Carnegie-Mellon University

COLLEGE OF HUMANITIES & SOCIAL SCIENCES

DISSERTATION

Submitted in partial fulfillment of the requirements

for the degree of DOCTOR OF PHILOSOPHY

Title TUTORING CONCEPTS, PERCEPTS, AND RULES

IN GEOMETRY PROBLEM SOLVING

Presented by KENNETH R. KOEDINGER

Accepted by ______________________________ Jan 1, 1991
Thesis Supervisor, for the Committee Date

Approved by the Dean

______________________________ 1/25/91
Dean Date

Tutoring Concepts, Percepts, and Rules in Geometry Problem Solving

Kenneth R. Koedinger
Psychology Department
Carnegie Mellon University
Pittsburgh, PA 15213

Dissertation
submitted in partial satisfaction of the requirements for the degree of
Doctor of Philosophy in Psychology

ACKNOWLEDGEMENT

Thanks to the United States Air Force Graduate Fellowship program for tuition and financial support for the last three years of my graduate work. For intellectual support, I thank my advisor John Anderson and my committee members Jill Larkin, Kurt VanLehn, and Herbert Simon. Also, I thank Richard Lehrer at the University of Wisconsin who helped me to realize that cognitive psychology (not computer science) was what I really wanted to do and that CMU's Psychology Department was the right place to do it. In addition, Rich has been a mentor and a close friend. There are many other friends who deserve special thanks for intellectual and personal support during this time, most importantly, Christina Allen, Stephen Casner, Jonathan Cohen, Heather McQuaid, and Lael Schooler. Finally, I must thank my parents for their constant love and encouragement.

TABLE OF CONTENTS

ABSTRACT

This thesis is a mixture of basic and applied Cognitive Science research which has three parts. The first part describes a new theory of geometry proof skill and the empirical support for it. The second part describes the use of this theory in the development of a second generation intelligent tutoring system called ANGLE: **A** **N**ew **G**eometry **L**earning **E**nvironment. The third part describes a preliminary evaluation of ANGLE comparing it with the first generation Geometry Proof Tutor (GPT).

The theory of geometry proof skill presented here is based on a "step-skipping" analysis of the verbal reports of subjects solving proof problems. The data collected contradict previous theories that characterize geometry problem solving as heuristic search through a problem space of formal geometry rules. Instead of using such a localized and formal strategy, skilled subjects take a global planning approach that uses more intuitive conceptual and perceptual knowledge and leaves the formal details for last. A cognitive model of this approach (implemented as a computer program) turns out to be both more efficient than previous models and a better match to the human data. In addition, this model ties together a number of empirical results on the nature of human expertise and supports an inductive explanation of skill acquisition that is in contrast with the deductive approach typical of some dominant theories of skill acquisition.

While the basic research has uncovered a psychologically plausible method for successful and efficient geometric reasoning, the goal of the applied research is to see if geometry instruction can be improved by teaching this method to students. I built a computer tutor, called ANGLE, to test this idea. In ANGLE, the instruction is delivered both implicitly through the structure of the computer interface and explicitly through tutoring strategies and messages. The design of ANGLE was theoretically motivated by the cognitive model. For example, ANGLE's interface reifies the key underlying processes in the cognitive model and, in doing so, provides students with a novel notation with which to think about geometry.

With the creation of ANGLE it is possible to test the hypothesis that the development of more accurate models of skilled problem solving can lead to better instruction. We can test this hypothesis by comparing ANGLE with GPT, an earlier tutor for geometry based on a less accurate cognitive model. A preliminary "shakedown" study of this kind was performed and at this point, ANGLE is neither significantly better nor worse than GPT. The study revealed some limitations in the implementation of ANGLE and in the training curriculum. The thesis concludes with suggestions for remedies to these limitations that will put us in a better position to perform a more realistic test of the hypothesis.

INTRODUCTION

This thesis is a mixture of basic and applied Cognitive Science research which comes in three chapters. Chapter 1 describes a study and a new theory of skilled geometry proof problem solving. Chapter 2 describes the use of this theory in the development of a second generation intelligent tutoring system called ANGLE: **A N**ew **G**eometry **L**earning **E**nvironment. Chapter 3 describes a preliminary evaluation study of ANGLE comparing it against the first generation Geometry Proof Tutor (GPT).

