of the algorithm template shown in table 1. The "slots" in this template are the functions for choosing a state, choosing an operator and pruning states from the set of active states. A variety of specific algorithms can be formed by instantiating these slots with specific computations. The class of algorithms formed this way are called *state space search* algorithms. Much work has been done on the properties of these algorithms.[5]

Let active-states be a set of states, which initially contains only the states mentioned in the problem statement.

1. Choose a state from active-states. If there are no states left in active-states, then stop and report failure.

2. Choose an operator that can be applied to the state. If no operator applies, then go to step 5.

3. Apply the operator to the state, producing a set of new states. If the set is empty, go to step 5.

4. Test whether any of the new states is a desired final state. If one is, then stop and report success. If none are, then place them in active-states and go to step 5.

5. Choose a subset of the states in active-states, and remove them from active-states. Go to step 1.

Table 1: A general search procedure

Although the search algorithm template of table 1 is simple, it does not have quite the right structure for describing human problem solving. People seem to distinguish between new states and old states, where a new state is one produced by the most recent operator application. In selecting a state (step 1 of the algorithm), choosing a new state is viewed as proceeding along the current path in the search, while choosing an old state is viewed as failing and backing up. For people, different principles of operation seem to apply to these two kinds of selections. In order to capture this distinction, the work of search can be allocated among two collaborating processes:

1. A process, called the *backup strategy*, that maintains the set of old states, and chooses one when necessary.
2. A process, called the *proceed strategy*, that (1) chooses an operator to apply to the current state, (2) applies it, and (3) evaluates the resulting states. If one of them is a desired, final state, the search stops and reports success. On the other hand, if none of them seem worth pursuing, then the backup strategy is given control. Otherwise, this process repeats.

Although this algorithm template is logically equivalent to the one of table 1, it has different slots, namely, one for the backup strategy and one for the proceed strategy (the latter is not a standard term in the field, but it should be).

Both the backup strategy and the proceed strategy are viewed as potentially nondeterministic procedures, in that there are a number of choice points (e.g., choosing a operator) where the procedure does not specify how the choice is to be made. However, some subjects seem to apply simple, efficient criteria, called *heuristics*, to narrow the set of choices. Sometimes the heuristics are so selective that they narrow the options to just a single, unambiguous choice. In short, this general template for search algorithms has three slots: (1) the backup strategy, (2) the proceed strategy, and (3) heuristics for the backup and proceed strategies.

It is generally held that there are a handful of distinct *weak methods* that novice subjects use for knowledge-lean task domains (Newell & Simon, 1972; Newell, 1980; Laird, Newell, & Rosenbloom, 1987). Most of these methods are proceed strategies. The simplest weak method is a proceed strategy called *forward chaining*. Search starts with the initial state. Heuristics are used to select an operator from among those that are applicable to the current state. The selected operator is applied, and the strategy repeats. Another strategy, called *backwards chaining*, can be used only when a solution state is specific and the operators are invertible; it starts at the solution state, heuristically chooses an operator to apply, and applies it inversely. Thus, it builds a solution path from the final state towards the initial state. A third strategy is *operator subgoaling*. It heuristically chooses an operator without paying attention to whether that operator can be applied to the current state. If the operator turns out to be inapplicable because some condition that the operator requires (such conditions are called *preconditions*) is not met, then a subgoal is formed, which is to find a way to change the current state so that the preconditions are true. The strategy recurses, using the new subgoal as if it were the solution state specified by the problem space.[6]

As indicated above, all these strategies may usefully incorporate *heuristics* (rules of thumb) in order to narrow the guesswork. Often, heuristics are specific to the particular task domain. However, a particularly general heuristic is based on having the ability to simply calculate the difference between a state and the description of a desired state. If states are notated as sets of assertions, then set difference can be used to calculate inter-state differences. The *difference reduction* heuristic is simply to choose operators such that the differences between the current state and the desired state are maximally reduced.[7]

There is a very general method, called *means-ends analysis*, that is so widely used that is worth examining in some detail. Table 2 shows the basic strategy. It subsumes two common strategies: forward chaining and operator subgoaling. For instance, if there are never any unsatisfied preconditions in step 3 of table 2, the method will do forward chaining. Thus, means-ends analysis is a generalization several other weak methods. (Such incestuous relationships

```
Let State hold the current state, and Desired hold a description of the
desired state.  Let Goal and Op be temporary variables.

1.  Calculate the differences between State and Desired.
      If there are no differences, then succeed.
      Otherwise, set the differences into Goal.
2.  See which operators will reduce the differences in Goal.
      If there are none, then fail.
      Otherwise, use heuristics to select one, and set it into Op.
3.  Calculate the differences between State and the preconditions of Op.
      If there are any, set Goal to the differences, and go to step 2.
      Otherwise, apply Op to State, and update State accordingly.
4.  Use heuristics to evaluate State.
      If it seems likely to lead to Desired, then go to step 1.
      Otherwise, fail.
```

Table 2: The method of means-ends analysis

among weak methods makes it difficult to give crisp definitions, so the terminology is rather fluid. Indeed, some authors would take issue with the definitions given in this chapter.)

Table 3 shows means-ends analysis as a model for a Cathy solving the LMS puzzle, which was discussed earlier. Note that the heuristics used in this task mention specific information in the task, such as men and river banks. This is typical. The heuristics are task-specific while the methods are general. Note also that means-ends analysis does not specify what happens when a failure occurs. It is only a proceed strategy and not a backup strategy. However, means-ends analysis alone suffices to model Cathy's behavior, because she never backs up during the solution of this puzzle.

Backup strategies are determined mostly by the types of memory available for storing old states. If external memory is used, such as a piece of scratch paper, then more old states may be available than when only internal memory is used. Also, some tasks place physical constraints on backup strategies. For instance, there are puzzles, such as the eight-puzzle, Rubik's cube, or the Chinese Ring puzzle, where the goal is to rearrange the puzzle's parts into a certain configuration. However, the parts are constructed so only some kinds of moves are physically possible. Thus, one cannot backup to arbitrary states, even if one writes them down.

Problem space:

(1) A state is a pair consisting of two sets, representing the contents of the source and destination banks, respectively. Both sets are subsets of {L,M,S,B}, which stand for Large, Medium, Small and the Boat. The union of the two sets is {L,M,S,B}. (2) There is only one operator, Sail. It takes a set of men and a bank as arguments. It only applies if the men are only the bank, and if their weight sums to 200 or less. It has a precondition that the boat be on the bank. (3)The initial state is LMS on the source bank. (4) The final state is that LMS be on the destination bank.

Heuristics:

(1) choose an opertor that will maximize the number of men on the destination bank. (2) Choose an operator that will maximize the weigth of the men on the destination bank.

line number		Protocol	simulation
0			Goal = LMS on destination bank
1	S:	The boat can only hold 200 pounds?	Op = ...
2	E:	The boat can only hold 200 pounds.	
3	S:	Okay...first...	
4		Small and medium go back,	Op = Sail MS to destination bank
5	E:	Uh-huh.	
6	S:	...go across the river on it.	Apply Op
7		and then, um, ... Oh	Goal = L on destination bank
8		Large... /3 second pause/	Op = Sail L to destination bank
9	E:	Yeah, go on... talk out loud.	
10	S:	... and... um...	
11		Large... um... /3 sec. pause/	
12	E:	Talk out loud.	
13		Tell me everything you're thinking.	
14	S:	But, I can't do it	Goal = Boat on source bank
15		because someone has to sail the boat back.	
15	E:	Ok... That's right.	
16		Somebody has to sail the boat back.	
18	S:	Oh! Ok... so... /4 sec. pause/	Op = Sail S to source bank
19		Small sails the boat back	Apply Op
20			Goal = LS on destination bank
21		and gets off,	Op = Sail L to destination bank
22		and lets Large sail the boat back.	Apply Op
23	E:	Um-hmm. And then what happens.	Goal = S on destination bank
24	S:	Uh... /3 sec. pause/	Op = Sail S to destination bank
25	E:	Talk out loud...	
26	S:	And then small...	
27		small...	Goal = boat on source bank
28		can't think of anything...	
29	E:	Keep talking.	Op = Sail M to destination bank
30	S:	So... Medium... sails back.	Apply Op
31		and...	Goal = MS on destination bank
32			Op = Sail MS to destination bank
33		Medium and small sail back.	Apply Op
34	E:	Keep talking.	
35	S:	And they're all across!	
36	E:	Very good!	

Table 3: Protocol and simulation of Cathy solving the LMS puzzle

Elaboration: search or understanding?

There is a special type of problem solving that deserves some extra discussion because it blurs the distinction between understanding and search. It concerns a certain class of beliefs, called *elaborations*, that subjects often develop about problems. Suppose, as usual, that the subjects' current beliefs about a problem are viewed as a set of assertions. As they work on the problem, they could add new assertions, take old ones away, or modify old assertions. They could even add assertions that, while not causing any of the old assertions to be removed, cause them to become irrelevant to subsequent problem solving. An elaboration is an assertion that is added to the the state without removing any of the old assertions or decreasing their potential relevance. As an illustration, consider the following problem:

Al is bigger than Carl. Bob is smaller than Carl. Who is smallest?

Such problems are called series problems (see Ohlsson, 1987, for a recent model of problem solving in this task domain and an introduction to the rather large literature on series problems). Suppose a subject reads this problem and immediately says "I guess it has to be one of the three of them." The subject apparently had some initial understanding of the problem, which could be modelled as a set of assertions. This statement indicates a reasoning process of some kind has run, producing a new assertion. The new assertion qualifies as an elaboration because it does not negate, remove or obviate any of the older assertions.

It is not clear what kind of reasoning produced this elaboration. On the one hand, the subject's behavior seems similar to the behavior of Cathy, who understood the LMS puzzle by assuming that the only transportation was a boat. This similarity suggests that the elaboration is a product of the understanding process. However, suppose the subject's next statement is "It can't be Carl, because Bob is smaller." This inference also qualifies as an elaboration. Indeed, there is nothing to distinguish it formally from the earlier elaboration. However, it is clear that the subject could go on to find a solution of the puzzle by making only elaborations of this sort. If all of them are considered to be products of understanding instead of operator application, then it follows that this problem can be solved by just understanding it. Search is not needed.

Clearly, elaborations can be classified either as part of the understanding process or as part of the search process. This might seem like a pointless terminological quibble. However, the search process is currently better understood than the understanding process. If elaboration is classified as search, then it inherits hypotheses (e.g., means-ends analysis, the paucity of backup) that might shed light on its organization and occurrence. Whether these hypotheses hold for elaboration remains to be seen.

Learning during problem solving

If subjects are given a knowledge-lean task, their initial performance may be stumbling and slow, but improve rapidly with practice. Mechanisms of practice-driven learning may be needed in order to give a sufficient explanation of such behavior. Several mechanisms have been proposed. Although this is a part of the field that is developing rather rapidly at present, its importance makes it worthwhile to describe some of the more widely known mechanisms. The mechanisms need not be used exclusively, but may be combined, and thus account for more phenomena that each can explain individually.

Compounding is a process that takes two operators in the problem space and combines them to form a new operator, often called a macro-operator (Fikes, Hart & Nilsson, 1972). Macro-operators are just operators, so they can be compounded with other operators to form even larger operators. As an illustration, suppose that a subject's algebra equation solving problem space originally has an operator for subtracting a constant from both sides of the equation, and a second operator for performing arithmetic simplifications. The following lines shows an application of each operator:

```
3x+5=20
3x=20-5
3x=15
```

Compounding can create a macro-operator that would produce the third line directly from the first line. When there are preconditions or heuristics associated with operators, then some bookkeeping is necessary in order to create the appropriate preconditions and heuristics for the macro-operator. This is easiest to see when operators are notated as productions so that the preconditions and heuristics appear in the condition of the operator's production. The two algebra operators can be represented as:

If "+ <constant>" is on the left side of the equation,
then delete it and put "- <constant>" on the right side of the equation.

