COGNITIVE SCIENCE 14, 511-550 (1990)

Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry

KENNETH R. KOEDINGER AND JOHN R. ANDERSON
Carnegie-Mellon University

We present a new model of skilled performance in geometry proof problem solving called the Diagram Configuration model (DC). While previous models plan proofs in a step-by-step fashion, we observed that experts plan at a more abstract level: They focus on the key steps and skip the less important ones. DC models this abstract planning behavior by parsing geometry problem diagrams into perceptual chunks, called diagram configurations, which cue relevant schematic knowledge. We provide verbal protocol evidence that DC's schemas correspond with the step-skipping inferences experts make in their initial planning. We compare DC with other models of geometry expertise and then, in the final section, we discuss more general implications of our research. DC's reasoning has important similarities with Larkin's (1988) display-based reasoning approach and Johnson-Laird's (1983) mental model approach. DC's perceptually based schemas are a step towards a unified explanation of (1) experts' superior problem-solving effectiveness, (2) experts' superior problem-state memory, and (3) experts' ability, in certain domains, to solve relatively simple problems by pure forward inferencing. We also argue that the particular and efficient knowledge organization of DC challenges current theories of skill acquisition as it presents an end-state of learning that is difficult to explain within such theories. Finally, we discuss the implications of DC for geometry instruction.

Detailed study of successful performance in difficult task domains can provide a strong basis for understanding the processes of problem solving and the nature of thought in general. To become an expert in a difficult field

---

This research was supported by grant 86-17083 from the National Science Foundation and by a United States Air Force Graduate Fellowship to the first author. Thanks to Jill Larkin, Kurt VanLehn, and Herbert Simon for insightful counsel. Thanks also to Christina Allen, Stephen Casner, Lael Schooler, editor Jim Greeno and the reviewers: Jeremy Kilpatrick, Robert Lindsay, and an anonymous cognitive psychologist for their helpful comments and advice. Portions of this work were reported at the 1989 meeting of the American Educational Research Association in San Francisco, CA and at the 1989 meeting of the Cognitive Science Society in Ann Arbor, MI.

Correspondence and requests for reprints should be sent to Ken Koedinger, Psychology Department, Carnegie-Mellon University, Pittsburgh, PA, 15213.

calls upon the full adaptive and flexible nature of human intelligence. Characterizing the adaptive processes that bring about the acquisition of expertise is an important goal of cognitive science and much progress has been made in creating general mechanisms of skill acquisition and learning (Anderson, 1983; Holland, Holyoak, Nisbeth, & Thagard, 1986; Newell, in press). However, progress in these areas is limited by the depth and accuracy of theories of expertise. Some of our efforts towards developing a general mechanism that models human learning may be wasted if we do not have an accurate and detailed understanding of the end state such mechanisms are designed to reach.

Newell and Simon (1972) pioneered the use of verbal protocol analysis and computer simulation as complementary tools for the study of successful performance in difficult task domains and the identification of the mechanisms behind such performance. The analysis of verbal reports given by subjects as they solve problems can provide both initial ideas for proposing a workable mechanism and empirical evidence to support one. In a complementary way, computer simulation provides a test of both the coherence and sufficiency of a proposed mechanism.

In this article we present verbal report data and a computer simulation of geometry proof problem solving. This domain is a difficult one for human problem solvers and has been studied by a number of cognitive science researchers (Anderson, Greeno, Kline, & Neves, 1981; Gelernter, 1963; Greeno, 1978; Nevins, 1975). We were motivated to take another look at this domain by the observation that skilled problem solvers are able to focus on key problem solving steps and skip minor ones in the process of generating a solution plan. We found a surprising regularity in the kinds of steps expert subjects skipped and built a computer model, called DC, to account for this regularity.

1. THE EXECUTION SPACE OF GEOMETRY

Geometry proof problem solving is hard. For a typical geometry proof, the search space of possible geometry rule applications (i.e., theorems, definitions, and postulates) is quite large. Problem 7 in Figure 1 is a typical high school geometry proof problem. At the point in the high school curriculum where this problem is introduced there are 45 possible inferences that can be made from the givens of this problem, from these inferences another 563 inferences can be made, from these more than 100,000 can be made.

While it is true, as Newell and Simon (1972) pointed out, that there are multiple possible problem spaces for any problem, there is one problem space for geometry which is perhaps the most natural extension of the way geometry is typically taught. This problem space is the one analyzed above and has the definitions, postulates, and theorems of geometry as operators.

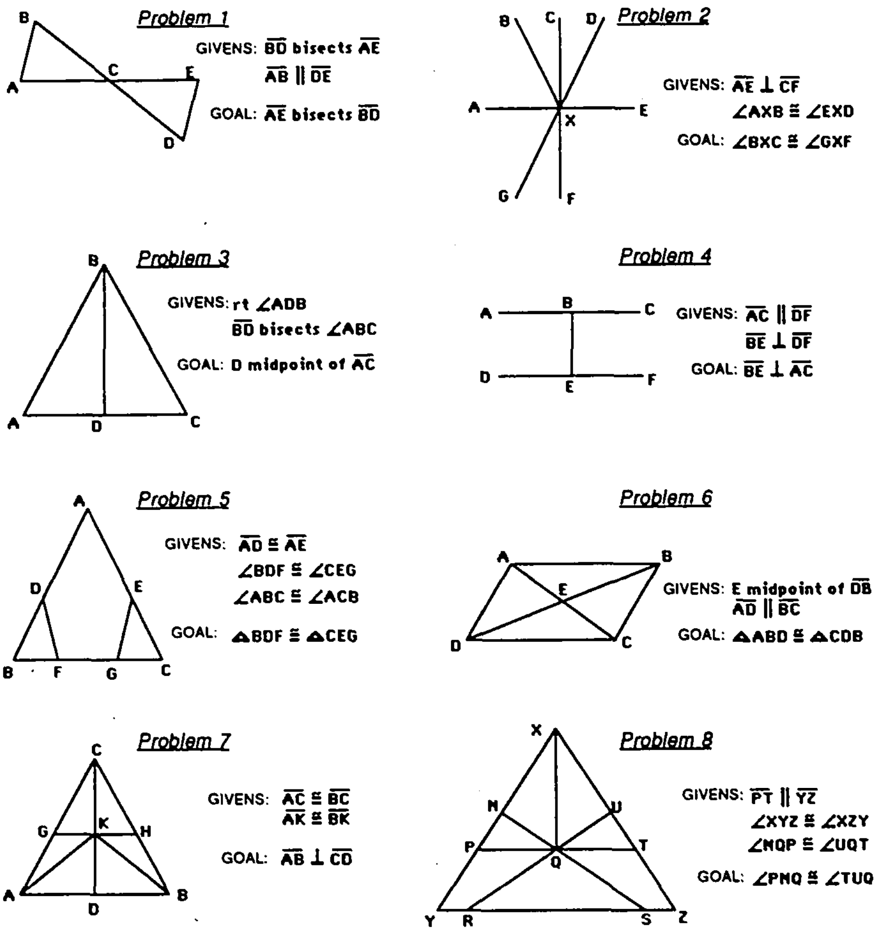

Problem 1
GIVENS: $\overline{BD}$ bisects $\overline{AE}$
$\overline{AB} \parallel \overline{DE}$
GOAL: $\overline{AE}$ bisects $\overline{BD}$
Problem 2
GIVENS: $\overline{AE} \perp \overline{CF}$
$\angle AXB \cong \angle EXD$
GOAL: $\angle BXC \cong \angle GXF$
Problem 3
GIVENS: rt $\angle ADB$
$\overline{BD}$ bisects $\angle ABC$
GOAL: D midpoint of $\overline{AC}$
Problem 4
GIVENS: $\overline{AC} \parallel \overline{DF}$
$\overline{BE} \perp \overline{DF}$
GOAL: $\overline{BE} \perp \overline{AC}$
Problem 5
GIVENS: $\overline{AD} \cong \overline{AE}$
$\angle BDF \cong \angle CEG$
$\angle ABC \cong \angle ACB$
GOAL: $\triangle BDF \cong \triangle CEG$
Problem 6
GIVENS: E midpoint of $\overline{DB}$
$\overline{AD} \parallel \overline{BC}$
GOAL: $\triangle ABD \cong \triangle CDB$
Problem 7
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$
GOAL: $\overline{AB} \perp \overline{CD}$
Problem 8
GIVENS: $\overline{PT} \parallel \overline{YZ}$
$\angle XYZ \cong \angle XZY$
$\angle NQP \cong \angle UQT$
GOAL: $\angle PNQ \cong \angle TUQ$

Figure 1. Geometry problems given to subjects and solved by DC.

We call it the *execution space* of geometry because the operators correspond with the steps that a problem solver writes down in the solution of a problem.

As illustrated above, the geometry execution space is enormous. In the DC model described below, we achieve search control by initially planning a solution sketch in a problem space that is more abstract, that is, more compact, than the execution space. In contrast, the traditional approach has been to look for better search strategies and heuristics to use within the execution space. Gelernter's (1963) geometry theorem proving machine used a backward search strategy in the execution space and used the diagram as a pruning heuristic. More recently, the second author and colleagues (Anderson, Boyle, & Yost, 1985) built a geometry expert system as a cognitive

TABLE 1
A Verbal Protocol for a Subject Solving Problem 3

	Planning Phase
B1: We're given a right angle—this is a right angle,	Reading given: **rt $\angle$ADB**
B2: perpendicular on both sides [makes perpendicular markings on diagram];	Inference step 1: **AC$\perp$BD**
B3: BD bisects angle ABC [marks angles ABD and CBD]	Reading given: **BD bisects $\angle$ABC**
B4: and we're done.	Inference step 2: **$\triangle$ABD$\cong$$\triangle$CBD**
	Execution Phase
B5: We know that this is a reflexive [marks line BD],	In this phase, the subject refines and explains his solution to the experimenter.
B6: we know that we have congruent triangles; we can determine anything from there in terms of corresponding parts	
B7: and that's what this [looking at the goal statement for the first time] is going to mean. . . that these are congruent [marks segments AD and DC as equal on the diagram].	

model of students and a component of an intelligent tutoring system. The Geometry Tutor expert (GTE) used an opportunistic or best-first bidirectional search strategy in the execution space and used various contextual features as heuristics for predicting the relevance of an operator. (We review these systems and a few others in Section 5.) While GTE provided a reasonably good model of students, as evidenced by the success of the Geometry Tutor (Anderson, Boyle, Corbett, & Lewis, 1990), we found that the mode of attack by human experts was distinctly different from that of GTE. It seemed important to be able to characterize this expertise both as a goal in and of itself, and for pedagogical purposes.

2. EXPERT HUMAN PROBLEM SOLVING

2.1 Step Skipping and Abstract Planning

One feature that distinguishes geometry experts is that they do not make all the steps of inference that students do while developing a solution plan. Consider the protocol in Table 1 of an expert (Subject R) solving Problem 3 shown in Figure 1. The left side of the table contains the protocol and the right side indicates our coding of the subject's actions.