The contributions of this thesis include:

1. A new methodology for verbal protocol analysis involving the identification of *step-skipping* with respect to the *execution space* of a domain.
2. A new theory of geometry expertise (DC) that accurately describes human behavior, has an efficient computer implementation, and pulls together a number of empirical results on the nature of human expertise.
3. A detailed characterization of the end-state of a complex learning process that challenges current learning theories and that can be used as a test-case for new learning theories.
4. A theory-based approach to the design of the interface and tutoring components of an intelligent tutoring system (ANGLE).
5. An initial test of the hypothesis that the development of more accurate and powerful cognitive models of problem solving can lead to major improvements in the instruction of problem solving, particularly within the context of an intelligent tutoring system.

CHAPTER 1.
A STUDY AND MODEL OF GEOMETRY PROOF PLANNING

In this chapter I present verbal report data and a computer simulation of geometry proof planning. This domain is a difficult one for human problem solvers and has been studied by a number of cognitive science researchers (Gelernter, 1963; Nevins, 1975; Greeno, 1978; Anderson, et. al., 1981). 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.

1.1 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 1.1 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 inference can be made, from these greater 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 domain, there is typically one problem space which is the most natural starting point for attempting to characterize human behavior in that domain[1]. It seems that quite often, particularly in math and science domains, this problem space is made up of operators which correspond one-to-one with the steps that problem solvers typically or conventionally write down in solving a problem in that domain. We call such a space the *execution space* of a domain as it corresponds with the way problem solutions are "executed" (though not necessarily with how they are planned). Applying this definition to geometry, we find the execution space operators to be the various definitions, postulates, and theorems that appear as the "reasons" in the steps of the conventional two-column format used for writing proofs.

The number of inferences reported at the start of this section were inferences within the execution space. Clearly, 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 (i.e., 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, Anderson, Boyle & Yost, (1985) built a geometry expert system as a cognitive 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

[1] Newell and Simon (1972, p. 144) refered to a "basic problem space" and identified a basic problem space in cryptarithmetic, chess, and logic.

used various contextual features as heuristics for predicting the relevance of an operator. (I review these systems and a couple others in Section 1.5.) While GTE provided a reasonably good model of students, as evidenced by the success of the Geometry Tutor (Anderson, Boyle, Corbett & Lewis, 1990), I found that the mode of attack of 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.