If "<constant1> <arithmetic operation> <constant2>" is in the equation,
then replace it with "<constant3>", where <constant3> is...etc.

The second production's condition cannot be added verbatim to the macro-productions left side, because it would not be true at the time the macro-production should be applied. Thus, the correct formulation of the macro-production is:

If "+ <constant1>" is on the left side of the equation,
  and "<constant2>" is on the right side of the equation,
then delete both and put "<constant3>" on the right side,
  where <constant3> is ...etc.

This demonstrates that compounding is not always a trivial process. Fikes, Hart and Nilsson (1972) give a general algorithm. Lewis 1981) and Anderson (1982) have investigated the special case of production compounding.

As mentioned earlier, heuristics are often used in deciding which operator to select while moving forward. *Tuning* is the process of modifying the operator selection heuristics. Suppose for the sake of illustration that there are two applicable operators, A and B, in a certain situation. The heuristic conditions associated with A are false, say, so A is deemed a poor choice in this situation. The heuristics associated with B are true, which makes it a good choice, so it is selected. Suppose that the application of B leads immediately to failure, so backup retreats, A is chosen instead, and success occurs immediately. Obviously, the two heuristics gave poor advice, so they should be tuned. A's condition was too specific: it was false of the situation, and it should have been true. The appropriate tuning is to generalize A's condition. Conversely, B's condition was too general: it was true of the situation and it should have been false; so its condition needs to be specialized. Generalization and specialization are the two most common forms of condition tuning. A variety of cognitive models have used one or both of them (Anderson, 1982; Langley, 1987; VanLehn, 1987).

Newell and Rosenbloom (1981) invented a mechanism that serves the function of both compounding and tuning. The mechanism, called *chunking*, requires that operators and heuristics be represented as productions that read and modify only the temporary information storage buffer called working memory. It also requires that there be a bookkeeping mechanism that keeps track of which working memory items were read and/or modified over a sequence of production applications. Given a sequence of production applications, chunking creates a new production by putting all the pieces of information that were read on the condition side, and all the pieces that were modified on the action side. This creates a production that does the work of several smaller productions. In this respect, it is just like compounding. However, because the chunking mechanism builds the new production directly from the working memory elements that were accessed, it builds very specific productions that incorporate all the detail of those elements. Thus, chunking "specializes" productions, in a sense. In some circumstances, it can also generalize productions (Laird, Rosenbloom, & Newell, 1986). For this reason, chunking is a form of tuning as well as compounding.

Another learning mechanism, called *proceduralization* is applicable only in models, such as ACT* (Anderson, 1983) or UNDERSTAND (Hayes & Simon, 1974) that distinguish between procedural and declarative knowledge. Such models view the mind as analogous to a program that employs both a data base (= declarative knowledge) and some functions for manipulating it (= procedural knowledge). Procedural knowledge is usually represented as a production system. ACT* and UNDERSTAND assume that when subjects encode the problem stimulus, a declarative knowledge representation of it is built. In order to explain how subjects solve problems initially, it is assumed that they have general-purpose productions that can read the declarative representation of the problem, infer what actions to take, and take them. Thus, the problem is solved initially by this slow interpretive cycle. Proceduralization gradually builds specific productions from the general interpretive ones. It copies a general production and fills in parts of it with information from the declarative knowledge. Thus, proceduralization creates task-specific productions by instantiating the general purpose productions.

Another common learning mechanism is *strengthening* (Anderson, 1982). It is assumed that each operator has a strength, and that the operator selection process prefers stronger operators over weaker ones. The learning mechanism is simply to increment an operator's strength whenever it is used successfully. In order to keep strengths from growing indefinitely, some kind of strength decay is usually assumed.

Another learning mechanisms is *rule induction* (Sweller, 1983). When the sequence of moves along a solution path has a salient pattern, such as two operators being applied alternately, then subjects may notice the pattern and induce a rule that describes it. Several mechanisms for such *serial pattern learning*, as it is sometimes called, have been described (Restle70; Kotovsky & Simon, 1973; Levine, 1975). Sweller and his colleagues (Sweller, 1983; Mawer & Sweller, 1982; Sweller & Levine, 1982; Sweller, Mawer & Ward, 1983) showed that this type of learning is rare when subjects employ means-ends analysis as their problem solving strategy, but that various experimental manipulations can reduce the use of that strategy and increase the occurrence of rule induction.

Notorious technical problems, and a standard solution.

Several of the mechanisms above (tuning and strengthening at least; perhaps also compounding and chunking) require knowing whether the application of a given operator led to success or failure. This presents problems. Often, the operator application may occur quite some time before the problem is successfully solved, so substantial memory capacity may be required in order to remember which operators contributed to the success. Moreover, making learning conditional on success means that no learning will occur until the problem has been solved, but it is quite clear the people can learn in the middle of problem solving. This set of difficulties is sometimes called the *credit assignment* problem.

Another problem common to several mechanisms is that they can build highly idiosyncratic operators. Not only do these idiosyncratic operators waste storage space, they can sometimes grab control of the model and cause it to predict absurd behaviors of the subjects. This problem is sometimes called the *mental clutter* problem.

To handle the assignment of credit problem, the mental clutter problem and others, it is standard to embed the learning mechanisms enumerated above in a sophisticated processing architecture that allows severe constraints to be placed on their operation. A common approach is to assume that the architecture is *goal-based.* All processing is done in the context of the current goal; goals may be pushed and popped, as in the method of operator subgoaling. Goals help solve the assignment of credit problem by allowing success to be defined relative to the given goal, thus providing earlier feedback. Mental clutter is avoided by only combining operators when they contribute directly to the success of the current goal.

This completes the desciption of problem spaces, understanding, search and learning -- the major components of contemporary as well as past theorizing about problem solving. With the theoretical framework in place, it is time to turn to describing the empirical findings that are the second pillar on which forthcoming theories of problem solving will be built.

Schema-driven problem

If one gives subjects the same set of problems many times, they may learn how to solve them and cease to labor through the understanding and search processes described in section 2. Instead, they seem to recognize the stimulus as a familiar problem, retrieve a solution procedure for that problem and follow it. The collection of knowledge surrounding a familiar problem is called a *problem schema*, so this style of problem solving could be called *schema-driven*. It seems to characterize experts who are solving problems in knowledge-rich domains. This section describes it, by first discussing how schemas are used to solve familiar problems, then how they are adapted in solving unfamiliar problems. The last subsection describes how schemas can be explained as the products of the learning mechanisms presented earlier.

Word problems in physics, mathematics and engineering

In many of the knowledge-rich task domains that have been studied, problems are presented as a brief paragraph that describes a situation and asks for a mathematical analysis of it (Paige & Simon, 1966; Bhaskar & Simon, 1977; Hinsley, Hayes & Simon, 1977; Simon & Simon, 1978; McDermott & Larkin, 1978; Larkin et al., 1980; Larkin, 1981; Chi, Feltovich, & Glaser, 1981; Silver, 1981; Chi, Glaser & Rees, 1982; Schoenfeld & Herrmann, 1982; Larkin, 1983a; Sweller, Mawer & Ward, 1983; Anzai & Yokoyama, 1984; Sweller & Cooper, 1985; Reed, Dempster & Ettinger, 1985). Because so much work has been done with word problems, and schemas are so prominent in subjects' behavior when solving word problems, such problems make a good starting place for the examination of schema-driven problem solving.

For purposes of exposition, let us distinguish two types of problem solving. If the subjects are experts and the problem given is an easy, routinely encountered problem, then the subjects will not seem to do any search. Instead, they will select and execute a solution procedure that they judge to be appropriate for this problem. For these subjects, the understanding process consists of deciding what class of problem this is, and the search process consists of executing the solution procedure associated with that class. Let us call this case *routine* problem solving. Of course, experts can solve non-routine problems as well, but on those problems, their performance has a different character. Routine problem solving is discussed first; a discussion of non-routine problem solving follows.

Schemas

Problem type: There is a river with a current and a boat which travels at a constant velocity relative to the river. The boat travels downstream a certain distance in a certain time, and travels upstream a certain distance in the same amount of time. The difference between the two distances is either given or desired.

Solution Information: Given any two of (a) the difference between the upstream and downstream distances, (b) the time and (c) the river current's speed, the other one can be calculated, because the boat's speed drops out. First write the distance-rate-time equations for the upstream and downstream trips, then subtract them, then solve the resulting equation for the desired in terms of the givens.

Table 4: A schema for river problems

In order to explain routine problem solving, it is usually assumed that experts know a large variety of problem schemas, where a *problem schema* consists of information about the class of problems the schema applies to and information about their solutions. Problem schemas have two main parts, one for describing problems and the other for describing solutions. As in illustration, table 4 shows a schema that an expert in high school algebra might have.[8] This schema applies to a very specific class of problems, and it contains the "trick" for solving problems in that class. If upstream/downstream problems are solved in a general way, they translate into a system of six linear equations in nine unknowns. Thus, given any three quantities, all the others can be calculated. But the trick upstream-downstream problems give only two quantities, not three. However, the quantities given just happen to be such that subtracting the distance-rate-time equations yields a solution. Thus, this schema encodes expert knowledge about how to recognize and solve this special "trick" class of river problems.

Routine problem solving consists of three processes: selecting a schema, adapting (instantiating) it to the problem, and executing its solution procedure. These three processes will be discussed in the order just given.

Schema selection often begins when a particular schema suddenly pops into mind. This *triggering* process, as it is called, is not well understood. It seems to occur early in the processing of the problem stimulus. For instance, when Hinsley, Hayes and Simon (1977) read algebra word problems slowly to their subjects, more than half the subjects selected a schema after hearing less than one-fifth of the text. Hinsley et al. give the following example.

> For example, after hearing the three words, "A river steamer..." from a river current problem, one subject said, "It's going to be one of those river things with upstream, downstream, and still water. You are going to compare times upstream and downstream -- or if the time is constant, it will be the distance." Another subject said, "It is going to be a linear algebra problem of the current type -- like it takes four hours to go upstream and two hours to go downstream. What is the current -- or else it's a trig problem -- the boat may go across the current and get swept downstream." [pg. 97]

These quotes indicated that the triggering process seem to happen very early in the perception of the problem. Experts reading physics problems also tend to trigger schemas early (Chi, Feltovich, & Glaser, 1981).

Once an initial schema has been triggered, it guides the interpretation of the rest of the problem. In this case, it appears that both subjects have selected a general river-problem schema that has several subordinate schemas, representing more specific river-problem schemas. The first subject seems to know about the schema of table 4, and is considering whether this problem might be an instance of it or of a different schema (constant distance river schema). The second subject is also consider several special cases of the generic river crossing problem. In this case, triggering the general river-crossing schema could guide subsequent processing by setting up some expectations about what kinds of more specific, subordinate schemas to look for. The subjects probably used these *expectations to read the problem statement selectively, looking for* information that will tell them which of their expectations is met. This strategy of starting with a general schema and looking for specializations of it may be a common one in understanding, since it appears in physics problem solving as well (Chi, Feltovich, & Glaser, 1981).

Selection of a schema goes hand in hand with *instantiating* it to the given problem. Instantiation means adapting the schema to the specific problem. For instance, to adapt the schema of table 4 to the problem

> A river steamer travels for 12 hours downstream, turns around and travels upstream for 12 hours, at which point it is still 72 miles from the dock that it started from. What is the river's current?

requires noting which two quantities are given and which is desired. In the standard terminology, the variable parts of a schema (i.e., the three quantities, in this case) are called *slots* and the parts of the problem which instaniate the slots are called *fillers*. So instantiating a schema, in the simplest cases at least, means filling its slots. Often, occasions of slot-filling are mingled with occasions of specialization, where a schema is rejected in favor of a subordinate schema. Indeed, it is sometimes not easy to distinguish, either empirically or computationally, between instantiation and specialization.