This expert had a reliable solution sketch for this problem in 13 seconds at the point where he said, *"we're done."* He plans this solution sketch without looking at the goal statement (more on this curious behavior in Section 4.3) and in the remainder of the protocol he elaborates the solution sketch, reads the goal statement, and explains how it is proven. His words "we're done" indicate his realization that the two triangles ABD and CBD are congruent and that therefore he knows everything about the whole

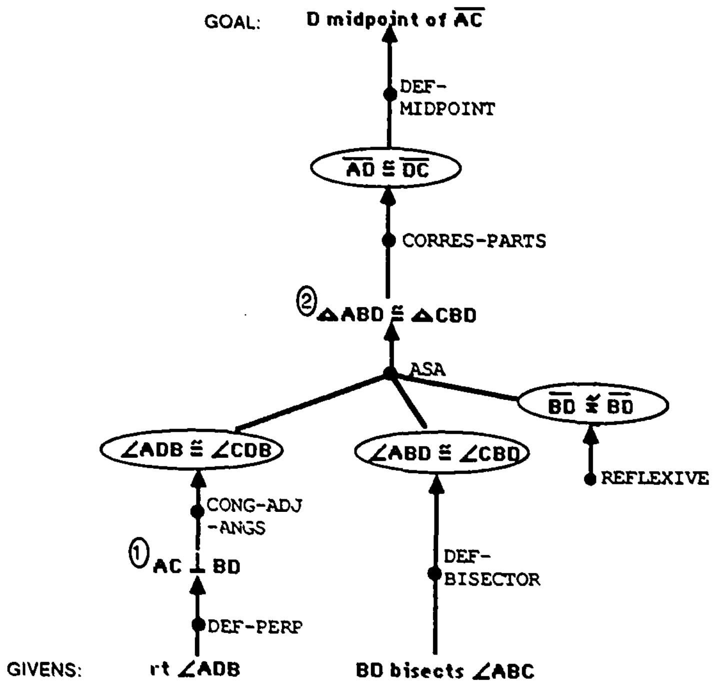


Figure 2. The final solution for Problem 3. The givens of the problem are at the bottom and the goal is at the top. The lines represent inferences with the conclusion at the arrow head, the premises at the tails, and the justifying geometry rule at the dot in between. The statements Subject R mentioned during planning (see Table 1) are numbered while the ones he skipped are circled.

problem. As he explains later, "we can determine anything from there in terms of corresponding parts."

Figure 2 shows the solution to the problem in the proof tree notation of the Geometry Tutor. Apart from the givens and goal, the statements which the expert mentioned while solving this problems are numbered in Figure 2, and the skipped steps are circled. Assuming this expert's verbalizations accurately reflect his working memory states (Ericsson & Simon, 1984), we conclude that the expert only makes certain key inferences in his search for a solution while skipping other, apparently minor inferences.

2.1.1. Abstraction. In the terminology of the problem-solving literature, it seemed clear that experts were initially planning their proof in an *abstract problem-solving space* (Newell & Simon, 1972; Sacerdoti, 1974; Unruh, Rosenbloom, & Laird, 1987). They were ignoring certain distinctions, such as the distinction between congruence and equality, and they were skipping over certain kinds of inferences, particularly the algebraic inferences. It turns out that ignoring the algebraic inferences considerably reduces the size of the search space. We establish this fact in the analysis of the model below by comparing the size of the execution space for Problem 7 with and without the algebraic inferences (see Section 4.1).

We distinguish the two types of abstract planning, risky and safe. *Risky abstraction* is a type of abstraction where details can be ignored that are sometimes critical to arriving at a correct solution. Newell and Simon (1972) showed that during planning, subjects solving logic problems would often ignore certain aspects of the expressions they were working with. This abstraction was often very effective in guiding their problem-solving search. However, sometimes subjects failed to refine an abstract plan successfully because one of the details ignored in the abstraction process turned out to be critical.

A *safe abstraction* only ignores irrelevant details, that is, details which only discriminate between objects that are functionally equivalent with respect to the problem-solving task. For example, in ignoring the details that distinguish between congruence statements (e.g., **AB ≅ CD**) and measure equality statements (e.g., **mAB = mCD**) geometry problem solvers are performing a safe abstraction since these statements are equivalent with respect to making proof inferences. Any inference that can be made from one can be made from the other.

2.1.2 Macro-Operators. In addition to performing useful abstractions, expert problem solvers have been characterized by the fact that they often collapse multiple problem-solving steps into a single step (Anderson, 1983; Larkin, McDermott, Simon, & Simon, 1980b). In the field of problem solving this is known as the formation of macro-operators (Korf, 1987; Nilsson, 1980). Macro-operators are the chunking together of a sequence of operators which are often used consecutively to achieve a particular goal. Although geometry experts appear to have certain macro-operators, these operators are not just arbitrary compositions of geometry rules which can be used in sequence. Rather, there is a regularity in the kinds of macro-operators experts have. Not only does the same expert skip the same kinds of steps on different occasions, but different experts appear to skip the same kinds of steps in similar situations.

In summary, we found that experts were not planning solutions in the execution space as previous models have. In addition, it appeared that expert's planning space could not be accounted for by a straightforward application of standard learning mechanisms to the execution space. Typical abstraction methods lead to risky abstractions, while experts' abstractions were safe. Typical macro-operator learning methods do not predict the kind of regularity in step skipping that we found of the experts. Thus, we were led to search for a new problem space for geometry theorem proving, one that was a safe abstraction of the execution space and left out the same kind of steps the experts did.

2.2 Use of the Diagram

Besides not working in the execution space, experts' inference making was largely tied to the diagram. We found that the regularity in experts' step skipping can be captured by knowledge structures that are cued by images in the problem diagram. In contrast, execution space inferences are cued off the known and desired statements in the problem. Larkin and Simon (1987) suggest two reasons why diagrammatic representations might be critical to problem solving in domains like geometry. First, one can use *locality* of objects in the diagram to direct inference, and second, perceptual inferences can be made more easily than symbolic inferences.

Consider their point about locality first. A familiar strategy of high school geometry students is to record proof steps by marking the problem diagram as an alternative to writing them down in statement notation. Such an annotated diagram aids students in holding together information they need to make further inferences. (This is even true if they do not explicitly mark the diagram, as long as they think in terms of it.) In contrast, information within a list of written statements may be visually separated and require search to identify. For instance, to use the side-angle-side rule for inferring triangle congruence, a problem solver must locate three congruence relationships: two between corresponding sides of the triangles and one between corresponding angles. In searching a list of statements for these three relationships, one might need to consider numerous possible combinations of three statements that exist in the list. However, if these relationships are marked on a diagram, one can quickly identify them since the side-angle-side configuration comes together in each triangle at a single vertex. In other words, related information is often easier to find in a diagram because it is typically in the same locality whereas the same information may be separated in a list of statements. This is the *locality feature* of diagrams.

The example above illustrates the role of the diagram in aiding knowledge search, that is, the search for applicable knowledge. The geometry diagram can also be used to aid problem search, that is, the search for a problem solution.[1] The idea is that images in the diagram can be used to cue chunks of knowledge which serve as operators in an abstract planning space. The notion that external representations can play a major role in guiding problem solving is the central notion of Larkin's (1988) display-based reasoning approach. Our approach elaborates on this one by showing how the organization of an external representation can be used to cue abstract planning

[1] See Chapter 2 in Newell (in press) for more discussion on the distinction between knowledge search and problem search.

operators. These abstract operators reduce problem search by packing many execution steps into a single inference.

Larkin and Simon's (1987) second point, that diagrams allow easy perceptual inferences to replace hard symbolic ones, is based on an assumption that perceptual inferences are generally easier than symbolic inferences. While we agree with this assumption, we believe it is unlikely that perceptual inferences are somehow inherently easier (except in terms of the locality feature noted above). Rather, it is possible that perceptual inferences appear easier because, in general, they have been much more highly practiced than symbolic inferences. Nevertheless, since it is likely that students of geometry have had more prior experience with geometric images than with formal notations, and since diagrams typically have the locality feature, students are likely to find perceptual inferences in this domain easier.

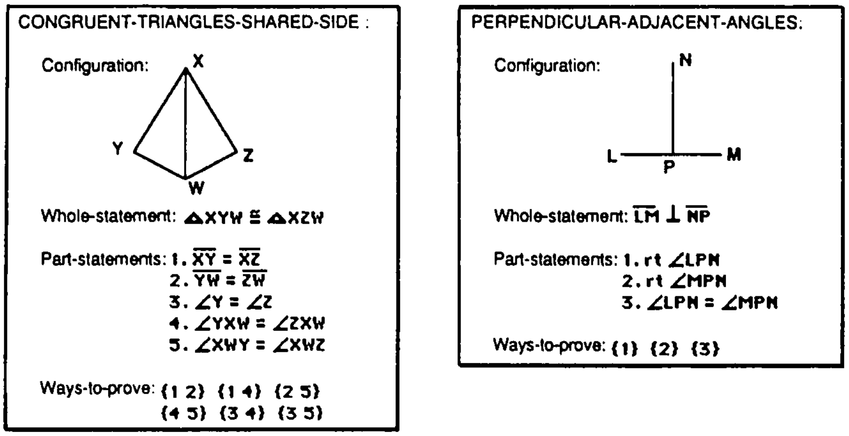
3. THE DIAGRAM CONFIGURATION MODEL

Based on our observations of experts, we tried to design a system for geometry theorem proving that would be both more powerful and more like human experts than previous systems. The model we came up with, the Diagram Configuration model (DC), has one major knowledge structure, diagram configuration schemas, and three major processes: diagram parsing, statement encoding, and schema search. Section 3.1 describes DC's diagram configuration schemas, while Section 3.2 describes DC's processing components. Section 3.3 describes how DC uses a special class of diagram configuration schemas to avoid difficult algebra subproofs.

3.1 Diagram Configuration Schemas

The core idea of the DC model is that experts have their knowledge organized according to diagrammatic schemas which we call *diagram configuration schemas*. These are clusters of geometry facts that are associated with a single prototypical geometric image. Figure 3 shows two diagram configuration schemas.

The *whole-statement* and *part-statements* attributes of a schema store the facts which are associated with the geometric image stored in the *configuration* attribute. The configuration is a prototypical configuration of points and lines which is commonly a part of geometry diagrams. In Figure 3, the configuration on the left is a prototype for any set of lines that form two triangles with a side in common. The whole-statement is the geometry statement which refers to the configuration as a whole. The part-statements refer to relationships among the parts of the configuration. The whole-statement of the CONGRUENT-TRIANGLES-SHARED-SIDE schema refers to the two triangles involved while the part-statements refer to the corresponding sides and angles of these triangles. The *ways-to-prove* are used to determine



CONGRUENT-TRIANGLES-SHARED-SIDE :

Configuration:

Whole-statement: $\triangle XYW \cong \triangle XZW$

Part-statements: 1. $\overline{XY} = \overline{XZ}$
2. $\overline{YW} = \overline{ZW}$
3. $\angle Y = \angle Z$
4. $\angle YXW = \angle ZXW$
5. $\angle XWY = \angle XWZ$

Ways-to-prove: {1 2} {1 4} {2 5} {4 5} {3 4} {3 5}

PERPENDICULAR-ADJACENT-ANGLES:

Configuration:

Whole-statement: $\overline{LM} \perp \overline{NP}$

Part-statements: 1. rt $\angle LPN$
2. rt $\angle MPN$
3. $\angle LPN = \angle MPN$

Ways-to-prove: {1} {2} {3}

Figure 3. Two examples of diagram configuration schemas. The numbers in the ways-to-prove indicate part-statements. Thus, in the CONGRUENT-TRIANGLES-SHARED-SIDE schema {1 2} means that if the part-statements **XY = XZ** and **YW = ZW** are proven, all the statements of the schema can be proven.

whether inferences can be made about a configuration. They indicate subsets of the part-statements which are sufficient to prove the whole-statement and all of the part-statements. For example, the first way-to-prove of the CONGRUENT-TRIANGLES-SHARED-SIDE schema, {12}, indicates that if the part-statements **AB = AC** and **BD = CD** have been proven, the schema can be proven, that is, all the other statements of the schema can be proven.

Our basis proposal is that planning is done in terms of these schemas rather than the statements of geometry. The problem solver tries to establish that various schemas are true of the diagram. Establishing one schema may enable establishing another. Because there are a small number of schemas possible for any particular problem diagram, the search space of schemas is much smaller than the execution space.