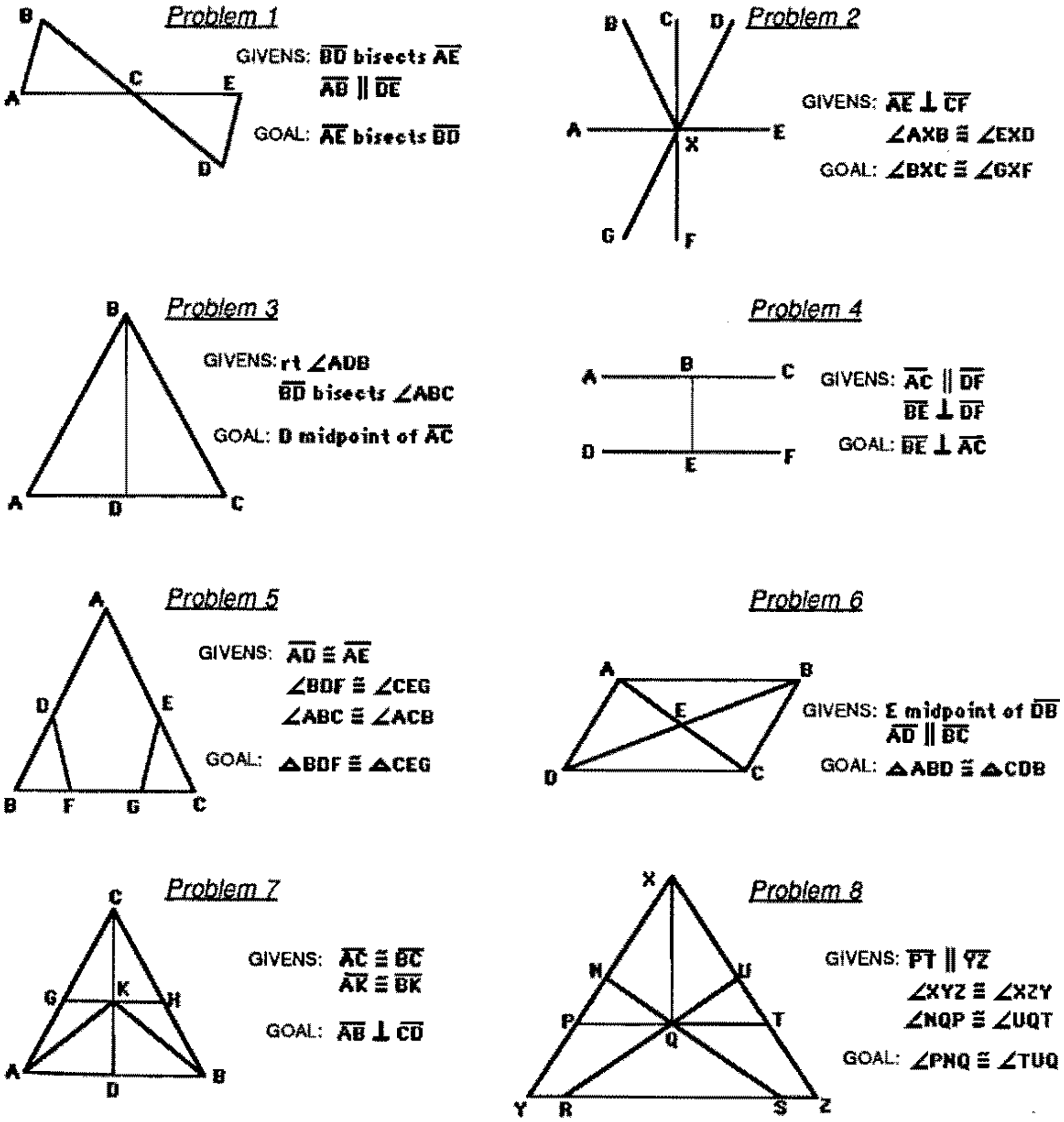

Problem 1
B
A
C
E
D
GIVENS: $\overline{BD}$ bisects $\overline{AE}$
$\overline{AB} \parallel \overline{DE}$
GOAL: $\overline{AE}$ bisects $\overline{BD}$
Problem 2
B
C
D
A
E
X
G
F
GIVENS: $\overline{AE} \perp \overline{CF}$
$\angle AXB \cong \angle EXD$
GOAL: $\angle BXC \cong \angle GXF$
Problem 3
B
A
D
C
GIVENS: rt $\angle ADB$
$\overline{BD}$ bisects $\angle ABC$
GOAL: D midpoint of $\overline{AC}$
Problem 4
A
B
C
D
E
F
GIVENS: $\overline{AC} \parallel \overline{DF}$
$\overline{BE} \perp \overline{DF}$
GOAL: $\overline{BE} \perp \overline{AC}$
Problem 5
A
D
E
B
F
G
C
GIVENS: $\overline{AD} \cong \overline{AE}$
$\angle BDF \cong \angle CEG$
$\angle ABC \cong \angle ACB$
GOAL: $\triangle BDF \cong \triangle CEG$
Problem 6
A
B
E
D
C
GIVENS: E midpoint of $\overline{DB}$
$\overline{AD} \parallel \overline{BC}$
GOAL: $\triangle ABD \cong \triangle CDB$
Problem 7
C
G
K
H
A
D
B
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$
GOAL: $\overline{AB} \perp \overline{CD}$
Problem 8
X
N
U
P
T
Q
Y
R
S
Z
GIVENS: $\overline{PT} \parallel \overline{YZ}$
$\angle XYZ \cong \angle XZY$
$\angle NQP \cong \angle UQT$
GOAL: $\angle PNQ \cong \angle TUQ$

Figure 1.1. Geometry problems given to subjects and solved by DC.

1.2 EXPERT HUMAN PROBLEM SOLVING

1.2.1 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 1.1 of an expert (Subject R) solving Problem 3 shown in Figure 1.1. The left side of the table contains the protocol and the right side indicates our coding of the subject's actions.

TABLE 1.1
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 ∠ADB Inference step 1: $\overline{AC} \perp \overline{BD}$
B2:	perpendicular on both sides [makes perpendicular markings on diagram];	
B3:	BD bisects angle ABC [marks angles ABD and CBD]	Reading given: $\overline{BD}$ bisects ∠ABC
B4:	and *we're done.*	Inference step 2: △ABD ≅ △CBD ****** Execution phase ****** In this phase, the subject refines and explains his solution to the experimenter.
B5:	We know that this is a reflexive [marks line BD],	
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].	