Experts seem to derive features of problem situations that novices do not and to use the derived features during selection and instantiation. Such features called *second-order* features (Chi, Feltovich, & Glaser, 1981) because they seem to be derived by some kind of elaboration process rather than being directly available in the text. An example of this is found in the remark quoted earlier of a Hinsley et al. subject, who said "if the time is constant, it will be the distance." But the problem states, "A river steamer travels for 12 hours downstream, turns around and travels upstream for 12 hours,..." The problem does not state that the time is constant, but as that seems to be the feature that the subject looks for, it is likely that the subject will notice the equality of the two given times, and immediately infer that the temporal second-order feature that he seeks is present. Chi et al. (1981, 1982) provide evidence that experts in physics notice second-order features but novices do not.

The whole process of selecting and instantiating a schema is a form of elaboration because it does not actually change the problem state, but augments it with a much richer description. An earlier discussion (section) indicated that elaboration could be viewed equally well as search or understanding.

Following the solution procedures

Once a schema has been selected and instantiated, the subject must still produce a solution to the problem. For routine problem solving, this is can be accomplished by simply following, in a step by step fashion, the solution procedure that constitutes the second half of the schema. For instance, the algebra schema of table 4 contains a three step solution procedure: write the two distance-rate-time equations, subtract them, and solve for the desired quantity in terms of the givens. Following procedures such as this one is the third and final process in schema-driven problem solving.

Procedure following is not always trivial. Sometimes the execution of a step may present a subproblem that requires the full power of schema-driven problem solving for its solution. For instance, the first step above asks that the subject write a distance-rate-time equations, but it does not say how. Schema-driven problem solving can easily solve this subproblem, provided that subject knows schemas such as the following one:

Problem: There is a boat moving downstream on a river at a constant rate. An distance-rate-time equation is desired.

Solution: The equation is the standard distance-rate-time equation with the rate equal to the sum of the boat's speed and the river current's speed.

These simple examples illustrate that the overall process of schema-driven problem solving is *recursive*, in that executing one small part of the process can potentially cause complete, recursive invocation of the problem solving process.

There is yet another complexity involved in following solution procedures. It is quite likely that some subjects do not follow the procedure's steps in their standard order. They prefer to use a permutation of the standard order, and sometimes these permutations produce different effects than the standard one. This particular effect is difficult to demonstrate experimentally, because it is difficult to find out exactly what the subjects' solution procedures are. Thus, one cannot be certain whether they are following a standard-order procedure in a non-standard way, or whether they simply have a procedure with permuted steps. Perhaps the best evidence so far comes from the task domain of subtraction calculation, where children are asked to work problems such as:

$$\begin{array}{r} 3\ 4\ 5 \\ -\quad 7\ 9 \\ \hline \end{array}$$

Although this is not at all a knowledge-rich task domain, it is a task where the subjects following procedures, so the findings there might generalize to expert's following the solution procedures of their schemas. Even if the results do not generalize in any detail, subtraction still serves as a convenient illustration for how procedures can be followed flexibly.

VanLehn and Ball (1987) discovered 8 students (out of a biased sample of 26) who used non-standard orders. All of the orders standardly taught in the United States have the student finish one column before moving on to the next, even if that column requires extensive borrowing from other columns. However, the 8 students did not always exhibit a standard order. For instance, some students did all of the problem's borrowing first, moving right to left across the columns, then returned, left to right, answering the columns as they go. It was also found that students would often shift suddenly from one order to another. This is consistent with the hypothesis that these students' underlying procedures were stable, but they choose to permute the order of steps during execution. This conclusion is bolstered by the authors' demonstration that a small set of standard-order procedures gives excellent fits to the observed orders when they are executed by a simple queue-based interpreter. Moreover, that set of standard-order procedures can all be produced by an independently motivated learning model when it is instructed with the same lessons that the subjects received (VanLehn, *1983; VanLehn, ress). These results led* VanLehn and Ball to conclude that their 8 subjects were indeed executing standard-order procedures in a non-standard way. Whether this same flexibility in execution will turn up in expert behavior remains to be seen.

Non-routine solving of word problems

The preceding section described the routine case of problem solving wherein a single schema matches the whole problem unambiguously and its solution procedure can be followed readily, encountering at worst only routine subproblems. This section describes some of the many ways that schema-driven problem solving can be non-routine. Research is just begining in this area, so many of the proposed processes are based only on a rational extension of the basic ideas of routine problem solving and as yet have not been scrutinized experimentally.

Perhaps the most obvious source of complexity in expert problem solving occurs when more than one schema is applicable to the given situation. Since the subjects do not know which schema to select (by definition), they must make a tenative decision and be prepared to change their mind. That is, they must search. Such cases illustrate that schema selection can be usefully viewed as the result of applying an operator that produces a new state in a problem space search. The new state differs from the old one only in that it contains an assertion marking the fact that the schema has been selected. Redoing a schema selection becomes a case of the usual backing up in search of a problem space. As remarked earlier, schema selection and instantiation are forms of elaboration, and thus can be viewed either as search or understanding. When the subject is uncertain which schema to select, it is useful to view schema selection as search.

Larkin (1983) provides a nice example of such a search. She gave five expert physicists a simple, but difficult physics problem. Although two subjects immediately selected the correct schema, and one even said, "I know how to do this one. It's virtual work." [Op cit., pg. 93], the other three subjects tried two or more schemas. Each schema was instantiated and its solution was partially implemented. Usually, the solution reached a contradiction (e.g., a sum of forces that should be zero is not). Only the *final schemas selected by these subjects* led to a contradiction-free solution. Thus, schema selection plays a crucial role in these subjects search for a solution.

Another type of difficulty occurs when no schema will cover the whole problem, but two or more schemas each cover some part of the problem. The problem is to combine the schemas so that they cover the whole problem. Larkin (1983) gives some examples of experts combining schemas.

A third type of difficulty occurs when execution of a solution procedure halts because the procedure mandates an impossible action or makes a false claim about the current state. Such an event is called an *impasse* (Brown & VanLehn, 1980; VanLehn, 1982). Although this notion was originally invented in order to explain the behavior of children executing arithemetic procedures (Brown & VanLehn, 1980), it readily applies to experts executing solution procedure. For instance, Larkin's (1983) three experts reached impasses during their initial solving of the physics problems because their selected schema's solution procedure claimed, for instance, that the balance of forces should be zero when it was not. The subject's response to an impasse is called a *repair*, because it fixes the problem of being stuck. In the case of Larkin's experts, the repairs were always to reject the currently selected schema and select another one. Such backing up may be a frequent type of repair, but it is certainly not the only type (Brown & VanLehn, 1980; VanLehn, 1982; VanLehn, ress).

This subsection has enumerated several processes that seem to occur regularly in non-routine problem solving: ambiguity in selecting a schema, schema combination, impasses and repairs. However, these are probably just a few of the many interesting types of behavior that occur when experts solve difficult problems. Much research remains to be done.

Expert problem solving in other task domains

It may be unsurprising that schemas provide the basis for a natural account of word problem solving, because the schemas have long been used in psychology to explain how people process paragraph-sized pieces of text (Bartlett, 1932). On this view, the prominence of schemas in expert solutions of word problems is due to the task domain, rather than the expertise of the subjects. However, there is some evidence that schemas, or something much like schemas, are used by experts in other task domains as well.

For instance, research on programing and algorithm design (Adelson, 1981; Jeffries, Turner, Polson & Atwood, 1981; Anderson, Farrell, & Saurers, 1984; Pirolli, 1985; Pirolli & Anderson, 1985; Kant & Newell, 1984) has shown that experts know many schemas such as the one shown in table

Problem: Given a list of elements and a predicate, remove the elements of which the predicate is false.

Solution: Use a trailing-pointer loop. The initialization of the loop puts a new dummy element on the front of the list, and it sets two variables. One variable (called the trailing pointer) points to the list, and the other points to the second element of the list (i.e., the first element of the original list). The main step of the loop is to call the predicate on the element, and if the predicate is false, then the element is spliced out of the loop, using the trailing pointer. If the predicate is true, then both pointers are advanced by one element through the list. The loop terminates after the last element has been examined. At the conclusion of the loop, the list must have the dummy first element removed.

Table 5: A schema for programming

5. This schema is midway between a schema for coding and one for algorithm design. Coding schemas often mention language-specific information. For instance, schemas for recursion in Lisp may mention positions in Cond clauses where one should place the code for the recusive step and the terminating step (Pirolli, 1985; Pirolli & Anderson, 1985). Algorithm design schemas mention more general techniques, such as dividing a set of data points in half, recursively performing the desired computation on each half, and combining the solutions for each half into a solution for the whole (Kant & Newell, 1984).

In many respects, the use of such schemas resembles the use of word problem schemas. In particular, they must be selected and instantiated before their solution halves are implemented. Moreover, the problem solving process is recursive in that doing a small part of the process, such as filling a slot in a selected schema, may create a subproblem whose solution requires more schemas to be selected and implemented (Kant & Newell, 1984).

In some task domains, schema-driven problem solving does not seem to play a prominent role in expert behavior. For instance, Lewis (1981) studied algebra equation solving using rather tricky problems in high-school algebra, such as:

Solve for x: $x+2(x+2(x+2))=x+2$

Lewis compared expert professional mathematicians with high-school and college students. If the experts were doing schema-driven problem solving, one might expect them to say, "Oh, one of those," and produce the answer in one step. This almost never occurred. In fact, Lewis concludes that "the expert's performance was not sharply different from that of the students,"[pg. 85] except that the experts make fewer mistakes.

There is no space in this chapter for a thorough review of the expertise literature. Fortunately, there is a recent review (Riemann & Chi, ????) and a recent collection of articles (Chi, Glaser & Farr, ???). The major purpose of this section is to introduce an analytical idea -- schema-driven problem solving -- that has sometimes proved useful in understanding problem solving. The last task of this section is show how this notion relates to the standard theory of problem solving, which was presented in the preceding section.

Relationship to the standard theory

It is quite plausible that schemas are acquired via the learning mechanisms of the standard theory. Although there is some disagreement about the exact nature of the learning mechanisms, they all predict that experts will acquire many large, specialized pieces of knowledge, regardless of whether they are called chunks, macro-operators or compounded productions. Each piece is highly tuned, in that it will only apply to a small class of problems, and yet it is be quite effective in solving those problems. At a rough qualitative level, the assumptions about the products of learning fit nicely with the assumptions about schemas.

Closer examination yields more points of agreement. In particular, the increased size of the units of knowledge can be expected to change the character of the problem solving somewhat. To demonstrate this, suppose that compounding glues together several operators that make physical changes in the world, and these actions cannot be performed simultaneously. This means that application of the macro-operator results in execution of a single action plus an intention (plan) to perform some others. Thus, the macro-operator is more like a procedure (or stored plan) than an operator per se. Thus, it is likely that the solution procedures of schemas correspond to the products of compounding, chunking or similar learning mechanisms.

Operator selection can also be expected to change character as learning proceeds. When a novice searches a problem space, operator selection is taken care of by a proceed strategy and some heuristics. But the experts' macro-operators/schemas are very specialized, so it might take some extra work to analyze the current state well enough to be able to discriminate among the relevant operators in order to find the appropriate one. Elaborations may be needed in order to build a case for selecting one operator over the others. Thus, increases in the number of available units of knowledge and in their specificity is consistent with the complicated selection processes that seems to characterize schema-driven problem solving.

Although it certainly seems that schema-driven problem solving is the product of learning during the course of problem space search, there are many technical details that stand in the way of demonstrating this. At this writing, no computer program exists that can start off as a novice in some knowledge-rich task domain and slowly acquire the knowledge needed for expert performance. Thus, we lack even a computationly sufficient account of the novice-expert transistion, let alone one that compares well with the performance of human learners. Needless to say, many theorists are hard at work on this project, so progress can be expected to be quite rapid.