Consider Problem 3 and the expert protocol in Table 1. In the planning phase, the subject made four verbalizations. Of these four verbalizations, two indicate his reading and encoding of the given statements and two indicate inferences. Essentially, the subject solved the problem in two steps. In contrast, the complete execution space solution (see Figure 2) requires seven geometry rule applications. In other words, a problem solver who was planning in the execution space would take at least seven steps to solve this problem. DC's solution to this problem, like the subject's, is much shorter: It involves only two schemas. An instance of the PERPENDICULAR-ADJACENT-ANGLES schema can be established from the givens of the problem, while an instance of the TRIANGLE-CONGRUENCE-SHARED-SIDE schema can be established as a result. We now describe the processes DC uses to recognize and establish schemas.

3.2 DC's Processing Components

DC has three major processing stages:

1. *Diagram parsing* in which it identifies familiar configurations in the problem diagram and instantiates the corresponding schemas;
2. *Statement encoding* in which it comprehends given and goal statements by canonically representing them as part-statements; and
3. *Schema search* in which it iteratively applies schemas in forward or backward inferences until a link between the given and goal statements is found.

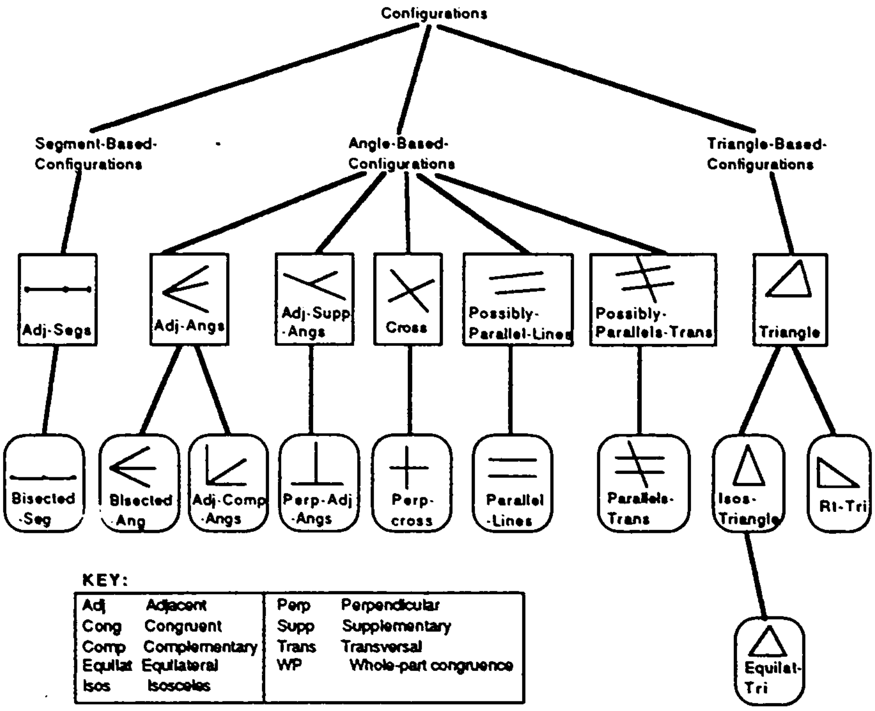
Human experts integrate these processes so that they do not occur in any fixed order except to the extent that some statement encoding and diagram parsing have to be done before any schema search can begin. However, in the computer simulation each process is done to completion before the next begins. We implemented these processes as separate stages so that we could independently evaluate the role each has in reducing search relative to planning in the execution space. It turns out that the diagram-parsing process plays a major role as we describe below.

3.2.1 Diagram Parsing and Schema Instantiation. Diagram parsing is the process of recognizing configurations in geometry diagrams and instantiating the corresponding schemas. Diagram parsing consists both of a low-level component which recognizes simple geometric objects and a higher level inductive component which hypothesizes plausible diagram configurations.

The DC simulation starts with a very simple point and line representation of a problem diagram. From this representation it must recognize line segments, angles, and triangles, and construct an internal representation of each. In addition, the algorithm records approximate size measures of the segments and angles it identifies.

Using the information created by this low-level object recognition process, DC looks for instances of abstract configurations. Figures 4a and 4b illustrate the diagram configurations for proof problems in a typical course up to and including the topic of triangle congruence. In some cases an image in a problem diagram may appear to be an instance of a known diagram configuration, but may not actually be an instance because it is not properly constrained by the givens of a problem. On the other hand, some configurations do not need to be constrained by the problem givens to be a diagram configuration instance. These are called *basic configurations* and appear in the square-cornered boxes in Figure 4a.[2]

[2] As you might notice from looking at some of the basic configurations, DC assumes that points which appear collinear (on the same line) in a problem diagram actually are collinear. This assumption is commonly made in high school classrooms and we told our subjects they could make it in the problems we gave them.



Adj	Adjacent	Perp	Perpendicular
Cong	Congruent	Supp	Supplementary
Comp	Complementary	Trans	Transversal
Equilat	Equilateral	WP	Whole-part congruence
Isos	Isosceles		

Figure 4a. The diagram configurations for geometry up to and including the topic of triangle congruence. The configurations in rectangles are basic configurations which can be recognized immediately in problem diagrams. The other configurations are specializations of these in which certain relationships appear to hold among the parts of the configuration.

DC uses the low-level object information to recognize instances of the basic configurations. The other configurations are either specializations of the basic ones (and thus are attached below them in Figure 4a) or specializations of pairs of basic configurations (see Figure 4b). To recognize possible specializations, DC uses the segment and angle size approximations to check whether any of the basic configurations have the necessary properties to be specialized. For example, to recognize the ISOSCELES-TRIANGLE configuration, DC checks the triangles it has identified to see if any have two equal sides.

DC's diagram-parsing algorithm corresponds with a very powerful visual process in humans. We make no claims that the internals of this algorithm match the internals of the corresponding human process. For instance, while it is quite likely that human perceptual processes make extensive use of symmetry in recognizing geometric images, DC makes no use of symmetry. We do claim that human experts are capable of recognizing these configurations and make extensive use of this ability in solving proof problems.

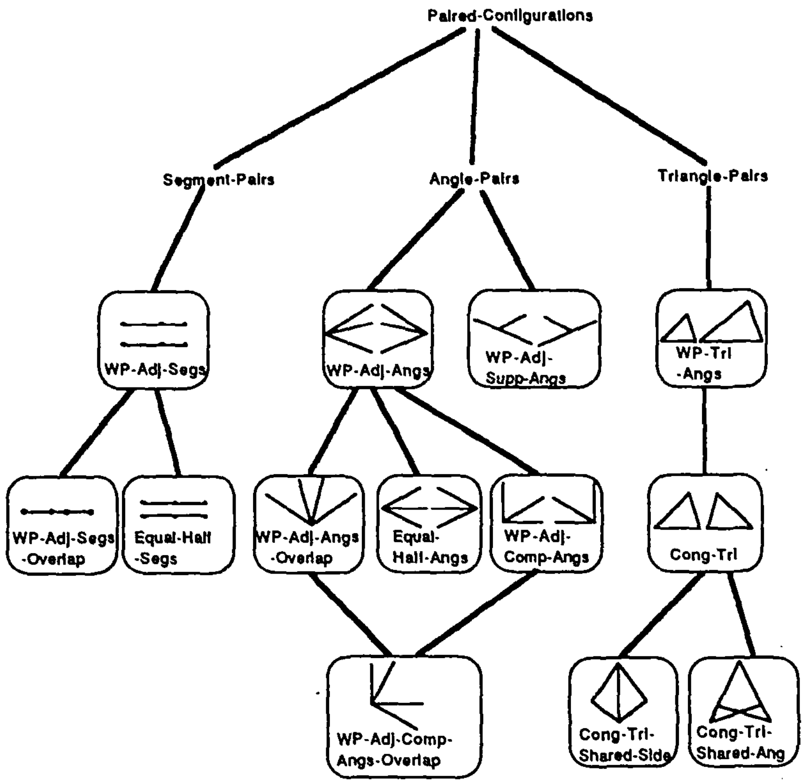


Figure 4b. The pairing of basic configurations where relationships hold among the corresponding parts of the configurations paired.

The final result of diagram parsing is a network of instantiated schemas and part-statements. Figure 5 illustrates this network for Problem 3. It is interesting to note that although no problem-solving search is done in this first stage, in effect, most of the problem-solving work is done here. The resulting network is finite and usually quite small. Searching it is fairly trivial.

3.2.2 Statement Encoding. After parsing the diagram in terms of diagram configurations, DC reads the problem given and goal statements. Statement encoding corresponds to problem solvers' comprehension of the meaning of given/goal statements. We claim that problem solvers comprehend given/goal statements in terms of part-statements. When a given/goal statement is already a part-statement, DC encodes it directly by appropriately tagging the part-statement as either "known" or "desired." However, there are two other possibilities.

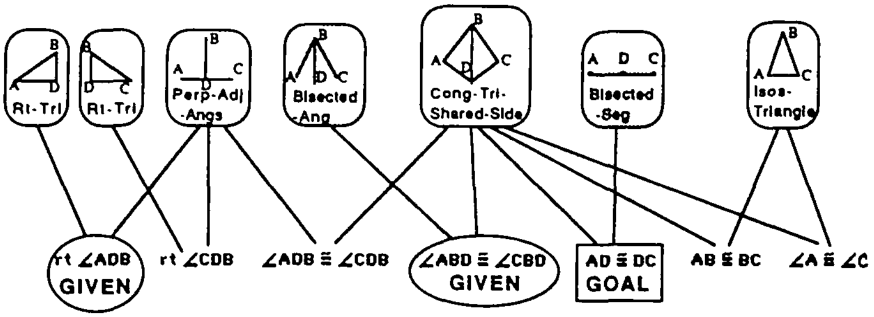


Figure 5. DC's solution space for Problem 3. The schemas DC recognizes during diagram parsing are shown in the boxes. The lines indicate the part-statements of these schemas. A solution is achieved by finding a path from the givens to the goal satisfying the constraints of the ways-to-prove slot of the schemas used.

First, if the given/goal statement is one of a number of alternative ways of expressing the same part-statement, it is encoded in terms of a single abstract or canonical form. For example, measure equality and congruence, as in **mAB = mBC** and **AB ≅ BC**, are encoded as the same part-statement. Using this abstract representation, DC avoids inferences, required in the execution space, that establish the logical equivalence of two alternative expressions of the same fact.

Second, if the given/goal statement is the whole-statement of a schema, it is encoded by appropriately tagging all of the part-statements of that schema as "known" in the case of a given or "desired" in the case of a goal. For example, the second given of Problem 3, **BD bisects ∠ABC**, is the whole-statement of a BISECTED-ANGLE schema. DC encodes it by establishing its only part-statement **∠ABD = ∠CBD** as known (see Figure 5). Similarly, DC encodes the goal statement of Problem 3 by tagging the part-statement **AD = CD** as desired.

3.2.3 Schema Search. Based on its parsing of the diagram, DC identifies a set of diagram configuration schemas which are possibly true of the problem. Its agenda then becomes to establish enough of these schemas as true so that the goal statement is established in the process. Typically, one of the ways-to-prove of a schema can be established directly from the encoded givens. So, for instance, in Problem 3 the PERPENDICULAR-ADJACENT-ANGLES schema can be concluded immediately. Other schemas require that additional statements be established about the diagram in order to conclude them. Thus, it is only after the PERPENDICULAR-ADJACENT-ANGLES schema is established in the example problem that the TRIANGLE-CONGRUENCE-SHAPED-SIDE schema can be established. At this level, DC is performing a search through the space defined by its diagram schemas much like the search GTE and other previous models perform through the execution space as defined by the rules of inference of geometry. We will refer to the space DC works in as the *diagram configuration space.*

As in the execution space, a search strategy and heuristics can be employed to guide search in the diagram configuration space. At any point DC has a number of schemas which it might apply. The system has a selection heuristic to chose among these schemas. Although a more powerful heuristic could be used, we have found that, because the diagram configuration space is so small, a simple heuristic is sufficient. In addition, this heuristic is consistent with our subjects who do not seem to spend much time evaluating alternatives, but rather forge ahead with the first reasonable inference that occurs to them.