This expert had a reliable solution sketch for this problem in 13 seconds at the point where he said "we're done" (emphasis mine). He plans this solution sketch without looking at the goal statement (more on this curious behavior in Section 1.4.3) 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 problem – as he explains later: "we can determine anything from there in terms of corresponding parts".

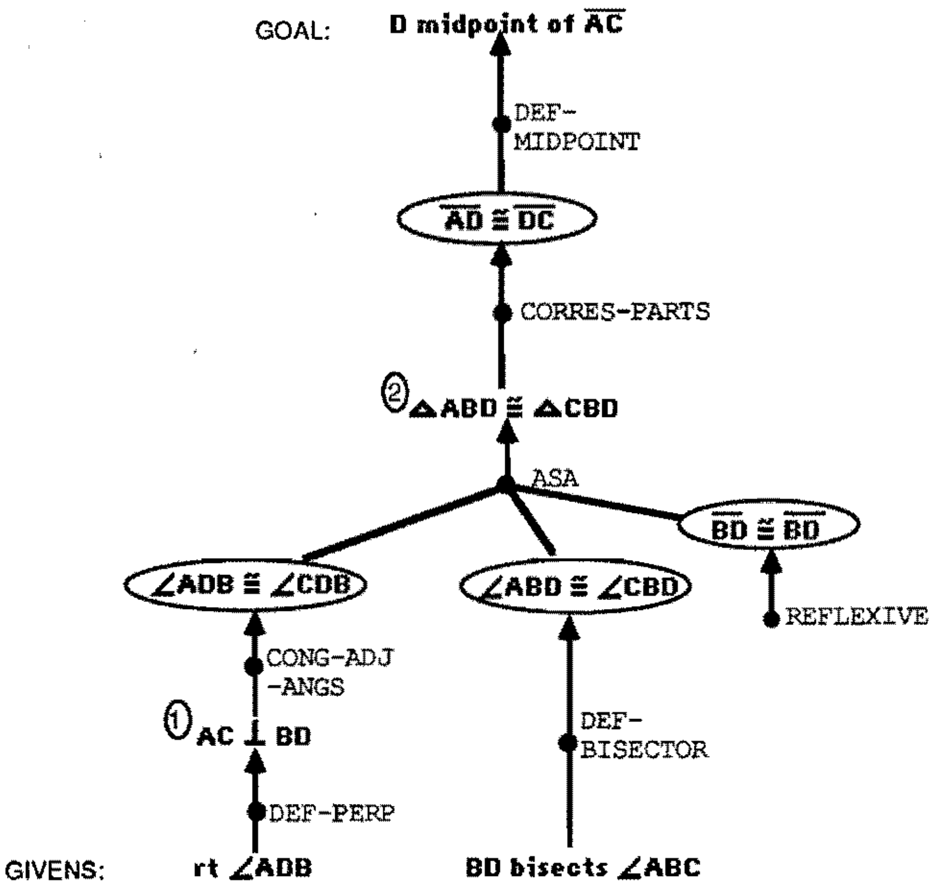


Figure 1.2. The final solution for Problem 3. The givens of the problem are at the bottom and the goal is at the top. The lines represent inferences with the conclusion at the arrow head, the premises at the tails, and the justifying geometry rule at the dot in between. The statements Subject R mentioned during planning (see Table 1.1) are numbered while the ones he skipped are circled.

Figure 1.2 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 problem are numbered in Figure 1.2 while the skipped steps are circled. Assuming this expert's verbalizations accurately reflect his working memory states (Ericsson and Simon, 1984), we conclude that the expert only makes certain key inferences in his search for a solution while skipping other, apparently minor inferences.

1.2.1.1 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, et. al., 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 1.4.1).

We distinguish 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 successfully refine an abstract plan because one of the details ignored in the abstraction process turned out to be critical.

A *safe abstraction* only ignores irrelevant details, i.e., 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., $\overline{AB} \cong \overline{CD}$) and measure equality statements (e.g., $m\overline{AB} = m\overline{CD}$) 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.

1.2.1.2 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, et. al., 1980b). In the field of problem solving this is known as the formation of macro-operators (Nilsson, 1972; Korf, 1985). 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, I 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 straight-forward 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, I was led to search for a new problem space for geometry theorem proving – one that was a safe abstraction of the execution space and that left out the same kind of steps as the experts did.