This concludes the discussion of theoretical concepts. The remainder of the chapter reviews empirical findings and their relationship to theory.

Major empirical findings

Recent work in Artificial Intelligence has dispelled much of the mystery surrounding human problem solving that was once called "inventive" (Stevens, 1951), "creative" (Newell, Shaw & Simon, 1962) or "insightful" (Weisberg & Alba, 1981). Computer programs now exist that can easily solve problems that were once considered to be so difficult that only highly intelligent, creative individuals could solve them. Many of the formal mechanisms mentioned earlier are used to build such programs. The new mystery of human problem solving is to find out *which* of the now-plentiful solution methods for "creative" or "inventive" problems are the ones employed by subjects. Thus, experimental findings in problem solving have taken on a new importance. This section reviews the experimental findings that seem most robust.

Practice effects

The literature on practice effects goes back to the turn of the century (see Fitts and Posner, 1967, for a dated but still relevant review). However, most of the earlier work dealt with perceptual-motor skills, such as sending Morse Code. This subsection discusses only the practice effects that have been demonstrated explicitly on problem solving tasks (also called *cognitive skills*). It starts with effects seen during the early stages of practice and progresses towards effects caused only by years of practice.

Reduction of verbalization.

It has often been noted that during the initial few minutes of experience with a new task, the subjects continually restate the task rules, but as practice continues, these restatements of rules diminishes. For instance, Sweller, Mawer and Ward (1983) tracked naive subjects as they learned how to solve simple kinematics problems that require knowing a half dozen equations relating velocity, distance and acceleration. They found that the number of times a subject wrote one of the equations without substituting any quantitites for its variables decreased significantly over the practice period. Similar findings have been reported by Simon and Simon (1978), Anderson (1982) and Krutetskii (1976). Reduction of verbalization can be explained as the result of proceduralization (Anderson, 1982).

Tactical learning

On some knowledge-lean tasks subjects quickly improve in their ability to select moves. Greeno (1974) showed that only 3.6 repetitions of the Missionaries and Cannibals puzzle were required on average before subjects met a criterion of two successive error-free trials.[9] Reed and Simon (1976) and Anzai and Simon (1979) present similar findings. Rapid tactical learning is consistent with several of the learning mechanisms mentioned earlier. Tuning, chunking and strengthening all suffice to explain the finding, provided that they are assumed to happen rapidly (e.g., at every possible opportunity). Atwood, Polson and their colleagues have also shown that simply remembering what states have been visited also suffices for modeling rapid tactical learning solution paths (Atwood & Polson, 1976; Jeffries, Polson, Razran & Atwood, 1977; Atwood, Masson & Polson, 1980).

The power law of practice

A great deal of experimental evidence shows that there is a power-law relationship between the speed of performance on perceptual-motor skills and the number of trials of practice (Fitts & Posner, 1967). If time per trial and number of trials are graphed on log-log paper, the curve is a straight line. Recently, the power law has been shown to govern some cognitive skills as well (Newell & Rosenbloom, 1981; Neves and Anderson, 1981).

The power law of practice does not fall out naturally from any single one of the learning mechanisms discussed above. Both chunking and compounding accelerate performance, but they tend to produce exponential practice curves instead of power-law curves (Lewis, 1979; Neves and Anderson, 1981; Newell & Rosenbloom, 1981). That is, they learn too fast. Various proposals have been put forward for slowing the mechanisms down (Anderson, 1982; Rosenbloom, 1983), but the experiments that split these hypotheses have yet to be performed.

The biggest theoretical problem presented by the power-law finding is that the effects of practice never stop. Crossman (1959) showed that a subject who rolled cigars for a living was still getting faster after several years of practice. Chunking, compounding and other such mechanisms will have long since built a single huge operator for the task, so they cannot readily explain how performance continues to improve.

Other possible effects, not yet demonstrated.

From the perceptual-motor literature, it seems likely that the following findings also apply to problem solving: (1) Within limits, subjects can trade speed for accuracy, reducing one at the expense of increasing the others. No theoretical work has tried to model this. (2) If exactly the same task is practiced for hundreds of trials, it can be be automatized, that is, it will be very rapid, cease to interfer with concurrent tasks, and run to completion once started even if the subject tries to stop it. However, if the task varies beyond certain limits during training, even hundreds of practice trials do not suffice for automatization (Schneider & Shiffrin, 1977; Shiffrin & Schneider, 1977). Although chunking, compounding and similar mechanisms are consistent with the general quality of automatization, it is not yet clear whether they can explain why some types of practice cause automization and others do not. (3) The distribution of practice makes a difference in the speed of learning, but the effect depends on the structure of the skill being practiced. Sometimes practicing parts of the skill before the whole is better, and sometimes not. Sometimes many short practice sessions are better than a few long ones, and sometimes not. Current cognitive theory has not yet tried to explain these effects. Also, experimental work is needed in order to check that these effects are not limited to perceptual-motor skills, but are found with cognitive skills as well.

Problem isomorphs

Many knowledge-lean tasks have an "intended" problem space, which is the problem space that people who are very familiar with the problem assign to it. The LMS puzzle discussed earlier is a case in point. The intended problem space is the one used by Cathy Of course, a subject's problem space is not necessarily the intended one, as illustrated by the 60-year old subject who initially interpreted the LMS puzzle as an arithmetic word problem.

Two problems are said to be *isomorphic* if their intended problem spaces are isomorphic. Two problem spaces are isomorphic if there is a one-to-one correspondence between states and operators such that whenever two states are connected by an operator in one problem space, the corresponding states are connected by the corresponding operator in the other problem space. This section compares problem solving behaviors on isomorphic problems.

Varying the cover story does not affect difficulty.

A simple way to create an isomorphic puzzle is to change the cover story. For instance, the Missionaries and Cannibals puzzle has three missionaries and three cannibals trying to cross a river subject to certain restrictions. Several investigators (Greeno, 1974; Jeffries, Polson, Razran & Atwood, 1977) created problem isomorphs by substituting elves and men (or other pairs of creatures) for the missionaries and cannibals. This change in the cover story of the puzzle had no measurable effect on the solution times or patterns of moves. This result tends to support the idea that subjects really are thinking of the puzzle as a formal problem space, and in fact, as the intended problem space.

Other variations significantly effect difficulty

Although changing the cover story does not seem to effect problem solving behavior, other manipulations of puzzles can have a very significant effect on the relative difficulty of problem isomorphs. It is not yet clear how these manipulations differ from the cover story manipulation. For instance, Kotovsky, Hayes and Simon (1985) studied isomorphs of the Tower of Hanoi, such as the tea ceremoney puzzle mentioned earlier, and found that some isomorphs took 16 times as long to solve as other isomorphs (29.39 minutes vs. 1.83 minutes). Reed, Ernst and Banerji (1974) obtained similiar but less dramatic results with isomorphs of the Missionaries and Cannibals puzzle.

Kotovsky et al. develop a model that exhibits good qualitative agreement with their data. They assume that subjects search in a problem space, but not the intended problem space. Rather, they search in a finer grained problem space where it takes several operator applications to achieve the same effect as one operator application in the intended problem space.

Transfer and problem solving by analogy

Before presenting some findings concerning transfer and analogy, a brief introduction to this rather complex subfield is in order. It is clear that complete transfer of expertise between domains never occurs (e.g., going to medical school does not make one a good lawyer). However, it may be that incomplete transfer occurs. There are two possibilities: (1) The domain-specific knowledge of two task domains may overlap. For instance, chemists and physicists overlap in their knowledge of mathematics and fundamental properties of matter and energy. Thus, there should be *specific transfer* of expertise in one domain to another. (2) If two task domains seem to have no overlap in their requisite knowledge, there still may be *general transfer* because problem solving in both domains may require an organized, methodical style of thinking, so training in that type of thinking in one task domain may give a subtle advantage in another task domain. For instance, learning to program a computer is often thought to increase one's ability to do logical and quantitative problem solving of all types (e.g., Papers, 1980).

However, general transfer is difficult to study, and as a consequence, there is some doubt as to whether general transfer even exists. For a recent review of the general transfer literature, see Pea (1986). The rest of the remarks here will concern specific transfer.

The existence of specific transfer has been amply demonstrated (Thorndike & Woodworth, 1901; Singley & Anderson, 1985; Singley & Anderson, 19??; Singley, 1986; Kieras & Bovair, 1986; Kieras & Polson, 19??; Reed, Ernst & Banerji, 1974; Kotovsky, Hayes & Simon, 1985). However, the exact nature of specific transfer is still being investigated. One leading theory, originated by Thorndike and rendered more precise by Kieras, Bovair, Singley and Anderson, is that transfer is accomplished by actually sharing (or copying) relevant units of knowledge. For instance, it is possible to notate knowledge of procedural skills as production systems in such a way that the degree of transfer is directly proportional to the number of shared productions (Kieras & Bovair, 1986; Kieras & Polson, 19??; Singley & Anderson, 1985; Singley, 1986; Singley & Anderson, 19??).[10] This theory is called the *identical elements* theory of transfer.

In the identical elements theory, knowledge is viewed a set, so calculating the overlap between two task's knowledge structures amounts to simply taking a set intersection. However, another common view has knowledge structured as a semantic net (see chapter ???, this book). Because a semantic net is a labelled directed graph rather than a set, there are multiple ways to calculate the overlap of two semantic nets. See Gentner (in press) and Holyoak (1985) for two contrasting views on how people do it. The identical elements view and the mapping view of transfer can be seen as compatible hypotheses which examine the same phenomenon at different levels of description. Identical elements theory counts the number of units transfered, while mapping theories explain exactly what parts of an element are transfered.

Having presented a few basic concepts about specific transfer and analogy, a few findings from this large and rapidly growing literature can be presented.

Asymmetric transfer occurs when one task subsumes another.

The identical elements theory of transfer predicts that when one task's productions are a subset of another's, transfer will appear to be asymmetric even though it is underlyingly symmetric. Training on the harder task--the one with more productions--will cause complete competence in the easier task because all the units for the easier task will have been learned. On the other hand, training in the easy task will cause only partial competence in the harder task. Thus, although the same number of units is being transfered in either case (i.e., the underlying transfer is symmetric), the measured transfer is asymmetric. This prediction is consistent with several findings of asymmetric transfer where competence in the more difficult task transfers to the easier task, but not vice versa (Reed, Ernst & Banerji, 1974; Kotovsky, Hayes & Simon, 1985; Singley, 1986)

Negative transfer is rare.

Negative transfer occurs when prior training on one task slows down the learning of another task or blocks its performance completely. The identical elements theory predicts that there will never be negative transfer. Singley and Anderson (1985, in press) tested this implication by using two versions of the same text editor. The only difference between the editors was the assignment of keys to commands. They trained two groups of subjects on the two editors for 6 hours, then switched one group to the other editor, and trained that group for 6 nours. If negative transfer occurs, then the learning curve for the transfer group after it had been switched over should start lower and/or rise more slowly than the learning curve on the control group during its first 6 hours of training. This did not occur. Instead, the learning curve for the transfer group started higher than the learning curve for the control group, thus indicatinq substantial positive transfer. Moreover, the transfer group's curve paralleled the control group's curve, indicating that there was no detrimental affect of the prior training on subsequent learning. Thus, the experimental results fit the predictions of the identical elements theory quite well. Kieras and Bovair (1986) found a similar lack of negative transfer.

Singley and Anderson (1985) point out that editor users probably hope for total positive transfer when they switch editors. That is, they anticipate being able to use the new editor just as well as they used the old. Because their actual performance on the new editor is not as fluid as their old performance, they say they have suffered "negative transfer." However, because their actual *performance is much better than* a novice, they actually are enjoying a large degree of positive transfer, even though it is not the total transfer they hoped for.