Essentially, DC's selection heuristic implements a bidirectional depth-first search. A schema is *applicable* if there are proven part-statements which satisfy one of the schema's ways-to-prove. It is *desired* if its whole-statement or one its part-statements are goals of backward reasoning. If a schema is both applicable and desired, then DC selects it. Otherwise, DC either makes a forward inference by selecting any applicable schema or makes a backward inference by selecting any desired schema that is one statement away from satisfying one of its ways-to-prove.

The selection heuristic is made more efficient by only considering schemas which a quick estimate determines are potentially applicable. A schema is potentially applicable when the number of its part-statements, which are proven, is equal or greater than the size of the smallest way-to-prove. This estimate of applicability is much quicker to compute than checking all the ways-to-prove, and it eliminates from consideration schemas which are clearly not applicable at the current moment. It also leads to an interesting prediction. Since the heuristic only estimates whether a schema is applicable, it is possible that a schema will be selected even though it is not applicable (and not desired). For example, a TRIANGLE-CONGRUENCE-SHARED-SIDE schema may be selected when two of its part-statements are known even though these part-statements do not make up a way-to-prove (e.g., because they form the insufficient angle-side-side combination). More than once we observed subjects doing just this, considering whether two triangles are congruent because they had the right number of statements but failing because they did not have the right combination of statements. In Section 5.4.2, we relate this phenomenon to an "indefinite subgoaling" phenomenon identified by Greeno (1976).

3.3 Avoiding Algebra in the Diagram Configuration Space

One of the places where GTE gets bogged down while attempting difficult problems is in the fruitless application of algebra inferences. Algebra expressions can be combined and manipulated in infinite variety and as a result, algebra inferences often lead problem solvers into black holes in the search space from which they may never return (see the analysis in Section 4.1). Thus, it is worth discussing how DC avoids the black hole of algebra.

DC avoids the algebra subspace by having schemas which abstract away from algebra. In other words, these schemas are essentially macro-operators that make the same conclusions in one step that would require many steps to do by algebra.[3] These schemas are not ad hoc additions to remedy the difficulty with algebra subproofs. They correspond with particular geometric images and are formally no different than other diagram configuration schemas. They are instantiated as a result of diagram parsing and can be used when needed in place of difficult algebra subproofs. Essentially, these schemas provide a way to recognize when algebra is needed and when it is not needed. GTE does not have such a capability.

A single type of algebra schema handles most of the algebraic inferences. We call these schemas WHOLE-PART congruence schemas and they correspond with the configurations in Figure 4b that begin with WP. Our WHOLE-PART schemas are essentially the same as the WHOLE-PART schemas discussed in Anderson et al. (1981) and Greeno (1983).

A great variety of WHOLE-PART schemas can be formed by pairing any two component configurations which have corresonding parts (see Figure 4b). However, it would be misleading to suggest that all algebra subproofs can be solved using some WHOLE-PART schema. For example, the geometric proof of the Pythagorean theorem requires an algebra subproof involving multiplication and squaring which are outside the scope of WHOLE-PART schemas. Nevertheless, the vast majority of problems in a high school curriculum that require algebra subproofs fall within the scope of WHOLE-PART schemas.

4. EVALUATION OF THE DC MODEL

The purpose of this section is to discuss the strengths and limitations of the DC model. First, we describe a formal analysis of relative size of the execution space and the diagram configuration space to argue for the computational efficiency of DC. Second, we show how the DC model captures the

[3] While geometry textbooks have many theorems to skip commonly occurring steps, they do not have any theorems equialent to the algebra schemas we are proposing (at least none of the textbooks we've seen do). We can think of two possible reasons for why they are absent. First, the utility of such theorems has been overlooked by textbook writers. We doubt that this first reason is right. Second, since these theorems are dependent on information which is implicit in the diagram but is not explicit in formal statements, they are left out because it is difficult to express them in geometry formalism.

regularity in expert step skipping that is contrary to straightforward abstraction and macro-operator learning approaches. Third, we provide protocol evidence for a forward-reasoning preference displayed by experts on easier problems. Finally, we discuss some of the limitations of the DC model; in particular, we try to identify the task situations which stretch or break the model.

4.1 A Combinatorial Analysis

Comparing the problem-solving effectiveness of DC with other models of geometry theorem proving is complicated by the fact that there are multiple sources of intelligence in these models. In particular, the most important factors are (1) the problem space representation and (2) the search heuristics used. In addition to GTE, many previous models (e.g., Gelernter, 1963; Goldstein, 1973) search in the execution space. Variations in the problem-solving effectiveness of these models can be characterized by differences in search heuristics. Since DC uses a different problem space as well as different heuristics, the task of comparison is complicated. A more tractable task is to compare the problem space representations independent of heuristics. Since search performance could be improved in both spaces by adding heuristics, an analysis of the size of the two spaces should approximate the relative effectiveness of models based on these spaces.

4.1.1 Method of Analysis. The relative size of the execution and diagram configuration spaces was measured by comparing the "bushiness" of a brute force forward search in each space on Problem 7 in Figure 1. The bushiness is measured by counting the number of operators that apply at each successive "ply" of operators. The first ply is all the operators that can apply to the initial state (the givens). The second ply is all the operators that can apply to the collection of known statements created in the first ply. And so forth.

The operators we consider as part of the execution space are a collection of 27 definitions, postulates, and theorems that represent a significant share of the rules in a standard geometry curriculum up to and including rules for proving triangles congruent. To simplify this analysis somewhat, some rules concerning complementary and supplementary angles were left out. The operators of the diagram configuration space are diagram configuration schemas that correspond with the same slice of the curriculum (as shown in Figures 4a and 4b).

In addition to performing this analysis on the execution space and diagram configuration space, we also analyzed the size of the execution space when all the algebra and algebra-related operators are eliminated from it. The three algebra rules are the ADDITION-POSTULATE, SUBTRACTION-POSTULATE, SUBSTITUTION. In addition to these, any rules whose conclusions relate angle or segment measures need not be considered since these relationships

TABLE 2
The Size of Three Different Problem Spaces on Problem 7

	Execution Space	Execution Space without Algebra	Diagram Configuration Space
1st ply[a]	45	14	3
2nd ply	563	1	3
3rd ply	$>10^5$	3	2
4th ply	$>10^5$	1	
5th ply	$>10^5$	2	
6th ply	$>10^5$	6	
Total	$>10^6$	27	8

[a] A ply is all the operator instantiations that apply to the known statements produced by the previous ply.

can only be acted upon by algebra rules. This eliminates six more rules: DEF-MIDPOINT, DEF-BETWEENNESS, ANGLE-ADDITION, DEF-RIGHT-ANGLE, DEF-CONGRUENCE, and SUM-TRI-ANGS. We did the same analysis with this reduced rule set.

4.1.2 Results and Discussion. Table 2 indicates the results for the analysis, which can be summarized as follows. In the execution space, 6 plies of breadth-first search are required and more than 10^6 operator applications are investigated. In the execution space without algebra, 6 plies are required but only 27 operator applications are investigated. Interestingly, the size of the search space is dramatically decreased if algebra-related rules are not considered. Although this result is revealing, it does not suggest that we can just throw out algebra. Many problems require algebra subproofs in their solutions and thus, the execution space without algebra is not a workable alternative. However, the analysis indicates that algebra-related inferences can be a major source of combinatorial explosion.

Because of the larger grained operators of the diagram configuration space, only 3 plies of breadth-first search and 8 operator applications are required. This space is so much smaller than the execution space that a brute force search of this space can be effective, whereas domain-specific heuristics are necessary to search the execution space effectively. The diagram configuration space is also significantly smaller than the execution space without algebra, indicating its power is not derived solely by the algebra-avoiding WHOLE-PART schemas. In addition, whereas the execution space, without algebra, cannot solve problems like Problem 5 in Figure 1 where algebra is required, DC can solve the majority of these problems.

4.2 Accounting for Experts' Step-Skipping Behavior

In the process of planning a solution, our expert subjects made inferences that skipped more than 50% of the steps necessary for a complete solution in the execution space. In addition, we found that our subjects were skipping the same kinds of steps. In this section, we show how the diagram configuration space accounts for this regularity in step-skipping behavior.

4.2.1 Experimental Procedure. The data used for this analysis comes from 4 subjects' (B, K, J, and F) verbal reports on one problem and 1 subject's (R) verbal reports on eight problems. Two of the single-problem subjects (B and K) were mathematics graduate students while the other 2 (J and F) were researchers on the Geometry Tutor project. Subject R is a high school geometry teacher. All protocols were collected using the concurrent protocol methodology of Ericsson and Simon (1984) where subjects are asked to report what they are thinking as they solve problems. The 4 single-problem subjects were audiotaped as they entered their solutions using the interface of the Geometry Tutor, while subject R was videotaped as he made pencil markings on a paper diagram and reported his solution verbally. The record of computer interactions on one hand, and the video record of diagram marking and pointing on the other hand, helped to resolve ambiguous verbal references like "this segment is equal to this segment."

4.2.2 Method of Protocol Analysis. The protocols were segmented into

1. *Planning episodes* where subjects made inferences for the first time in the process of developing a proof sketch;
2. *Refinement episodes* where subjects refined their proof sketch by filling in skipped steps; and
3. *Execution episodes* where subjects indicated steps in their final solution.

The execution episodes of the single-problem subjects correspond with the verbalizations they made while entering steps into the Geometry Tutor interface. The execution episodes of Subject R, on the other hand, correspond with the verbalizations he made while reporting his final proof to the experimenter.

This particular analysis is focussed on the planning episodes. The goal of the data analysis was to identify the steps in a complete execution space solution that were mentioned by the subject during planning.[4] The execution space solution for each subject-problem pair was recorded in a proof tree diagram, and each statement that the subject mentioned during planning (except the given and goal statements) was circled on this diagram. Figure 2 illustrates the result of this analysis for the protocol of Subject R in Table 1.

[4] The complete execution space solution for the single-problem subjects is the one they entered into the Geometry Tutor interface. The multiple-problem subject, R, was not forced to indicate all the details of a complete execution space solution and thus, to decide what execution steps he skipped, we filled in the gaps with the shortest execution space path possible.

TABLE 3
Model-Data Fit for All Subjects Solving Problem 7

	Predicted Mention		Predicted Skip	
Subject	Actually Mention	Actually Skip	Actually Mention	Actually Skip
R	3	0	3	2
B	2	0	1	3
K	3	0	1	6
J	2	0	1	3
F	3	2	3	9
Total	13	2	9	23

TABLE 4
Model-Data Fit for Subject R Solving Eight Problems

	Predicted Mention		Predicted Skip	
Prob. No.	Actually Mention	Actually Skip	Actually Mention	Actually Skip
1	1	2	1	1
2	1	0	2	3
3	2	0	0	4
4	1	0	0	5
5	2	0	0	8
6	3	0	1	2
7	3	0	3	2
8	0	2	1	9
Total	13	4	8	34

4.2.3 Model Predictions. We derive predictions from DC by assuming that a statement will be mentioned for each schema application. If the schema has a whole-statement, we predict that this statement will tend to be mentioned. If it does not contain a whole-statement, for example, like the WHOLE-PART schemas, we predict the concluding part-statement will tend to be mentioned. We predict that all other statements will tend to be skipped. This prediction entails a quite simple assumption about the verbalization of problem states, that is, one verbalization per schema application, however, it provides a good fit to the data. Below we discuss how the major difference between the predictions and the data might be accounted for by a slightly more complex assumption about verbalization.