1.2.2 Use of the Diagram

Besides not working in the execution space, experts' inference making was largely tied to the diagram. I 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.

Let us 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 that they need to make further inferences. 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 – i.e., the search for applicable knowledge. The geometry diagram can also be used to aid problem search – i.e., 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 display-based reasoning approach (Larkin, 1988). Our approach elaborates on this one by showing how the organization of an external representation can be used to cue abstract planning operators. These abstract operators reduce problem search by packing many execution steps into a single inference.

Larkin and Simon's 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 I agree with this assumption, it seems 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.

1.3 THE DIAGRAM CONFIGURATION MODEL

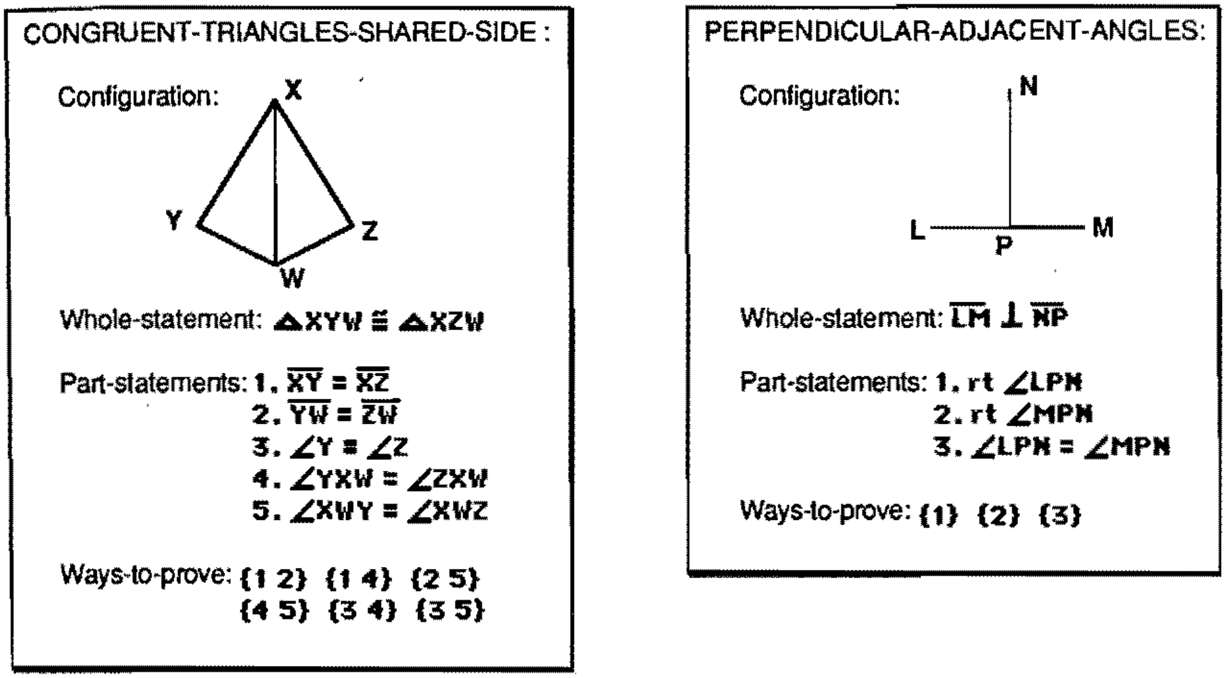
Based on the observations of experts, I tried to design a system for geometry theorem proving that would be both more powerful and more like human experts than previous systems. The model I came up with, the Diagram Configuration model (DC), has one major knowledge structure, diagram configuration schemas, and three major processes: diagram parsing, statement encoding, and schema search. Section 1.3.1 describes DC's diagram configuration schemas, while Section 1.3.2 describes DC's

[1] See chapter 2 in Newell (1990) for more discussion on the distinction between knowledge search and problem search.

processing components. Section 1.3.3 describes how DC uses a special class of diagram configuration schemas to avoid difficult algebra sub-proofs.

1.3.1 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 1.3 shows two diagram configuration schemas.



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 1.3. 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 $\overline{XY} = \overline{XZ}$ and $\overline{YW} = \overline{ZW}$ are proven, all the statements of the schema can be proven.

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 1.3, 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 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, {1 2}, indicates that if the part-statements $\overline{XY} = \overline{XZ}$ and $\overline{YW} = \overline{ZW}$ have been proven, the schema can be proven – that is, all the other statements of the schema can be proven.