Set effects

The lack of negative transfer contradicts intuition. For instance, Fitts and Posner (1967) give the following rather compelling examples of negative transfer:

> If you drive in a country in which traffic moves on the opposite side of the road from the side on which your are accustomed to driving, you are likely to find it difficult and confusing to reverse your previous learning; similarly, in cases where the faucets which control hot and cold water are reversed from their usual positions, months of learning are often required before their operation is smooth. [pg. 20]

The first example probably does not constitute true negative transfer, because it probably takes less time to learn to drive on the opposite side of the road than it takes to learn to drive initially. This is another case of frustrated expectations for massive positive transfer. On the other hand, it does not take months to learn the positions of the hot and cold water controls initially, so the last example constutes a clear case of negative transfer. How does this example differ from the Singley and Anderson experiments?

In a more fine-grained analysis of their data, Singley and Anderson (in press) found that subjects would sometimes choose a less efficient method during the transfer task for achieving certain of their text editing goals, presumably because the chosen method was more familiar to them from their prior training. This is similar to the set effects observed by Luchins, Duncker and others (Luchins, 1942; Duncker, 1945; Greeno, Magone & Chaiklin, 1979; Sweller & Gee, 1978). *Set effects* occur when there are alternatives in a problem solving task, and some of the alternatives are more familiar than others. The set effect is that the subjects tend to pick the familiar alternative even if it is not the best. The hot and cold water controls are an example of a set effect. In short, set effects are a special kind of negative transfer that does seem to take place. Moreover, its existence is not predicted by the identical elements theory.

There are two major kinds of set effects in the literature. *Functional fixity* refers a familiarity bias in the choice of functions for an object. In Duncker's famous task of constructing a wall-mounted candle holder, the subjects tend to view the box as a container for tacks, rather than as a platform for the candle (Duncker, 1945; Weisberg & Alba, 1981; Greeno, Magone & Chaiklin, 1979). *Einstellung* refers to a familiarity bias in the choice of a plan. In Luchin's water jug task, the subjects are given a series of problems that can all be solved with the same sequence of operations. Presumably, this induces the person to formulate this repetitive sequence as a plan, and reuse it on the later problems in the series. The Einstellung effect occurs when the subject is given a problem that can be solved two ways. The plan will solve it, and so will a sequence of operations that is much shorter than the plan. Although the short sequence of operations is the best choice, many subjects use the plan instead (Luchins, 1942).

Spontaneous noticing of a potential analogy is rare.

In the experiments on negative transfer, the stimuli were identical in the training and transfer phases, but the responses were supposed to be different. In experiments on problem solving by analogy, there are also training and transfer phases, but it is the stimuli (tasks) that are different across the two phases. The responses are supposed to be the same, or at least analogous. For instance, a subject might be given one puzzle to solve, then another isomorphic puzzle. If they use the solution of the first puzzle in solving the second, then they are said to have done problem solving by analogy. Problem solving by analogy can be detected by a number of means, such as verbal protocols or decreases in solution times compared to a control.

In some analogy experiments, the subjects are not told that the two tasks are related. Instead, they are simply given training on one task, then switched to another task without comment. In such circumstances, it is common to find that no transfer occurs. For instance, Reed, Ernst and Banerji had subjects solve two problem isomorphs in the same 30 minute experiment. They demonstrated that transfer occurred only when subjects were told the relationship between the two puzzles, otherwise the subjects did not seem to notice that the two tasks were analogous. Similar findings are reported by Gick and Holyoak (1980, 1983), Gentner (in press), and others.

This result is consistent with the common finding that problem solving by analogy is often used by students working problems in an instructional setting (Anderson, Farrell, & Saurers, 1984; Pirolli & Anderson, 1985; LeFevre & Dixon, 1986; Chi, Bassok, Lewis, Reimann & Glaser, ress). For instance, a student working physics problems at the end of a chapter expects that problems solved as examples in the chapter will use the same methods, so they actively page through the chapter seeking such solved problems in order to use them as analogs (Chi, Bassok, Lewis, Reimann & Glaser, ress). Although students may not have been explicitly told that the chapter's examples are similar to the exercises, experienced students make that assumption anyway.

Spontaneous noticing is based on superficial features.

Even in experiments where subjects are neither told to look for analogies nor led by their past experience to look for them, spontaneous noticing of analogies does sometimes occur. When it does, it seems to be based most frequently on noticing superficial similarities between the tasks. Ross (1984, 1987) taught subjects several methods for solving probability problems. Each method was taught with the aid of an example. The example contents varied (e.g., dice, car choice, exam scores, etc.). Subjects were tested with problems whose contents were either new, superficially similar to some training example for the appropriate method for solving that problem, or superficially similar to an inappropriate example. Subjects often chose to use the method whose example is similar to the test problem, even if that method is inappropriate. Thus, subjects seem to have been cued by the surface similarities, rather than the deep structures of the examples. Similar effects have been found for text editing (Ross, 1984), algebra story problems (Reed, 1987), and simple stories (Gentner & Toupin, 1986).

Noticing that an analogy is useful is only part of the process of solving problems by analogy.

In some cases, it is quite non-trivial to map the solution from the analog over to the target problem. Several studies (Reed, Dempster & Ettinger, 1985; Catrambone & Holyoak, 1987) have shown that even when subjects are told about the existence of the analogy, they sometimes have difficulty making use of them. However, the conditions that facilitate and inhibit such transfers are not yet entirely clear.

Expert-novice differences: problem solving studies

The next few sections contrast expert and novice problem solvers. The term "expert" is usually reserved for subjects with several thousand hours of experience (there are 2000 working hours in a year). Hayes (1981) argues that no one, not even a child prodigy, becomes a world-class expert without at least 20,000 hours of experience. Although the term "expert" is used in a fairly uniform way, there is substantial variation in the literature on the use of "novice." For some experiments, subjects who know nothing about the task domain are selected, given an hour or two of training, and then asked to solve the experimental problems. In other experiments, the novices are students who have taken one or two college courses in the subject. These substantial differences in training explain many of the apparent contradictions in the findings. In order to keep things straight in this chapter, "pre-novice" will be defined to mean someone with only a few hours training, and "novice" will mean someone with several hundred hours of training (approximately a college course's worth). Given these definitions, there are several unsurprising findings to mention before bringing out the findings that could really be called discoveries.

Experts can perform faster than novices.

If required to perform quickly, an expert can generally perform faster than a novice. For instance, a master chess player can play lightning chess but a novice cannot (de Groot, 1965). Somewhat surprisingly, if experts are not required to perform quickly, they often take about as long to solve a task as novices (Chi, Glaser & Rees, 1982).

Experts are more accurate than novices.

Expertise is correlated with the quality of the solution given by the subject. With one exception, all the expert-novice studies cited in this section show that experts perform better than novices.[11] The exception is making decisions based on uncertain evidence. In a recent review, Johnson (1988) summarizes the evidence as follows:

> In many studies, experts have not performed impressively at all. For example, many expert judges fail to do significantly better than novices who, at best, have slight familiarity with the task at hand. This result has been replicated in diverse domains such as clinical psychology (Goldberg, 1970), graduate admissions (Dawes, 1971), and economic forecasting (Armstrong, 1978). Not surprisingly, this has led to strong recomendations. Consider the following recomendation about experts' forecasts: "Expertise beyond a minimal level in the subject area is of almost no value...The implication is obvious and clear cut: Do not hire the best expert you can -- or even close to the best. Hire the cheapest expert." (Armstrong, 1978, pg. 84-85).

Note that these authors, while denigrating the performance of experts, never claim that experts perform *worse* than novices. In fact, Johnson's review goes on to show that experts are usually better than novices, although they are sometimes substantially worse than simple mathematical decision-making models.

Strategy differences.

In as much as general strategy can be characterized, it appears that experts and novices tend to use the same general strategy for a given problem, but pre-novices sometimes use quite different strategies. For instance, Jeffries, Turner, Atwood and Polson (1983) contrasted the protocols of expert, novice and pre-novice software engineers as they solved a complex design problem. Both experts and novices used a top down, breadth-first, progressive-refinement design strategy. They decomposed the overall system into a few big modules, refined each module into submodules, then refined each submodules into sub-submodules, and so on until the design was detailed enough that they could begin writing program code. The pre-novice, however, began writing code almost immediately, with no sign of a top-down design strategy. Similarly, in the solution of physics problems, no strategic differences were found between experts and novices (Chi, Glaser & Rees, 1982), but pre-novices were found to use a different strategy than either novices (Sweller, Mawer & Ward, 1983) or experts (Simon & Simon, 1978). Several other investigators (de Groot, 1965; Charness, 1981; Lewis, 1981) found no major strategic differences between experts and novices. In short, at a general level of description, the strategies of experts and novices are the same, while pre-novices may have a quite different strategy.

Self-monitoring skill

Experts seem better at monitoring the progress of their problem solving and allocating their effort appropriately. Schoenfeld (1981) analyzed protocols of experts and novices who were solving unusual mathematical problems. Both experts and novices had to search; the problems were not routine even for the expert. However, the experts' search was more closely monitored.

Approximately once a minute, the experts would make some comment that either evaluated their current direction (e.g., "Isn't that what I want?"), or assessed the likelihood of a contemplated approach (e.g., "knock this off with a sledgehammer" meaning that the approach is too-high powered and unlikely to work), or assessed the difficulty of a subproblem before attempting it (e.g., "this is going to be interesting ...."). In contrast, the novices would generally adopt a single approach with little assessment of the liklihood of success, then follow it for ten or twenty minutes, without considering abandoning it. Schoenfeld concludes that "metacognitive or managerial skills are of paramount importance in human problem solving." The same sort of managerial monitoring is also evident in Larkin's (1983) protocols of physicists and Jeffries et al. (1981) protocols of programmers.

A related finding is that experts are able to estimate the difficulty of a task with higher accuracy than novices. For instance, Chi, Glaser and Rees (1982) found that experts are more accurate than novices at rating the difficulty of physics problems. Chi (1978) found that expert chess players are better than novices at estimating how many times they will need to see a given board position before being able to reproduce it correctly. This ability to estimate the difficulty of subtasks is probably important for allocating effort.

The hypothesis that experts have more schemas than novices is consistent with their superior self-monitoring ability. Suppose that subjects estimate the difficulty of a subproblem by first finding the best fitting schema, then combining its known difficulty with an estimate of the quality of the fit. The estimated quality of fit is needed because a poorly fitting schema means some extra work may be required in order to derive the information the schema needs from the problem. If this is how subjects estimate difficulty, then experts should be better at it, because their schemas more plentiful and more specialized, so the fits will be better. Thus, their estimates of difficulty are dominated by the known difficulties of the schema, which is presemably more accurate than the process that estimates the quality of the fit.

Expert-novice differences: memory studies

As the preceding subsection indicated, the speed and accuracy of experts is not accomplished by major, qualitative changes in their problem solving strategies. The effects of expertise are more subtle. For instance, whenever an expert and a novice are deciding which chess move to make, both consider the same number of moves and investigate each move for about the same amount of time. The difference is that the expert only considers the good moves and usually chooses the best one, while the novice considers mediocre moves as well, and often do not choose the best move from those considered (de Groot, 1965; Charness, 1981). Thus, expertise lies not in having a more powerful overall strategy or approach, but rather in having better knowledge for making decisions at the points where the overall strategy calls for a problem-specific choice.

Protocol data is excellent for studying overall strategies because the strategies can be inferred from the patterns of observable moves. However, protocols of even the most articulate subjects are too often silent at the points where the subject is making a problem-specific decision. When subjects do talk, they often say that the choice was obvious (de Groot, 1965). In short, protocols have not proven to be a rich source of data about how experts make decisions. Other types of experiments, however, have been much more illuminating. This sections discusses some of the more robust findings.