4.2.4 Results and Discussion. In the 12 subject–problem pairs, fewer than half of the intermediate steps were mentioned (37/98) and more were skipped (61/98). The model predicted that 29 steps would be mentioned and 69 skipped. Tables 3 and 4 show the data for each subject–problem pair and will be discussed below (note that Subject R, Problem 7 is in both tables).

Of the 29 steps that DC predicts will be mentioned, 23 were actually mentioned and only 6 were not. Of the 69 that DC predicts will be skipped, 55 were skipped and only 14 mentioned. A chi-square test was used to determine whether this distribution could have occurred by chance. The chi-square value, $\chi^2(1,N=98)=30.3$, indicates it is unlikely that the model's fit to the data is a chance occurrence ($p<.001$). We can take a closer look at the data to see how well the result generalizes across subjects and problems, particularly since the subjects are over-represented by Subject R and the problems by Problem 7.

Table 3 shows the data for all 5 subjects on Problem 7 and indicates the model to data fit is not peculiar to Subject R. A chi-square test on the column totals yields $\chi^2(1,N=47)=14.1$, $p<.001$. Table 4 shows the data for Subject R on eight problems and indicates that the results are not peculiar to Problem 7. A chi-square test on the column totals yields $\chi^2(2,N=59)=22.0$, $p<.001$.

If the model fit perfectly, the totals for Columns 2 and 3 in the tables would be zero. The predictions are most deviant from the data in Column 3: The subjects mentioned 14 steps[5] that were predicted to be skipped. Eleven of these cases are situations where the subject must use more than one part-statement in order to prove a schema. In such situations, subjects often mention one or more of these part-statements. For example, in planning a solution to Problem 3, part-statements $\angle \mathbf{ADB} = \angle \mathbf{CDB}$ and/or $\angle \mathbf{ABD} = \angle \mathbf{CBD}$ might be mentioned because both are needed to prove the TRIANGLE-CONGRUENCE-SHARED-SIDE schema. To account for such situations, our simple model of verbalization, namely, "one step mentioned per schema," could be elaborated to predict that extra verbalizations will tend to occur for schemas which require more than one part-statement to be proven. This more complicated model of verbalization would only provide a slightly better match to the data. While the number of misses (Column 3) would be reduced by 11, the number of false alarms (Column 2) would be increased by 6. The increase in false alarms results from the fact that subjects occasionally skipped part-statements the alternative model of verbalization predicts they should mention.

Other reasons why the predictions do not exactly fit the data include: (1) subjects may fail to mention an inference step for some model-unrelated reason, for example, because they momentarily forgot the experimental instruction to think aloud; (2) subjects, especially teachers, may feel inclined to explain themselves and thus, immediately report intermediate steps that support a leap of inference but were not a part of it; or (3) subjects may be

[5] Adding the third column totals from Tables 3 and 4 yields 17. However, since Subject R, problem 7 appears in both tables, we need to subtract 3 from 17 to get the proper overall total of 14.

at a different stage of expertise than DC by either (a) being behind, having not yet acquired certain configuration schemas, or (b) being ahead, having acquired larger configurations than the ones DC uses. A potential instance of (3b) may explain the two steps in Subject R's solution to Problem 8 (see Figure 1) that he skipped although we predicted he would mention them (see Column 2 of Table 4). In this case, it appeared that the subject used a diagram configuration that combined two of DC's schemas and thus was able to skip extra steps that the current version of DC cannot.

4.3 Forward Inferencing and Completion by Exhaustion

Of the eight problems Subject R solved, he solved five by a purely forward search (Problems 1, 3, 4, 5, & 8 in Figure 1), one by a forward search guided by the goal (Problem 2), and two using some backward inferences (Problems 6 & 7). By pure forward search, we mean that the problem solver did all of his reasoning without using, and often without reading, the goal statement. The five purely forward solutions were on problems that tended to be easier for him in the sense that he solved them in less time. Only one of these five took longer than any of the other three.

One somewhat peculiar and interesting aspect of subject R's forward reasoning was that on a number of the simpler problems he was able to decide he had finished the proof before reading the goal. For instance, while solving Problem 5 he said, "I didn't even look at the goal but I've got it." At some point in solving these problems he knows everything he can about it. As he says while solving Problem 3, "we can determine anything from there" (see Table 1). It is as if he exhaustively searches all possible forward inferences. But, an exhaustive search of the execution space for a particular problem is unlikely given its typical vast size, particularly since algebra inferences could chain on infinitely. On the other hand, the size of the diagram configuration space for these problems is quite small. In fact, it is bounded by the number of plausible diagram configurations which appear in the problem diagram. Thus, it seems that Subject R is able to stop his forward inferencing and conclude he is done when he has proven (or considered) all the plausible configurations.

Larkin, McDermott, Simon, & Simon (1980a, p. 1338) describe physics experts as working forward on simpler problems where they are relatively sure that "solving all possible equations will lead quickly to a *full understanding* of the situation, including the particular quantity they are asked for." This description provides a good characterization of Subject R if we simply replace "solving all possible equations" by "applying all possible configuration schemas." One difference, though, is that physics equations typically correspond with one step in the solution of a physics problem, while diagram configurations correspond with multiple steps in a geometry proof. This is particularly important since the execution space of geometry is so large. Without the chunking provided by diagram configurations, it seems unlikely that a working-forward strategy could work on all but the simplest geometry proof problems. Subject R's ability to work purely forward on relatively difficult problems, as well as his ability to recognize that he is done before reading the problem goal, are further evidence for the DC model.

4.4 DC's Limitations

We discuss DC's limitations both in terms of how the computer simulation could be extended to be a more complete and accurate model of geometry expertise and in terms of what situations cause trouble for DC's particular problem-solving approach. The computer simulation could be made more completely by adding procedures (1) to refine and execute the abstract plans DC currently creates, (2) to determine when and where constructions are necessary, (3) to integrate diagram parsing and schema search, and (4) to draw diagrams from general geometric statements.

4.4.1 Plan Execution. A model of plan execution would involve finding solutions, either by retrieval or by search in the execution space, to the series of short subproblems that result from planning. The majority of these subproblems are only one or two execution steps long. The longer subproblems are algebra proofs of the steps skipped by the WHOLE-PART schemas. These proofs share the same general structure, and experts do them by retrieval for the most part. Even if the solutions to these subproblems are done from scratch, they are small enough that they can be easily solved by search in the execution space. Adding procedures for doing search in the execution space would have the additional advantage of providing a way to perform certain types of algebra inferences that do not correspond with any of DC's current diagram configurations. These inferences often involve the pairing of two different types of configurations. For example, the RIGHT-TRIANGLE and the ADJACENT-COMPLEMENTARY-ANGLES configurations (see Figure 4a) can be paired to form an equation between the two nonright angles of the right triangle and the two adjacent complementary angles. We could supplement DC with such kinds of paired configurations (as in Figure 4b) or, alternatively, the execution space search component could be used to discover such pairings.

4.4.2 Constructions. The computer simulation could also be made more complete by adding procedures to perform "constructions," that is, the drawing of auxiliary lines in a problem diagram to provide new inference possibilities. Currently DC is not capable of performing constructions and thus, cannot solve the class of geometry problems which requires them. However, we believe that DC is particularly well-suited for adding a construction capability. The major decision points in solving proof problems that may require constructions are (1) deciding when a construction might be needed, and (2) deciding what construction to introduce. Typically, geometry systems attempt to perform constructions only when other methods appear to be failing. Since the diagram configuration space for any particular problem is relatively small compared to the execution space, DC could quickly and efinitively determine when a construction is necessary by exhaustively searching this space. The task of proposing potentially useful constructions could be performed in DC by completing configurations that partially match images in the diagram.

4.4.3 Integrating Diagram Parsing and Schema Search. The computer simulation could be made more efficient and more accurate as a model of human problem solving by integrating the diagram parsing and schema search processes that are currently performed in separate stages. Instead of doing all of the diagram parsing ahead of time, it should only be done on demand when the system is focussed on a part of the diagram which has not been parsed. Initially, the encoding of the problem given and/or goal statements could provide a focus of attention on a particular part of the diagram that involves these statements. DC could parse this portion of the diagram in terms of the configurations that appear there. Later, any new part-statements proven via schema search could shift the focus of attention to other parts of the diagram which could be similarly parsed. What remains to be defined is the range of attention, that is, how much of the diagram should be parsed at one time.

Integrating the parsing and schema search would make DC more efficient in cases where the diagram contains overspecialized figures, that is, configurations that look true, but do not follow from the problem givens. In such cases, the current diagram-parsing process instantiates configuration schemas that will never be used in problem solving. For example, the line **GH** in Problem 7 turns out to be irrelevant to the solution: There is no given information that bears on it. However, since it appears parallel to line **AB**, the diagram parser instantiates numerous schemas that correspond with apparent relationships like **△GCK≅△HCK, ISOS△CGH, AB∥GH**, and **GH⊥CD**. Without line **GH** the diagram contains 15 schema instances, with **GH** it contains 28 more. In the process of schema search these schemas are never used, so the work of instantiating them is wasted. If diagram parsing were done on demand, however, this extra work would not be necessary.

4.4.4 Diagram Drawing. While overspecialized problem diagrams can cause a slight amount of extra work, they do not cause DC to fail on problems. However, if the diagram is improperly drawn, that is, if it does not correctly represent the problem givens, the current simulation will not be able to solve the problem. For example, if the line **BD** in the diagram for

Problem 3 did not appear perpendicular to the base, DC would not instantiate the PERPENDICULAR-ADJACENT-ANGLES schema and thus, could not solve the problem. One way to extend DC to deal with such diagrams is to allow it to consider configurations beyond those which are apparent in the diagram, like PERPENDICULAR-ADJACENT-ANGLES in the example above. An alternative involves following the standard classroom wisdom which suggests that such diagrams should be redrawn. In particular, we could extend DC to deal with inaccurate diagrams by adding a diagram-drawing facility that could draw diagrams to reflect a problem's givens accurately.

5. COMPARISON WITH PREVIOUS GEOMETRY EXPERT SYSTEMS

Geometry theorem-proving models have been developed by numerous researchers, most primarily with AI concerns (Gelernter, 1963; Goldstein, 1973; Nevins, 1975) and at least one, besides GTE, based on human data (Greeno, 1978). We make comparisons with Gelernter's model because it was the first, Nevins's model because it is the most powerful system we are aware of, and GTE and Greeno's model because they were based on human data.

5.1 Gelernter's Geometry Theorem-Proving Machine

Gelernter's (1963) model was the first AI model of geometry proof problem solving and it worked by performing a backward heuristic search in the execution space. The use of the execution space puts the model at a disadvantage that could only be overcome if the heuristics in Gelernter's model make up for the power gained by the abstract nature of the diagram configuration space. However, this is not the case. The major heuristic of Gelernter's model was to reject backward paths when they became implausible in the diagram. Since only plausible configurations are considered by DC, these backward paths that Gelernter's model rejects are not even in the diagram configuration space for a particular problem. Thus, they are rejected implicitly without ever being considered.

Gelernter made no claims about modeling the inference-by-inference behavior of human problem solvers. And even at a more descriptive level, his model's emphasis on backward reasoning is inconsistent with the opposite forward reasoning emphasis of human geometry experts. In addition to Subject R's clear forward reasoning preference, a much larger proportion of the other subjects' inferences were forward rather than backward.