Classification of problems

Chi, Feltovitch and Glaser (1981) pioneered the use of a card-sorting technique for assessing differences in how experts and novices classify problems. In the study each card holds the text and diagram for a single elementary physics problem. The subject is asked to sort 24 cards into piles, placing problems that "seem to go together" into the same pile. Subject could sort at their own rate. The novices tended to sort problems on the basis of literal, surface features, such as the types of objects involved (i.e,. inclined planes, pulleys, etc.). On the other hand, the experts tended to sort problems on the basis of the physics principles used to solve the problem (e.g., Newton's second law, or work-energy). Moreover, the names for the piles given by the experts and novices reflected these observational characterizations. A specially constructed set of problems that crossed surface features with solution principles replicated the result. Similar results have been found in mathematics (Silver, 1979; Schoenfeld & Herrmann, 1982) and programming

(Weiser & Shertz, 1983).

It is possible that the classification difference is due to some between-subjects factor For instance, a natural aptitude for mathematics or physics might cause both the classification difference and the career choice of the subject. Schoenfeld and Herrmann (1982) and Silver (1979) showed that this could not be the case. They tested mathematics students before and after courses in mathematical problem solving. The training causes student's classifications to become more expert-like.

These results led Chi and the other authors to hypothesize that experts have problem schemas that novices lack. Roughly put, subjects would put problems into the same category if those problems could be solved using the same problem schema.

However, it could be that the classifications/schemas of experts are not causally related to their improved problem-solving ability. Although this is difficult to test unequivocally, Chi, Feltovitch and Glaser (1981) found that experts could give an abstract "basic approach" to a physics problem (e.g., "I'd use dynamics, F=MA"), while novices could not. Instead, the novices would either give very global statements (e.g., "First, I figured out what was happening ... then I, I started seeing how these different things were related to each other..." [op cit., pg 142]) or they would launch into a detailed solution of the problem. Voss and his colleagues (Voss, Tyler & Yengo, 1983) also found that experts and not novices tended to state basic approaches as they solved problems in governmental policy formation.

In short, it seems that experts and not novices are able to classify problems according problem schemas, and that these same schemas are used to solve problems.

Association structures.

A variety of experimental techniques have been used in memory research to find out about the connectivity of the semantic network of concepts that is assumed to constitute people's declarative knowledge base (see chapter ??, this book). Some of these have been used to try to differentiate the associative structures of experts and novices. For instance, Schvaneveldt et al. (1985) asked expert and novice fighter pilots to rate the similarities of pairs of technical terms from combat flying (e.g., "high yo yo", "switchology"). They used two multi-dimensional scaling algorithms to uncover how the underlying association structures of experts differed from novies. McKeithen, Reitman, Rueter and Hirtle (1981) and Adelson (1981) used item order in free recall, and Pennington (1985) used priming, and Chi, Feltovitch and Glaser (1981) used an elaboration technique to contrast the knowledge structures of experts and novices. All these studies showed that traditional methods for measuring semantic distance or connectedness succeeded in uncovering expert-novice differences in knowledge structure, and in most cases, these differences are readily interpretable in terms of their utility in solving problems.

Episodic memory for problems and solutions

Since Tulving (1972), it is customary to distinguish between semantic memory, which contains generic knowledge applicable to many situations, and episodic memory, which contains specific episodes in the subject's history. The preceding findings concerned differences in the semantic memory of expert and novices. There are also differences in the episodic memory of experts and novices.

A typical experiment on episodic memory presents a stimulus to the subject for a certain length of time, then occupies the subject in various ways for another interval of time, then asks the subject either to recall the stimulus, sometimes with the aid of a cue (hint), or to recognize the stimulus from amongst a set of similar items. Sometimes these three phases are repeated until the subject is able to recall the stimulus perfectly.

The general finding is that experts outperform novices in all versions of this paradigm that have been used so far, but only if the stimuli are ones that the expert would normally encounter in the course of problem solving.

The first experiment of this type was de Groot's (1965) demonstration that chess masters could recall almost all the pieces and positions of a chess board after having seen the board for only five seconds. Novices could recall only a few peices. Moreover, if the stimulus was a chess board with the pieces arranged randomly, the recall of the expert sank to the level of a novice. This finding has been replicated many times with stimuli consisting of chess boards (Chase & Simon, 1973a; Charness, 1976; Frey & Adesman, 1976), Go boards (Reitman, 1976), electronic circuit diagrams (Egan & Schwartz, 1979), and bridge hands (Engle & Bukstel, 1978; Charness, 1979). In all these experiments, the subject was tested almost immediately after the stimulus was presented. Thus, it seems that experts have better short-term memory for problems.

Long-term memory for problems and solutions has also been measured. Chiesi, Spilich and Voss (1979) demonstrated that experts have better long-term recognition and recall of episodes of baseball games. The experts long-term memory is also better for chess games (Chase & Simon, 1973b), bridge hands (Engle & Bukstel, 1978; Charness, 1979) and mathematics problems (Krutetskii, 1976).

These results on episodic memory present a puzzle. Suppose it is assumed that the major knowledge difference between experts and novices is that experts have more schemas. This assumption is quite compatible with the finding that experts have better long-term episodic memory for problems and solutions. As Bartlett (1932) and many others have shown, stimuli that fit well into an existing schema are recalled better than stimuli that fit poorly. Since experts have more schemas than novices, chances are better that they can select a schema that fits the problems or solutions well, and hence, they will have better long term recall. However, it is not so easy to see how schemas facilitate short-term memory. This issue is so important that is has been given a subsection of it own, which follows this one.

Recall structures.

It is common to try to account for observed differences in episodic memory performance in terms of an underlying differences in the contents of the subjects' semantic memory. A standard technique for showing the influence of semantic memory contents on episodic memory performance is to use a stimulus consisting of several items, and allow the subject to recall the items in any order. Subjects often reorder the items from their original presentation order and recall them in runs of items, separated by pauses. The usual interpretation is that a run of items corresponds to the contents of an instantiated semantic memory structure. In particular, the longer the run, the larger the unit of semantic memory (e.g., Chase & Simon, 1973a). Thus, recall structure is important for determining the influence of semantic memory on episodic recall.

A common experiment is to contrast recall structure with some measure of semantic relatedness. Often, semantic relatedness is obtained by a classification task, such as asking experts to circle the stimulus items that go together (Reitman, 1976; Egan & Schwartz, 1979). Sometimes a copying task is used, where the subject glances back and forth between the stimulus array and a blank array, copying the items seen in the stimulus onto the response array (Chase & Simon, 1973a; Reitman, 1976). The items copied with each glance are interpreted as being semantically related. Another technique is to use an expert or textbook to obtain a list of important relationships that one item can have to another (e.g., in chess, whether one piece defends another). The semantic relatedness of two items can be equated with the number of relationships connecting them (Chase & Simon, 1973a).

In experiments of this sort, the the major finding is the recall *orders* can be predicted by the expertise and semantic relatedness of items, but the recall *pauses* cannot. In particular, for experts and not novices, items that have strong semantical relationships to each other are more likely to be recalled consecutively than items that have little semantical relationship (Chase & Simon, 1973a; Reitman, 1976; Egan & Schwartz, 1979; Engle & Bukstel, 1978). On the other hand, pause times do not correlate strongly with the degree of semantic relatedness (Reitman, 1976; Egan & Schwartz, 1979). Thus, item order seem to be a *function of the underlying knowledge structures*, but inter-item retrieval times do not. This finding turns out to play an important role in the discussion of the next subsection.

Expert-novice differences: chunking

It is well accepted that human perceptual processes are driven by knowledge in the form of *chunks*. (N.B., Earlier sections used "chunking" for the learning mechanism developed by Newell and his collaborators (Newell, 1987; Laird, Rosenbloom, & Newell, 1986; Laird, Newell, & Rosenbloom, 1987; Newell & Rosenbloom, 1981; Rosenbloom, 1983). This section uses the term as it is used in the general psychological literature.) For instance, an AI expert will perceive "SHRDLU" as a single chunk because it is the name of a famous AI program while non-experts will see it as a string of six letters. On the other hand, someone who is unfamiliar with the Roman alphabet will see it as a configuration of lines, because they do not have chunks for the letters. The chunking assumption is that the perceptual system will rapidly parse the stimulus, forming a hierarchical structure of instanstiated chunks that covers as much of the stimulus as possible given the set of chunks known by the subject. The result is a set of instantiated chunk trees. The roots of these chunk trees are what the subject "notices." Thus, the AI expert will have one tree/chunk, whose decendents are trees/chunks corresponding to each of the letters of "SHRDLU." The nonexpert will see only the six trees/chunks corresponding to the letters.

Chunks rose to prominence as the unit of measurement for memory capacity with Miller's (1956) hypothesis that short-term memory was limited to 7±2 chunks. Although Miller's simple hypothesis is no longer tenatable, chunks still play an important role in contemporary theories of short-term memory (Baddeley, 1986; Zhang & Simon, 1985) as well as other memory phenomena.

Chunks have played an important role in the development of theories of expert-novice differences. In particular, a leading hypothesis, first proposed by Chase and Simon (1973a), is that at least some of the second-order features of experts are chunks. Thus, an expert looking at a situation litterally *sees* more than a novice because the expert has more chunks. Chase and Simon pointed out that the hypothesis that experts have larger chunks than novices would explain the de Groot (1965) result that chessmasters could recall many more pieces from a briefly exposed chess position than novices. Assuming that both the novices and the experts have a short-term memory capacity of 7±2 chunks, if the experts have an average chunk size of 3 or more, they they could recall 20 or 30 chess pieces. On the other hand, if novices have only one piece per chunk, then they can recall only a few pieces.

In order to test this prediction of their hypothesis, Chase and Simon needed some independent measure of chunk size. They used recall structure, which was mentioned earlier. Unfortunately, they hypothesized that pauses represented the boundaries between chunks. By this measure, the chunk sizes of experts was only a little larger than novices (2.5 pieces vs. 1.9 pieces). Moreover, the experts recalled more chunks than the novices, contrary to the assumed constant capacity of short-term memory. The support for the Chase and Simon hypothesis was weakened further by Charness' (1976) demonstration that immediate memory for chess positions was not affected by the kinds of interference manipulations that were known to effect short-term memory for other types of stimulus material. Also, Reitman (1976) demonstrated that pauses were not a reliable indicator of chunk structure in the recall of Go positions, and Egan and Schwartz (1979) demonstrated that increased study time led to larger "chunks," as determined by pause structure. These results undermined the support for the chunk-size effect of expertise.

A few years later, Chase and Ericsson (1981) proposed a new explanation for the expert's short term memory. They showed that training could increase the apparent short-term memory capacity to 22 or more chunks. The primary device employed by subjects is a version of the venerable pattern of loci device used by mneumonists. The idea is to form a schema with specific slots that can be filled in with the stimulus material. The material can be recalled (in fact. in any order) by visiting the slots and reading out their contents. Chase and Ericsson named this device a *retrieval structure*. They showed that their digit-span expert's schema/retrieval structure was a specific 3-level tree whose 22 leaves consituted the slots in which stimulus material could be stored.

Chase and Ericsson hypothesized that the superior memory of chess masters and other experts is due to possession of schemas/retrieval structures. This hypothesis is consistent with the findings that familiar stimuli permitted the expert to exhibit superior memory (because they can be used to select and instantiate schemas) where as random stimuli do not. Moreover, the Chase-Ericsson hypothesis can be used to make sense of the Chase and Simon finding that expert's runs were only a little larger than novices, and that experts tended to have more runs than novices. If one assumes that pauses in recall protocols correspond to moving from one slot to another, then the number and size of the runs is a function of the instantiated retrieval structure, rather than the subject's chunks. The Chase-Erisson hypothesis is also consistent with the finding of Charness (1976) and Egan and Schwartz (1979) assumption that instantiated schemas are held in long-term memory rather than short-term memory.

The Chase-Ericsson hypothesis has thus far survived empirical challenges. It is consistent with all the major short-term memory findings in the expert-novice literature. Moreover, there is indeoendent evidence for schemas from several sources: (1) categorization studies (section), (2) protocol studies (section), and (3) learning mechanisms (section). Thus, it looks like schemas are the key to understanding expertise.

However, because the Chase-Ericsson hypothesis explains everything that the Chase-Simon hypothesis explains, there is currently no direct evidence that experts have larger chunks than novices. But it still remains an extremely plausible hypothesis, given all the evidence for chunking from experiments on verbal learning and perception (see chapter ??, this book).

Summary

Three ingredients of any future theory of problem solving have been presented. They are (1) the existing theory of problem solving in knowledge-lean task domains, (2) ideas for analysing expert problem solving in knowledge-rich task domains, and (3) some robust experimental findings. Comments on the contact between theory and findings were sprinkled throughout the preceding sections, so this summary can be mercifully brief. Table 6 lists the robust experimental findings, organized as they were presented in section 4. Table 7 lists most of the major theoretical concepts, organized as they were presented in sections 2 and 3.

Practice effects

1. Reduction of verbalization
2. Tactical learning
3. The power law of practice

Problem isomorphs

4. Varying the cover story does not affect difficulty
5. Other variations significantly affect difficulty

Transfer and problem solving by analogy

6. Asymmetric transfer
7. Negative transfer
8. Set effects
9. Spontaneous noticing of potential analogies is rare
10. Spontaneous noticing is based on supericial features

Expert-novice differences: problem solving studies

11. Experts perform faster than novices
12. Experts are more accurate than novices
13. Strategy differences
14. Self-monitoring

Expert-novice differences: memory studies

15. Classification of problems
16. Association structures
17. Episodic memory for problems and solutions
18. Recall structures

Expert-novice differences: chunking

19. Experts may have larger chunks than novices

Table 6: Robust empirical findings

The standard theory

Problem spaces
- States
- Operators

Understanding

Search
- Backup strategies vs. proceed strategies
- Heuristics
- Weak methods: forward and backwards chaining, operator subgoaling, etc.
- Means-ends analysis

Elaboration

Learning mechanisms
- Compounding
- Tuning
- Chunking
- Proceduralization
- Strengthening

Schema-driven problem solving

Schemas
- Problem half
- Solution half

Selection and instantiation
- Triggering
- Slot filling
- Second-order features

Following solution procedures
- Recursion
- Flexibility

Non-routine problem solving
- Search
- Schema compounding
- Impasses and repairs

Table 7: Major theoretical terms

References

Adelson, B. (1981). Problem solving and the development of abstract categones in programming languages. *Memory & Cognition, 9*(4), 422-433.

Akin, O. (1980). *Models of Architectural Knowledge.* London: Pion.

Anderson, J.R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369-406.

Anderson, J. R. (1983). *The Architecture of Cognition.* Cambridge, MA: Harvard.

Anderson, J. R., Farrell, R., & Saurers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-129.

Anzai, Y. & Simon, H.A. (1979). The theory of leaming by doing. *Psychological Review, 86*, 124-140.

Anzai, Y. & Yokoyama, T. (1984). Internal models in physics problem solving. *Cognition and Instruction, 1*, 397-450.

Atwood, M.E. & Polson, P.G. (1976). A process model for water jug problems. *Cognitive Psychology, 8*, 191-216.

Atwood, M.E., Masson, M.E.J. & Polson, P.G. (1980). Further explorations with a process model for water jug problems. *Memory & Cognition, 8*(2), 182-192.

Baddeley, A. (1986). *Working Memory.* Oxford, UK: Clarendon Press.

Bartlett, F.C. (1932). *Remembering: A Study in Experimental and Social Psychology.* Cambridge, UK: Cambridge University Press.

Bhaskar, R. & Simon, H. A. (1977). Problem solving in a semantically rich domains: An example from engineering thermodynamics. *Coginitve Science, 1*, 193-215.

Brown, J. S. & VanLehn, K. (1980). Repair Theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4*, 379-426.

Catrambone, R. & Holyoak, K.J. (1987). Transfer of training as a function of procedural variety of training examples. In *Proceedings of the Ninth Annual Conference.* Hillsdale, NJ: Cognitive Science Society,

Charness, N. (1976). Memory for chess positions; resistance to interference. *Journal of Experimental Psychology, 2*, 641-653.

Charness, N. (1979). Components of skill in bridge. *Canadian Joumal of Psychology, 33*(1), 1-16.

Charness, N. (1981). Search in chess: age and skill differences. *Journal of Experimental Psychology, 7*(2), 467-476.

Chase, W.G. & Ericsson, K.A. (1981). Skilled memory. In J.R. Anderson (Ed.), *Cognitive Skills and Their Acquisition*. Hillsdale, NJ: Erlbaum.

Chase, W.G. & Simon, H.S. (1973). Perception in chess. *Cognitive Psychology 5*, 55-81.

Chase, W.G. & Simon, H.A. (1973). The mind's eye in chess. In W.G. Chase (Ed.), *Visual Information Processing*. New York: Academic.

Chi, M. T. H. (1978). Knowledge Structures and Memory Development. In R. S. Siegler (Ed.), *Children's Thinking: What Develops?*. Hillsdale, NJ: Erlbaum.

Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P. & Glaser, R. (in press). Learning problem solving skills from studying examples. *Cognitive Science*, .

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*(2), 121-152.

Chi, M.T.H., Glaser, R. & Farr, M. (???). *The Nature of Expertise*. ???: ???

Chi, M. T. H., Glaser, R. & Rees, E. (1982). Expertise in problem solving. In R. J. Sternberg (Ed.), *Advances in the Psychology of Human Intelligence*. Hillsdale, NJ: Erlbaum.

Chiesi, H.L., Spilich, G.J. & Voss, J.F. (1979). Acquisition of domain-related information in relation to high and low domain knowledge. *Journal of Verbal Learning and Verbal Behavior, 18*, 257-273.

Crossman, E.R.F.W. (1959). A theory of the acqusition of speed-skill. *Ergonomics, 2*, 159-166.

de Groot, A.D. (1965). *Thought and Choice in Chess*. The Hague: Mouton.

Duncker, K. (1945). On problem-solving. *Psychological Monographs, 58*(270), 1-113.

Egan, D.E. & Schwartz, B.J. (1979). Chunking in recall of symbolic drawings. *Memory & Cognition, 7*(2), 149-158.

Engle, R.W. & Bukstel, L. (1978). Memory processes among bridge players of differing expertise. *American Journal of Psychology, 91*, 673-689.

Fikes, R.E., Hart, P.E. & Nilsson, N.J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence, 3*, 251-288.

Fitts, P.M. & Posner, M.I. (1967). *Human Performance*. Belmont, CA: Brooks/Cole.

Frey, P.W. & Adesman, P. (1976). Recall memory for visually presented chess positions. *Memory and Cognition, 4*, 541-547.

Gentner, D. (in press). Mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Ed.), *Similarity and Analogical Reasoning*. London: Cambridge University Press.

Gentner, D. & Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science, 10*, 277-300.

Gick, M.L. & Holyoak, K.J. (1980). Analogical problem solving. *Cogntive Psychology, 12*, 306-355

Gick, M. L. & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15*, 1-38.

Greeno, J.G. (1974). Hobbits and orcs: Acquisition of a sequential concept. *Cognitive Psychology, 6*, 270-292.

Greeno, J.G., Magone, M.E. & Chaiklin, S. (1979). Theory of constructions and set in problem solving. *Memory & Cognition, 7*, 445-461.

Hayes, J.R. (1981). *The Complete Problem Solver*. Philadelphia, PA: Franklin Institute Press.

Hayes, J. R. & Simon, H. A. (1974). Understanding written problem instructions. In L. W. Gregg (Ed.), *Knowledge and Cognition*. Hillsdale, NJ: Erlbaum. Reprinted in H.A.Simon (1979) Models of Thought, New Haven, CT: Yale University Press.

Hayes, J. R. & Simon, H. A. (1976). The understanding process: Problem Isomorphs. *Cognitive Psychology, 8*, 165-190. Reprinted in H.A.Simon (1979) Models of Thought, New Haven, CT: Yale University Press.

Hinsley, D.A., Hayes, J.R. & Simon, H.A. (1977). From words to equations meaning and representation in algebra word problems. In M.A.Just (Ed.), *Cognitive Processes in Comprehension*. Hillsdale, NJ: Erlbaum.

Holyoak, K. (1985). The pragmatics of analogical transfer. In G.H. Bower (Ed.), *The Psychology of Learning and Motivation*. New York: Academic.

Jeffries, R., Polson, P.G., Razran, L., & Atwood, M.E. (1977). A process model for missionaries-cannibals and other river-crossing problems. *Cognitive Psychology, 9*, 412-440.

Jeffries, R., Turner, A.A., Polson, P.G. & Atwood, M.E. (1981). The processes involved in designing software. In J.R. Anderson (Ed.), *Cognitive Skills and Their Acquisition*. Nillsdale, NJ: Erlbaum.

Kant, E. & Newell, A. (1984). Problem solving techniques for the design of algorithms. *Information Processing & Management, 20*(1-2), 97-118.

Kieras, D.E. & Bovair, S. (1986). The acquisition of procedures from text: A production-system analysis of transfer of training. *Journal of Memory and Language, 25*, 507-524.

Kieras, D. E. & Polson, P.G. (19??). An approach to the formal analysis of user complexity.

International Journal of Man-Machine Studies, ??, ??

Kotovksy, K. & Simon, H.A. (1973). Empirical tests of a theory of human acquisition of concepts for sequential patterns. *Cognitive Psychology*, *4*, 399-424.

Kotovsky, K., Hayes, J.R., & Simon, H.A. (1985). Why are some problems hard? Evidence from tower of Hanoi. *Cognitive Psychology*, *17*, 248-294.

Krutetskii, V.A. (1976). *The Psychology of Mathematical Abilities in Schoolchildren*. Chicago, IL: University of Chicago Press.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An Architecture for General Intelligence. *Artificial Intelligence*, *33*, 1-64.

Laird, J. E., Rosenbloom, P. S., and Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, *1*(1), 11-46.

Langley, P. (1987). A general theory of discrimination learning. In Klahr, D., Langley, P. & Neches, R. (Ed.), *Production System Models of Learning and Development*. Cambridge, MA: MIT Press.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve problems in physics. In J. R. Anderson (Ed.), *Cognitive Skills and their acquisition*. Hillsdale, NJ: Lawrence Erlbaum.

Larkin, J. H. (1983). The role of problem representation in physics. In D. Gentner and A. Collins (Ed.), *Mental Models*. Hillsdale, NJ: Lawrence Erlbaum.

Larkin, J.H. (1983). The role of problem representation in physics. In D. Gentner & A. Collins (Ed.), *Mental Models*. Hillsdale, NJ: Erlbaum.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science*, *208*, 1335-1342.

LeFevre, J. & Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, *3*(1), 1-30.

Levine, M. (1975). *A cognitive theory of learning: Research on hypothesis testing*. Hillsdale, NJ: Erlbaum.

Lewis, C. H. (1979). *Production system models of practice effects*. Doctoral dissertation, University of Michigan, Department of Psychology.

Lewis, C. (1981). Skill in algebra. In J. R. Anderson (Ed.), *Cognitive Skills and their Acquisition*. Hillsdale, NJ: Lawrence Erlbaum.

Luchins, A.S. (1942). Mechanization in problem solving. *Psychological Monographs*, Vol. *54*(248).

Mawer, R.F. & Sweller, J. (1982). Effects of subgoal density and location on learning during problem solving. *Journal of Educational Psychology, 8*(3), 252-259.