5.2 Nevins's Model

Nevins (1975) presents a geometry theorem-proving program that is probably more effective and efficient than any other geometry model. His major emphasis was on structuring the problem space of geometry such that a predominantly forward reasoning strategy could be effective. Nevins claimed that human experts engage in much more forward inferencing than backward inferencing. Although Nevins provided no evidence and was probably reacting to the purely backward reasoning strategy of most expert systems at that time, it is interesting that he made this claim well before empirical evidence came out verifying his intuition in physics problem solving (Larkin et al., 1980a), medical diagnosis (Patel & Groen, 1986), and now in geometry. The success of forward inferencing in Nevins's model is made possible by the way in which he structured the problem space. Unfortunately, Nevins is not very clear about the exact structure of this problem space. The structure is embedded in the processes he describes.

However, the problem space implicit in Nevins's description is much more like the diagram configuration space than the execution space. Because the model only recognizes six predicates (LN = line, PR = parallel, PRP = possibly parallel, RT = right angle, ES = equal segment, and EA = equal angle), it is effectively working in an abstract problem space. It ignores the distinction between congruence and measure equality, as well as the distinction between midpoint and bisector predicates and their corresponding equality predicates. The model makes inferences using a number of "paradigms," which are cued by certain features of the diagram, and which make conclusions in the form of the predicates. These paradigms share many characteristics with diagram configuration schemas: (1) They are cued by the diagram; (2) They can make multiple conclusions; and (3) They are often macro-operators, that is, capable of inferences which require multiple steps in the execution space. However, they are embedded in complex procedures within Nevins's model and are not clearly and uniformly represented like diagram configuration schemas are. Nevins's model does not use appearances in the diagram as DC does to create candidate schemas.

Although he did not present it this way, the success of Nevins's model can be considered further evidence for the computational efficacy of abstract planning in geometry. What the DC model adds is an explicit and uniform representation which (1) makes clear why Nevins's model worked, and (2) makes clear how it could be extended, say, by adding diagram configurations for circles. An important side effect of DC's explicit and uniform representation is that it is teachable (see Section 6.4). In addition to its computational advantages, we have provided empirical evidence that human experts solve problems like DC.

5.3 The Geometry Tutor Expert System (GTE)

The GTE, as described in Section 1, was designed as a model of ideal student problem solving to use as a component of an intelligent tutoring system (ITS). The system works in the execution space and uses a best-first bidirectional search strategy. To be successful in the otherwise intractable execution space, GTE uses heuristics to guide its search. These heuristics were designed to be psychologically realistic and consistent with the ACT* theory of cognition (Anderson, 1983). The general idea behind heuristics in ACT* is that student problem solvers learn various contextual features that predict the relevance of an inference. These contextual features are incorporated in the left-hand sides of the production rules and, in GTE, are either features of the problem diagram, previously established statements, or goals. As an example, consider the diagram of Problem 7 in Figure 1. Although one can immediately infer GK = GK and CD = CD by the reflexive rule, only the latter is a sensible inference that good students make. According to GTE, this is because good students have learned that one situation where the reflexive rule is useful is when the segment is a shared side between two triangles that might be congruent. Thus, GTE has a rule of the form:

IF there are plausibly congruent triangles ACD and BCD,
THEN conclude CD = CD using the reflexive rule.

GTE has a large set of such rules, some of which reason forward from the givens of a problem and others which reason backward from the goal. Each rule has an aptness rating which reflects how likely it is to be useful. For instance, a variant of the rule above, which tests whether there is a goal to prove the two triangles actually congruent, has a higher aptness rating than the rule above which, in turn, has a higher aptness rating than a rule simply suggesting that any segment is congruent to itself. These aptness ratings correspond with production strengths in the ACT* theory.

GTE provides a reasonably good model of student problem solving and has the advantage of being embedded in a unified theory, that is, ACT*, which provides an account of many other cognitive tasks. However, from a computational point of view, the model has the disadvantage that it often gets bogged down in fruitless search while attempting difficult problems, especially ones where algebraic inferences are required. In addition, there is no systematic way to assign aptness ratings to rules, so extending the model becomes increasingly difficult. From an empirical point of view, GTE's problem-solving approach does not correspond with the abstract-planning approach we observed experts using.

5.4 Greeno's Perdix

Greeno (1978) used verbal report data from geometry students as the basis for the design of a geometry theorem-proving model called Perdix. Like GTE, it is more accurately characterized as a model of geometry students rather than geometry experts. Unlike Nevins, Greeno's goal was not so much to build a powerful problem-solving model, but rather to capture the problem-solving behavior of geometry students. In relation, our goal in building DC was to capture the problem-solving behavior of geometry experts in order to have a model which is not only a powerful problem solver, but also solves problems in a way that can be profitably taught to students.

Perdix used a mixture of execution space operators and more abstract macro-operatorlike operators. With respect to algebraic reasoning, Perdix contained operators that are essentially the same as DC's whole-part schemas (Greeno, 1983) and thus, could skip over the details of algebraic proofs. However, with respect to geometric reasoning, Perdix operators appear to have been procedural encodings of geometry rules, that is, execution space operators. In the empirical research associated with Perdix, Greeno made a couple of observations which are particularly notable in relation to DC. The first concerns the use of perceptual processing in geometric reasoning, and the second concerns a useful type of nondeductive or "indefinite" reasoning that both students and experts appear to engage in.

5.4.1 A Physical Distance-Reducing Heuristic. The first observation is the way in which good students appear to use a visually based heuristic to guide their selection of appropriate inferences in a certain class of "angle-chaining" problems (Greeno, 1978). These problems are common in the parallel-line lessons of geometry curricula, and typically involve sets of parallel lines, for example, two sets of two parallel lines forming a parallelogram on the inside. Students are either (1) given the parallel-line relationship(s) and the measure of some angle and asked to find the measure of another angle, or (2) given only the parallel lines and asked to find a relationship between two angles. In either case, the problem usually involves finding some other angle which connects the two angles in question via the transitivity rule. Although these problems typically contain numerous angles to choose from, Greeno observed that students are fairly regular (and accurate) in their selection of this "chaining angle." They tend to pick an angle which, in the diagram, is physically between (or close to) the two angles to be connected.

Perdix models this behavior by forming a "scanning line" between the known and desired angles in the diagram, and candidate chaining angles are considered in order of their proximity to this scanning line. This scanning line method is an instance of a more general method for proposing subgoals by identifying objects that are physically between the known and desired objects. The method is based on a heuristic: An operation which reduces the physical distance between known and desired objects may also reduce the logical distance between them. Although DC has not been programmed with such a distance-reducing heuristic, such a heuristic might aid DC on harder problems in identifying diagram configurations that are most likely to provide a link between known and desired configurations. The protocol data provides no evidence that experts use this heuristic; however, the problems subjects solved were not particularly demanding of such a heuristic.

5.4.2 Indefinite Goals. A second notable behavior that Greeno (1976) observed of geometry students is that they often engage in the setting of what he called "indefinite goals." When given a problem, like Problem 5, with a goal to prove two triangles congruent, instead of attempting to prove particular corresponding parts congruent that are a part of a particular triangle congruence rule, for example, side-angle-side, subjects attempt to prove any of the corresponding part-statements they can. These statements are indefinite goals because they are not associated with any definite rule. DC accounts for indefinite goals because they are a natural consequence of the way in which it applies schemas in backward inferences. In DC, a schema is applied in a backward inference by making all part-statements desired. In cases where the ways-to-prove of the schema require multiple statements, the desired part-statements are indefinite goals since they were not set in order to achieve any particular subset.

A related type of reasoning is characteristic of certain types of *forward* inferencing in DC. In particular, the selection heuristic may chose to apply a TRIANGLE-CONGRUENCE-SHARED-SIDE schema in the forward direction because a sufficient number (2) of the schema's part-statements are known. This selection is indefinite in the sense that these two part-statements may not be the right ones to match any of the ways-to-prove. Geometry experts also appear to make such indefinite selections. At some point during Problem 7, Subjects R, B, K, and F all considered proving $\triangle$**ACD**$\cong\triangle$**BCD** and/or $\triangle$**AKD**$\cong\triangle$**BKD** because they had established the congruence of three corresponding parts, but found that they could not since these parts formed the insufficient angle-side-side combination.

It should be noted that both Nevins's model and Perdix (Greeno, Magone, & Chaiklin, 1979) are capable of introducing constructions into the geometry diagram allowing them to solve a class of problems that DC cannot because it currently does not have a construction capability. However, as noted in Section 4.4, we believe that DC is particularly well suited for adding a construction capability.

6. DISCUSSION AND IMPLICATIONS

Previous models of geometry problem solving do not provide an explanation of the abstract-planning abilities of experts. Geometry experts can quickly and accurately develop an abstract proof plan that skips many of the steps required in a complete proof. We built a computer simulation of geometry expertise, DC, which models this abstract planning behavior. DC's planning is based on perceptual chunks called diagram configurations, which provide a reliable index to clusters of relevant geometry facts. To establish the computational advantages of DC, we performed a problem space analysis, which showed that DC is more efficient than models based on the execution space of geometry. In addition, we showed that DC's particular approach to abstract planning is much like that of human experts. Making a conservatively simple assumption about how DC would verbalize its inferences, we found that the model does a good job of accounting for what steps experts mention (and skip), while developing an abstract proof plan.

We now turn to a discussion of how these findings relate to or might inform other issues in cognitive science. In particular, we discuss:

1. How these findings bear on the controversy in the human-reasoning literature (see Holland et al., 1986) between specific instances, mental models, schemas, and natural logic rules as the representational basis for human reasoning.
2. How these findings contribute to the study of expertise in general.
3. How these findings fit (and don't fit) within unified theories of cognition like ACT* and Soar.
4. How these findings might be applied to improve geometry instruction.

6.1 The Raw Material of Reasoning: Instances, Models, Schemas, or Rules

Holland et al. (1986) discuss four alternative theoretical views on human reasoning that have grown primarily out of the empirical research on syllogism problems and Wason's (1966) selection task. These views present different hypotheses about the nature of the basic material with which we reason. They are listed in order below, from a view of reasoning knowledge as extremely specific to a view of knowledge as extremely general.

- *Specific instances:* Reasoning proceeds by recalling specific instances of past reasoning events which indicate an appropriate conclusion (see Griggs & Cox, 1982).
- *Mental models:* Reasoning is performed by domain-independent comprehension procedures that construct a concrete model of the problem situation from which conclusions can be read off (Johnson-Laird, 1983; Polk & Newell, 1988).
- *Pragmatic reasoning schemas:* Reasoning is performed by the application of pragmatic reasoning schemas that are abstractions of past reasoning events (Cheng & Holyoak, 1985).
- *Natural logic rules:* Reasoning proceeds by the application and chaining together of abstract rules, much like the formal rules of logic, to deduce a conclusion (see Braine, 1978; Rips, 1983)

While the knowledge elements of the specific instance and mental model views are more concrete and declarative in nature, the knowledge elements of the pragmatic reasoning schema and natural logic rule views are more abstract and procedural. In the first two views, the knowledge elements are descriptions of concrete objects and situations in the world which must be interpreted in order to derive actions or conclusions. In the latter two views, the knowledge elements do not correspond to any particular situation or set of objects, but to large categories of situations, and they prescribe an action to be performed or conclusion to be made in that general situation.

The question we wish to pursue is how the growing understanding of reasoning in geometry fits within the spectrum of these four alternative views of human reasoning. Geometry reasoning, as characterized by DC, is least like the natural logic rule view. DC's schemas are specific to geometry and thus, are quite unlike the general natural logic rules. On the other hand, DC's schemas are not specific enough to equate them with the specific instance view. In general, neither students nor experts solve geometry problems simply by recalling past experiences of solving them.