McDermott, J. & Larkin, J. H. (1978). Representing textbook physics problem. In *Proceedings of the 2nd National Conference.* Canadian Society for Computation Studies of Intelligence.

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

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

Neves, D.M. & Anderson, J.R. (1981). Knowledge compilation: Mechanisms for the automatization of cognitive skils. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition.* Hillsdale, NJ: Erlbaum.

Newell, A. (1980). Reasoning, problem solving and decision processes: The problem space as a fundamental category. In R. Nickerson (Ed.), *Attention and Performance VIII.* Hillsdale, NJ: Erlbaum.

Newell, A. (1987). Unified Theories of Cognition: The 1987 William James Lectures. On video tape, available from Harvard University, Dept. of Psychology.

Newell, A. & Rosenbloom, P.S. (1981). Mechanism of skill acquisition and the law of practice. In J.R. Anderson (Ed.), *Cognitive Skills and Their Acquisition.* Hillsdale, NJ: Erlbaum.

Newell, A. & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.

Newell, A., Shaw, J.C. & Simon, H.A. (1962). The process of creative thinking. In H.E. Gruber, G. Terrell & M. Wertheimer (Ed.), *Contemporary Approaches to Creative Thinking.* New York: Lieber-Atherton. Reprinted in H.A. Simon's (1979) *Models of Thought*, New Haven, CT: Yale Univeristy Press.

Ohlsson, S. (1984). Restructuring Revisited, II: An information processing theory of restructing and insight. *Scandinavian Journal of Psychology, 25*, 117-129.

Ohlsson, S. (1987). Truth versus appropriateness: Relating declarative to procedural knowledge. In D.Klahr, P.Langley & R. Neches (Ed.), *Production System Models of Learning and Development.* Cambridge, MA: MIT Press.

Paige, J. M., & Simon, H. A. (1966). Cognitive processes in solving algebra word problems. In B. Kleinmuntz (Ed.), *Problem solving: Research, method and theory.* New York, NY: Wiley.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas.*. New York, NY: Basic Books.

Pea, R. D. (????). The knowledge transfer problem. Unpublished manuscript, New York University.

Pennington, N. (1985). *Stimulus structures and mental representations in expert comprehension of computer programs* (Technical Report 2-ONR). Graduate School of Business, The University of Chicago.

Pirolli, P.L. (1985). *Problem solving by analogy and skill acquisition in the domain of programming.* Doctoral dissertation, Carnegie-Mellon University, Dept. of Psychology,

Pirolli, P.L. & Anderson, J.R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology, 39*(2), 240-272.

Reed, S.K. (1987). A structure-mapping model for word problems. *Journal of Experimental Psychology: Learning, Memory and Cognition, 13*(1), 124-139.

Reed, S.K. & Simon, H.A. (1976). Modelling strategy shifts in a problem-solving task. *Cognitive Psychology, 8*, 86-97.

Reed, S. K., Dempster, A. & Ettinger, M. (1985). Usefulness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory and Cognition, 11*, 106-125.

Reed, S. K., Ernst, G. W. & Banerji, R. (1974). The role of analogy in transfer between similar problem states. *Cognitive Psychology, 6*, 436-450.

Reitman, W.R. (1965). *Cognition and Thought: An Information-Processing Approach.* New York: Wiley.

Reitman, J.S. (1976). Skilled perception in Go: Deducing memory structures from inter-response times. *Cognitive Psychology, 8*, 336-356.

Reimann, P. & Chi, M.T.H. (????). Human expertise in complex problem solving. In Gilhooly (Ed.), *Human and Machine Problem-Solving.* ???

Rosenbloom, P.S. (1983). *The chunking of goal hierarchies: A model of practice and stimulus-response compatibility.* Doctoral dissertation, Carnegie-Mellon University, Available as CMU Computer Science Technical Report #83-148.

Ross, B.H. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology, 16*, 371-416.

Ross, B.H. (1987). This is like that: the use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*(4), 629-639.

Schneider, W. & Shiffrin, R.M. (January 1977). Controlled and automatic human information processing: I. detection, search and attention. *Psychological Review, 84*(1), 1-66.

Schoenfeld, A.H. (1981). Episodes and executive decisions in mathematical problem solving. Paper presented at the 1981 AERA Annual Meeting, Los Angeles, Ca.

Schoenfeld, A.H. & Herrmann, D.J. (1982). Problem perception and knowledge structure in expert and novice mathematical problem solvers. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 8*(5), 484-494.

Schvaneveldt, R.W., Durso, F.T., Goldsmith, T.E., Breen, T.J. & Cooke, N.M. (1985). Measuring the structure of expertise. *International Journal Man-Machine Studies, 23*, 699-728.

Shiffrin, R.M. & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review, 84*(2), 127-190.

Silver, E.A. (1979). Student perceptions of relatedness among mathematical verbal problems. *Journal for Research in Mathematics Education, 10*, 195-210.

Silver, E. A. (1981). Recall of mathematical problem information: Solving related problems. *Journal for Research in Mathematical Education, 12*, 54-64.

Simon, H.A. (1973). The structure of ill-structured problems. *Artificial Intelligence, 4*, 181-201.

Simon, H. A. (1983). Why should machines learn? In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto, CA: Tioga.

Simon, D. P. & Simon, H. A. (1978). Individual difference in solving physics problems. In R. Siegler (Ed.), *Children's thinking: What Develops?*. Hillsdale, NJ: Erlbaum.

Singley, M.K. (1986). *Developing models of skill acquisition in the context of intelligent tutoring systems*. Doctoral dissertation, Department of Psychology, Carnegie-Mellon University,

Singley, M.K. & Anderson J.R. (1985). The transfer of text-editing skill. *International Journal of Man-Machine Studies, 22*, 403-423.

Singley, M.K. & Anderson, J.R. (19??). A keystroke analysis of learning and transfer in text editing. submitted.

Stevens, S.S. (1951). *Handbook of Experimental Psychology*. New York: Wiley.

Sweller, J. (1983). Control mechanisms in problem solving. *Memory & Cognition, 11*(1), 32-40.

Sweller, J. & Cooper, G.A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*(1), 59-89.

Sweller, J. & Gee, W. (1978). Einstellung, the sequence effect and hypothesis theory. *Journal of Experimental Psychology: Human Learning and Cognition, 4*, 513-526.

Sweller, J. & Levin, M. (1982). Effects of goal specificity on means-ends analysis and learning. *Journal of Experimental Psychology: Learning, Memory and Cognition, 8*(5), 463-474.

Sweller,J., Mawer, R.F. & Ward, M.R. (1983). Development of Expertise in Mathematical Problem Solving. *Journal of Experimental Psychology: General, 112*(4), 639-661.

Thorndike, E.L. & Woodworth, R.S. (1901). The influence of improvement in one mental function upon the efficiency of other functions. *Psychological Review, 8*, 247-261.

Tulving, E. (1972). Episodic and semantic memory. In E.Tulving & W.Donaldson (Ed.), *Organization and Memory*. New York: Academic Press.

VanLehn, K. (1982). Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. *The Journal of Mathematical Behavior, 3*(2), 3-71.

VanLehn, K. (1983). Human skill acquisition: Theory, model and psychological validation. In *Proceedings of AAAI-83*. Los Altos, CA: Morgan Kaufman,

VanLehn, K. (1987). Learning one subprocedure per lesson. *Artificial Intelligence, 31*(1), 1-40.

VanLehn, K. (in press). *Cognitive Procedures: The Acquisition and Representation of Basic Mathematical Skills*. Cambridge, MA: MIT Press.

VanLehn, K. & Ball, W. (1987). *Flexible execution of cognitive procedures* (Technical Report PCG-5). Carnegie-Mellon University, Dept. of Psychology.

Voss, J.F., Greene, T.R., Post, T.A. & Penner, B.C. (1983). Problem solving skill in the social sciences. In G.H.Bower (Ed.), *The Psychology of Learning and Motivation, Vol.17*. New York: Academic Press.

Voss, J.F., Tyler, S.W. & Yengo, L.A. (1983). Individual differences in the solving of social science problems. In R. Dillon & R. Schmech (Ed.), *Individual Differences in Cognition*. New York: Academic Press.

Weisberg, R.W. & Alba, J.W. (1981). An examination of the alleged role of 'fixation' in the solution of several 'insight' problems. *Journal of Experimental Psychology: General, 110*(2), 169-192.

Weiser, M. & Shertz, J. (1983). Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies, 19*, 391-398.

Zhang, G. & Simon, H.A. (1985). STM capacity for Chinese words and idioms: Chunking and acoustical loop hypotheses. *Memory and Cognition, 13*(3), 193-201.

Notes

[1]Standard techical terms in the field are italicized when they are introduced.

[2]Sequence extrapolation has the expository advantage of being a simple task domain that most readers are familiar with. However, it is not a knowledge-lean task domain because subjects are usually not told what types of patterns are legal. Thus, the subjects must use their common sense and/or their prior experience with the task in order to decide what kinds of patterns are legal, and hence, what kinds of states and operators to use in the problem space. For sequence extrapolation, the understanding process is just as important as the search process. See Kotovsky and Simon (1973) for a serious treatment of this task domain.

[3]Many specific models of problem solving in the literature do not distinguish between assertions generated by perception and assertions generated by inference. The models sometimes produce a state with several dozen assertions in it, and this worries students who are familiar with the limitations of human short term memory. However, some of these assertions may *represent information* that does not reside in the subject's short term store. Rather, it is information that the subject once saw in the external environment and could easily see again just by directing their gaze to the appropriate location. Although the limitations of short term memory obviously do place some constraints on human problem solving, it would be far to simple equate the contents of a problem state with the contents of short term memory. Section 4.6 discusses the issue of short-term memory limitations in more detail.

[4]In principle, the initially available states could be much larger. The problem statement could mention final states or have hints that mention intermediate states. The subjects could even derive intermediate states deductively from their state representation language.

[5]Often, courses on human problem solving include discussions of the major types of state space search algorithms, such as depth-first search, breadth-first search or even A*. Although it is doubtful that these types of search occur in human performance, they are nonetheless an important part of the conceptual vocabulary of a well-trained cognitive scientist. Fortunately, these terms are almost always covered in introductory courses on AI, so a discussion of them has been omitted in this chapter.

[6]Backwards chaining and operator subgoaling are similar and often confused. However, backwards chaining computes with concrete problem states, while operator subgoaling computes with descriptions of desired problem states. Backwards chaining requires invertible operators, but operator subgoaling does not.

[7]A special case of this strategy is called *hillclimbing*. It is applicable when the differences between the current and desired states can be measured numerically. In this case, the heuristic simply choose the operator that minimizes that numerical distance.

[8]The algebraic examples used in this section have the advantage that they come from a task domain that most readers know quite well, so they make easily understood illustrations. However, much less is known about specific schemas in algebra than in physics, where more work with detailed computer simulations has been done. No claim about the actual existence of this particular algebraic schema or the others mentioned herein are intended.

[9]The missionaries and cannibals puzzle is: "Three missionaries and three cannibals wish to cross a river. There is a boat, but it holds only two people. Find a schedule of crossings that will permit all six people to cross the river in such a way that at no time do the cannibals outnumber the missionaries on either bank." This version of the puzzle, with three missionaries, three cannibals and a boat that holds two, is the most common. Other versions vary the number of people, the size of the boat and other constraints. The mathematics of river crossing puzzles is explored by Fraley, Cooke and Detrick (1966) and others.

[10]Although the knowledge is notated as productions, Kieras and Singley have argued that the type of knowledge transfered in some experiments is actually declarative, rather than procedural, because these subjects had too little practice to allow them to proceduralize their declarative knowledge before the transfer task was given.

[11]This finding must be qualified slightly for some domains, such as political science, where there is no objective measure of solution correctness or quality, so the experts' solutions are defined to be the correct ones.