We are left with the two intermediate views. Because the distinction between them is somewhat subtle, we describe them in more detail. The mental model approach is of intermediate generality in that it uses general language abilities to construct a model (referent) of the problem statement, but the effectiveness of this model is limited by the reasoner's specific knowledge of the language of the domain. The pragmatic reasoning schema view is intermediate in that reasoning is based on knowledge elements (schemas), which are general enough to apply to numerous problem types and domains, but are not as general as formal logic rules that are applicable in any domain. One implication of the difference between these approaches is that the mental model approach explains reasoning errors in terms of working-memory failures, while the schema approach explains them in terms of negative transfer, that is, the mapping of a schema to a situation where the schema-based inference is incorrect (Holland et al., 1986).

DC has similarities to both the mental model and pragmatic reasoning schema view. It is similar to the mental model approach in that it uses the problem diagram as a specific referent or model of the abstract problem statement indicated by the givens and goals. Many features of this model are usually too specific to be relevant, for example, the particular lengths of segments. However, other specifics of the model can be important as they can provide a cue to relevant inferences, for example, congruent-looking triangles can cue an inference to prove them congruent. A concrete model has the advantage of making important features or relationships clearly apparent (visible in this case) whereas they are only implicit in abstract statements. In addition, the cues from the model have the effect of allowing the problem solver to ignore much potentially applicable, but irrelevant logical knowledge. A model-building procedure like the one Johnson-Laird proposes is not necessary since the diagram provides a ready-made model.[6]

[6] While many geometry proof problems given in classrooms include a diagram, it is not uncommon to state proof problems without a diagram, for example, the problem in Figure 2 could be stated as "prove that if the perpendicular attitude of a triangle bisects the angle, it also bisects the base." Such problems are typically solved by drawing an appropriate diagram, a concrete model of this abstract statement, and then proceeding as usual. In this case, the problem solver is constructing a mental model.

According to the mental model approach, what is left for the problem solver to do is to annotate the model properly and read off the conclusion. This is essentially what we propose experts do: They annotate the diagram, on paper or in the mind's eye, by noting established relationships.

However, the annotation process is not as straightforward as it is in other problems the mental model approach has been applied to. Rather, it involves fairly complicated logical inferences, including, for example, the checking of ways-to-prove. This inferencing requires the abstract geometric knowledge which is part of the DC schemas. This knowledge is more like pragmatic reasoning schemas in that it is applied procedurally, and appears to be acquired as abstractions of past geometry problem-solving experiences.

Although the four views can be posed as competing hypotheses, it is likely that human reasoning, in general, contains elements of each. While the DC model lends support for the use in geometry of a combination of the mental model and pragmatic reasoning schema approaches, neither approach by itself is sufficient.

6.2 Contributions to the Study of Human Expertise

6.2.1 What's Behind Expert's Forward-Reasoning Ability? One claim that has been made about human experts is that they show a greater tendency than novices (especially on easier problems) to work forward from the givens of a problem rather than backward from the goal. This result has been observed in physics word problems (Larkin et al., 1980a), in classical genetics word problems (Smith & Good, 1984), and in medical reasoning (Patel & Groen, 1986) by comparing the problem-solving behavior of experts and novices. Although the comparisons were made between different subjects, the invited inference is that, as a person acquires skill in one of these domains, his or her problem-solving strategy will tend to shift from working backward to working forward. To observe this shift within the course of skill acquisition, Sweller, Mawer, and Ward (1983) developed a toy domain, using three equations from kinematics, where subjects could become "experts" in a relatively short period of extensive practice (77 problems). They found the expected shift as subjects worked forward on significantly more of the final problems than they did on the initial problems.

In geometry, we observed an expert (Subject R) working exclusively forward on a number of the simpler problems we asked him to solve. This ability to solve certain problems, essentially without looking at the goal, is an ability geometry novices do not have. We would like to address the issue of how Subject R, and experts in general, are able to work forward successfully.

It should be pointed out, first, that this shift to working forward is not characteristic of all domains of expertise. In some domains the given information is inadequate to solve problems successfully by forward search. Jeffries, Turner, Polson, and Atwood (1981) showed that expert programmers do not work forward from the problem givens (i.e., the programming-language primitives), rather they work backward from the goal information (i.e., the program specifications). The shift to working forward appears to be characteristic of deductive domains, like equation chaining or proof domains, where the given information is quite rich, and uncharacteristic of design domains, like programming, where the given information is poor.

In domains where working forward can be performed successfully, it should not be a surprise that learners adapt toward using it more often. By working forward, problem solvers can write down inferences as they make them and relieve the memory burden of storing previous solution steps. Backward or bidirectional search, on the other hand, demands that the problem solver encode and integrate more information, as well as remember intermediate goals. Sweller (1988) makes similar arguments and presents a computational model and experimental evidence to support them. The upshot is that if a learner can develop the ability to work forward successfully, he or she can alleviate some of the extra working-memory burden required by a backward strategy.

Sweller (1988) also proposes an explanation for expert's ability to work forward successfully. He suggests that experts use *schemas* to classify problems into categories that carry implications for appropriate moves to make. Sweller (1988, p. 259) defines a schema as "a structure which allows problem solvers to recognize a problem state as belonging to a particular category of problem states that normally require particular moves." The diagram configuration schemas of the DC-model fit Sweller's definition. They allow the categorization of subproblems based on recognizing prototypical images in the problem diagram and the retrieval of the relevant subproof.

The key point is not so much that experts will necessarily prefer working forward. Rather, it is that, as a result of their superior skill, experts are *capable* of working forward successfully without recourse to backward reasoning. Knowledge in the form of schemas permits them to do so. However, schemas alone are not enough. The schemas must be large enough, or the problem small enough, so that they reduce the search space sufficiently for forward reasoning to be effective. We have seen how DC's schemas make the search space of even relatively difficult problems quite small, for example, the forward search space of Problem 7 is only eight schemas (see Table 2). Still, all of our experts did some backward reasoning on Problem 7. It was only on simpler problems, like 3 and 5 with only three relevant schemas, that Subject R performed a purely forward search.

6.2.2 Perceptual Chunks and Problem-Solving Performance. One of the more robust results regarding expert–novice differences is the enhanced memory of experts for problem-state displays. This difference has been established in a variety of domains: chess (De Groot, 1966), electronic circuits (Egan & Schwartz, 1979), baseball (Voss, Vesonder, & Spilich, 1980), computer programming (Jeffries et al., 1981), and algebra (Sweller & Cooper, 1985). In the earliest study of this type, it was shown that chess masters can remember realistic board positions much better than chess novices can (De Groot, 1966). This result does not arise from any innate perceptual or memorial advantages experts might have, rather it arises from their extensive chess experience. Experts are not better than novices at remembering boards with randomly placed pieces.

While these recall abilities are correlated with game-playing skills, it has yet to be established decisively whether they are necessary parts of game-playing skills or whether they are merely side effets of spending a great deal of time staring at a chess board. The theory behind the recall results is that subjects perceive the board in terms of prototypical configurations of pieces, "chunks," and that experts' chunks are made up of more pieces than those of novices (Chase & SImon, 1973). Chase and Simon have suggested that experts associate appropriate chess moves with these chunks, and Simon and Gilmartin (1973) have a model of chess perception. However, a model has yet to be written capable of both performing the recall task and playing chess. At the same time, the proposal that experts associate moves with these chunks has received criticism (Holding, 1986).

The DC model is a step toward establishing a detailed theoretical connection between perceptual chunks and problem-solving performance. The diagram configurations of DC provide a ready-made theory of perceptual chunks in geometry. We have already seen that these perceptual chunks provide the basis for expert problem-solving performance. It would not be difficult to model superior problem-state recall in geometry by chunking problem diagrams in terms of diagram configurations. Thus, it appears that the appropriate knowledge representation is in place in DC to model both problem-state recall and problem-solving skill in geometry. Implementing a recall component and replicating De Groot's empirical result in geometry are tasks for future research.

Turning back to chess, DC's use of diagram configurations for abstract planning might be the appropriate analogy for an integrated chess model. Rather than cueing particular moves, chunks in chess may be more effectively thought of as problem-state abstractions that provide the basis for an abstract problem space in which players can plan and evaluate multiple-move strategies.

6.3 DC's Relation to Comprehensive Theories of Cognition

In 1972, Newell (1973) gave his well-known "20 questions" talk in which he argued that, to avoid spinning wheels in cognitive science research, it is necessary to begin integrating local hypotheses and doman models into global theories that account for cognition across a wide variety of tasks. Creating such comprehensive theories has now become a major research effort (Anderson, 1983; Holland et al., 1986; Johnson-Laird, 1983; Newell, in press). In this section we try to place DC in terms of two of these theories, ACT*

(Anderson, 1983) and Soar (Newell, in press). We address the issue of whether the mechanisms of problem solving and learning in these theories can account for expert geometry problem solving as modeled by DC.

Because both ACT* and Soar use a production rule representation of knowledge, our first challenge is to find a way to express DC's schemas as production rules in such a way as not to change the resulting behavioral predictions. Consider the TRIANGLE-CONGRUENCE-SHARED-SIDE schema in Figure 3. This schema can be represented as six production rules whose left-hand sides correspond to the six ways-to-prove of the schema, and whose right-hand sides contain five actions corresponding with the five part-statements of the schema. A similar translation could be made to express backward schema application in terms of productions. Note that these production rules are *macro-operators* with respect to the execution space of geometry, in that they have the effect of numerous execution space operators.

Is anything lost in translating schemas into productions? In terms of problem-solving behavior the answer is probably no. However, another question we need to ask with respect to ACT* and Soar is whether the particular productions that correspond with DC's schemas could result from the learning mechanisms of these theories. This question is more problematic. The clusters of productions corresponding with DC's schemas organize the formal rules of geometry in a particular and efficient way. It is not clear how the production rule learning mechanisms in either ACT* or Soar could arrive at such an organized set of productions.

These theories essentially view skill acquisition as involving two phases: knowledge acquisition and knowledge tuning. In the knowledge acquisition phase, the learning system uses information about the problem domain, for example, problem descriptions, problem constraints, example solutions, and so forth, to build some kind of basic problem space. In geometry, this would involve acquiring the formal rules of geometry, that is, the execution space operators, through instruction and examples. In the knowledge tuning phase, the basic problem space is elaborated through problem-solving practice so that the system becomes more effective and efficient. Much of the research on skill acquisition in ACT* and Soar has focussed on this second knowledge-tuning phase. The basic approach of these theories to knowledge tuning is a process of reducing the number of productions required to perform a procedure. Essentially, both use a type of macro-operator creation mechanism in which consecutively applicable productions or operators are composed into a single production or macro-operator.[7]

[7] To cut off potential confusion based on the distinction in Soar between operators and productions, we would like to make clear that when we use "macro-operator" in reference to Soar, we are not referring to the combination of Soar operators into macro-operators: Soar has no direct mechanism for doing this. Rather, we are talking about the chunking of Soar *productions* into bigger productions.

There are both empirical and computational reasons to doubt the DC derives from creating macro-operators of the execution space operators. First, the step-skipping regularity we observed is an unlikely consequence of this approach. Although ACT* and Soar have some stipulations on the appropriate context in which macro-operators are formed, there is little in them indicating which sequences of consecutively applicable productions are more likely to be composed than others. Thus, we would not expect any regularity in the kinds of steps that would be skipped in an abstract problem space of composed execution operators. However, such a regularity is exactly what we observed of subjects.

To be more precise, both theories stipulate that macro-operator formation occurs within a goal structure, that is, macro-operators are formed of consecutive productions applied to achieve the same goal. Thus, the clustering of productions into macro-operators will reflect the organization of a problem solver's goals and subgoals, and to the extent that this goal structure is consistent across many problems, a step-skipping regularity could emerge. However, it appears more likely that macro-operatorlike knowledge in geometry is not primarily organized around goals but is organized around objects and aggregations of objects in the domain. According to this view, DC's schemas are not really macro-operators in the sense of being derived from execution operators. Rather, they derive from perceptual chunking of domain objects and they merely bear a macro-operator relation with execution space operators.

A second reason to question the macro-operator learning approach comes from evidence in the verbal reports that subjects could not always immediately fill in the steps they had skipped during planning in the process of executing an abstract plan. For example, in Problem 7 subjects would plan to prove the goal from $\angle\mathbf{ADC}=\angle\mathbf{BDC}$, apparently using the PERPENDICULAR-ADJACENT-ANGLES schema. During plan execution, some subjects did not immediately know how to justify the link between these two statements: They attempted an algebra proof or searched the list of available geometry rules we provided. However, if they had learned this schema by composing execution space operators, that is, the very operators they needed at this point, we would expect that these operators would be readily available. Since these execution operators remain necessary to execute proof plans, there is no reason why they would be forgotten in the course of skill acquisition. It appears that experts' knowledge of the macro-operatorlike schemas is occasionally stronger than their knowledge of the corresponding execution operators. This evidence is inconsistent with a view of the schemas deriving from the execution operators, provided, as is the case here, that the execution operators are still necessary to solve problems.

Finally, there are computational reasons to question macro-operator explanation of step skipping. Recall the macro-operator characterization of the TRIANGLE-CONGRUENCE-SHARED-SIDE schema given above. The collection of such macro-operators for each schema, call it *S*, is a restricted subset of the space of possible macro-operators. *S* is restricted in two ways. First, *S* does not contain any of the possible macro-operators that could make inferences between statements, which are whole-statements of schemas; for example, it doesn't contain an oeprator that could infer perpendicularity directly from triangle congruence in a problem like Problem 3. Second, *S* does not contain any of the 2, 3, or 4 action macro-operators that would be learned on the way to a 5-action macro-operator like the one corresponding with the TRIANGLE-CONGRUENCE-SHARED-SIDE schema. To achieve DC's simplicity in search control and match to the human data, a composition mechanism would need to prevent a proliferation of unnecessary macro-operators. It is not clear how this restriction could be implemented in ACT* or Soar.

One might consider whether this restriction could be achieved within the Soar architecture by having a hierarchy of problem spaces corresponding with the desired organization. However, this approach begs the question: How would this hierarchy be learned in the first place?

6.4 Implications for Geometry Instruction

One of the goals of this research is to develop a second generation geometry tutoring system based on DC. The Geometry Tutor based on GTE has already been demonstrated as an effective alternative to homework problems, improving average student performance by about one standard deviation (Anderson et al., 1990). We have two reasons for believing that DC might lead to an even more effective tutor. The first has to do with DC's abstract planning abilities and the second has to do with the way DC uses the problem diagram.

6.4.1 Tutoring Advantages of an Abstract Problem Space. One of the difficulties involved in building an ITS is finding a way to communicate about the thinking that students do between their observable problem-solving actions. If the grain size of the problem-solving steps that the tutor allows is the same as the grain size of students' "thinking steps," then there is no problem. However, if the student and tutor are working at different grain sizes, then the tutor will be at a disadvantage in trying to diagnose student errors and provide appropriate feedback.

One of the complaints we have heard about the Geometry Tutor is that it does not provide very good global feedback. The feedback it provides is focussed locally on the *next* proof step the student might take, rather than more globally at the next few steps or an overall plan. Critics had the intuition that proof ideas can be born at a more global level. Our current research on geometry experts has identified this more global level and has characterized it in terms of DC's diagram configuration schemas. In other words, skilled geometry problem solvers think at a larger grain size than the grain size at which the Geometry Tutor works. A tutor working at the smaller grain size cannot give instruction at the larger one and thus, is disabled with respect to helping students reach skilled performance. However, a tutor based on diagram configuration schemas could give instruction at the larger grain size characteristic of skilled performance, and better aid students in reaching this level of skill.

6.4.2 Tutoring Advantages of a Diagram-Based Method. We are of the opinion that if you discover a clever way to solve problems in a domain, you should tell it to students. There are two caveats. First, the method must be one that is "humanly tractable." For example, although the Simplex method for linear programming is a clever way to solve certain optimization problems, it is not a tractable method for humans. Second, there must be a way to communicate the method so that it takes less time and effort for students to understand it than it would for them to induce it on their own through problem-solving practice.

We know that DC's problem-solving method is humanly tractable because it appears to be the method human experts are using. The next question is whether we can communicate the method to students effectively. Some ITS designers have addressed the problem of communicating about planning that occurs at a more abstract level than the level at which solution steps are written or executed. Some examples of the resulting tutoring systems include Bridge (Bonar & Cunningham, 1988), GIL (Reiser et al., 1988), and Sherlock (Lesgold, Lajoie, Bunzo, & Eggan, 1988). The basic approach is to develop a command language, usually menu-based and possibly graphical, which reifies this planning level.

Conveniently, we do not need to invent such a command language to reify DC's abstract problem space. Essentially, it already exists in the form of the problem diagram. What we envision is that rather than selecting an operator from a list of geometry rules as in the Geometry Tutor, students will select an operator from a list of diagram configuration icons. These icons will be the building blocks for proofs just as geometry rules were the building blocks for proofs in the Geometry Tutor.

6.4.3 Implications for Geometry Instruction in the Classroom. While our main focus has been on how DC can provide the basis for an improved ITS, our improved understanding of geometry problem solving may also have more general implications for how geometry is taught in the classroom. On one hand, the DC model is a theory of the internal thinking processes of skilled geometry problem solvers. On the other hand, it can be taken seriously as a new method for doing geometry proofs that can be explicitly taught in the classroom. In addition, the organization of knowledge in DC suggests an alternative task-adapted organization of the geometry curriculum. Typical geometry curricula are organized around topics, and focus on teaching the formal rules of geometry. Alternatively, a curriculum could be organized around diagram configuration schemas and have the structure in Figures 4a and 4b. The formal rules, then, could be taught in the context of how they are used to prove schemas. Such a task-adapted curriculum organization can help students remember rules and access them in the appropriate situations (Eylon & Reif, 1984).

REFERENCES

Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J.R., Boyle, C.F., Corbett, A., & Lewis, M. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence.*

Anderson, J.R., Boyle, C.F., & Yost, G. (1985). The geometry tutor. In *Proceedings of the International Joint Conference on Artificial Intelligence-85.* Los Angeles: International Joint Conference on Artificial Intelligence.

Anderson, J.R., Greeno, J.G., Kline, P.J., & Neves, D.M. (1981). Acquisition of problem-solving skill. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Erlbaum.

Bonar, J.G., & Cunningham, R. (1988, June). *Intelligent tutoring with intermediate representations.* Paper presented at the meeting of the ITS-88 [ITS = Intelligent Tutoring System]. Montreal, Canada.

Braine, M.D.S. (1978). On the relation between the natural logic of reasoning and standard logic. *Psychological Review, 85,* 1-21.

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

Cheng, P.W., & Holyoak, K.J. (1985). Pragmatic reasoning schemas. *Cognitive Psychology, 17,* 391-416.

De Groot, A. (1966). Perception and memory versus thought: Some old ideas and recent findings. In B. Kleinmuntz (Ed.), *Problem solving.* New York: Wiley.

Egan, D., & Schwartz, B. (1979). Chunking in recall of symbolic drawings. *Memory and Cognition, 17,* 147-158.

Ericsson, K.A., & Simon, H.A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Eylon, B., & Reif, F. (1984). Effects of knowledge organization on task performance. *Cognition and Instruction, 1,* 5-44.

Gelernter, H. (1963). Realization of a geometry theorem proving machine. In E.A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* [Report issued from MIT's computer science dept.] New York: McGraw-Hill.

Goldstein, I. (1973). *Elementary geometry theorem proving* (AI Memo 280). Cambridge, MA: MIT.

Greeno, J.G. (1976). Indefinite goals in well-structured problems. *Psychological Review, 83,* 479-491.

Greeno, J.G. (1978). A study of problem solving. In R. Glaser (Ed.), *Advances in Instructional psychology* (Vol. 1). Hillsdale, NJ: Erlbaum.

Greeno, J.G. (1983). Forms of understanding in mathematical problem solving. In S.G. Paris, G.M. Olson, & H.W. Stevenson (Eds.), *Learning and motivation in the classroom.* Hillsdale, NJ: Erlbaum.

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

Griggs, R.A., & Cox, J.R. (1982). The elusive thematic-materials effect in Wason's selection task. *British Journal of Psychology, 16,* 94-143.

Holding, D.H. (1986). *The psychology of chess skill.* Hillsdale, NJ: Erlbaum.

Holland, J.H., Holyoak, K.J., Nisbett, R.E., & Thagard, P.R. (1986). *Induction: Processes of inference, learning, and discovery.* Cambridge, MA: MIT Press.

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.* Hillsdale, NJ: Erlbaum.

Johnson-Laird, P.N. (1983). *Mental models.* Cambridge, MA: Harvard University Press.

Korf, R.E. (1987). Macro-operators: A weak method for learning. *Artificial Intelligence, 27,* 35-77.

Larkin, J. (1988). Display-based problem solving. In D. Klahr & K. Kotovsky (Eds.), *Complex information processing: The impact of Herbert A. Simon.* Hillsdale, NJ: Erlbaum.

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

Larkin, J., McDermott, J., Simon, D., & Simon, H.A. (1980b). Models of competence in solving physics problems. *Cognitive Science, 4,* 317-348.

Larkin, J., & Simon, H.A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science, 11,* 65-99.

Lesgold, A.M., Lajoie, S., Bunzo, M., & Eggan, G. (1988). *Sherlock: A coached practice environment for an electronics troubleshooting job* (LRDC Report). Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

Nevins, A.J. (1975). Plane geometry theorem proving using forward chaining. *Artificial Intelligence, 6,* 1-23.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W.G. Chase (Ed.), *Visual information processing.* New York: Academic.

Newell, A. (in press). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

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

Nilsson, N.J. (1980). *Principles of artificial intelligence.* Palo Alto, CA: Tioga.

Patel, V.L., & Groen, G.J. (1986). Knowledge-based solution strategies in medical reasoning. *Cognitive Science, 10,* 91-116.

Polk, T.A., & Newell, A. (1988). Modeling human syllogistic reasoning in Soar. *Program of the Tenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Reiser, B.J., Friedmann, P., Gevins, J., Kimberg, D.Y., Ranney, M., & Romero, A. (1988). A graphical programming language interface for an intelligent LISP tutor. In *Proceedings of the CHI'88 Conference.* New York, NY.

Rips, L.J. (1983). Cognitive processes in propositional reasoning. *Psychological Review, 90,* 38-71.

Sacerdoti, E.D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence, 5,* 115-136.

Simon, H.A., & Gilmartin, K.J. (1973). A simulation of memory for chess positions. *Cognitive Psychology, 5,* 29-46.

Smith, M., & Good, R. (1984). Problem solving and classical genetics: Successful vs. unsuccessful performance. *Journal of Research in Science Teaching, 21,* 895-912.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12,* 257-285.

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

Sweller, J., Mawer, R.F., & Ward, R.W. (1983). Development of expertise in mathematical problem solving. *Journal of Experimental Psychology: General, 112,* 639-661.

Unruh, A., Rosenbloom, P.S., & Laird, J.E. (1987). Dynamic abstraction problem solving in Soar. In *Proceedings of the AOG/AAAIC Joint Conference.* Dayton, OH.

Voss, J., Vesonder, G., & Spilich, G. (1980). Text generaton and recall by high-knowledge and low-knowledge individuals. *Journal of Verbal Learning and Verbal Behavior, 19,* 651-667.

Wason, P.C. (1966). Reasoning. In B.M. Foss (Ed.), *New horizons in psychology.* Harmondsworth, England: Penguin.