The basic 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 1.1. 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 1.2) 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 CONGRUENT-TRIANGLES-SHARED-SIDE schema can be established as a result. We now describe the processes DC uses to recognize and establish schemas.

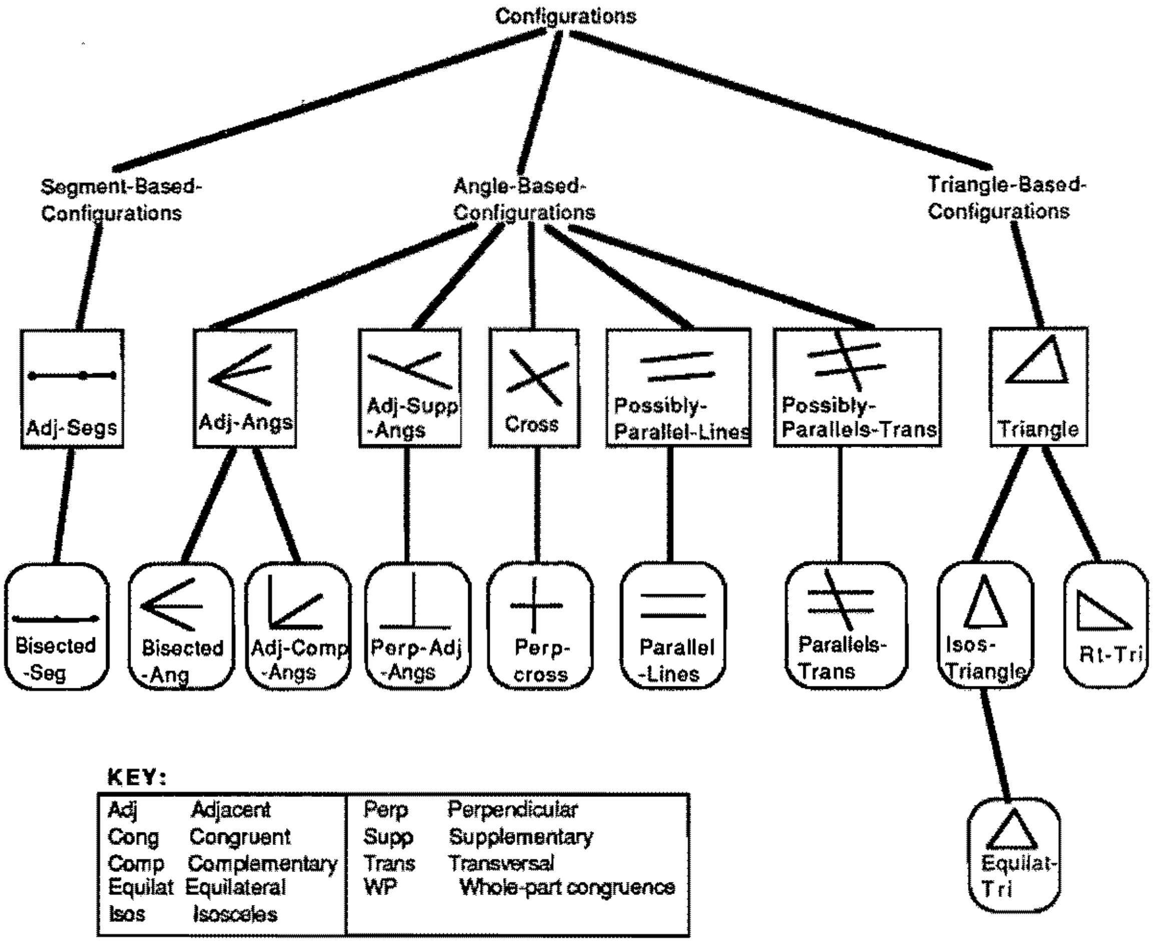
1.3.2 DC's Processing Components

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 has 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. In turns out the diagram parsing process plays a major role as we describe below.

1.3.2.1 Diagram Parsing and Schema Instantiation. Diagram parsing is the process of recognizing configurations in geometry diagrams and instantiating the corresponding schemas. Diagram parsing consists of both 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 1.4a and 1.4b 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 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 1.4a[1].



KEY:

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

Figure 1.4a. 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.

[1] 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 subjects were told that they could assume it in the problems they solved.

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 1.4a) or specializations of pairs of basic configurations (see Figure 1.4b). 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.

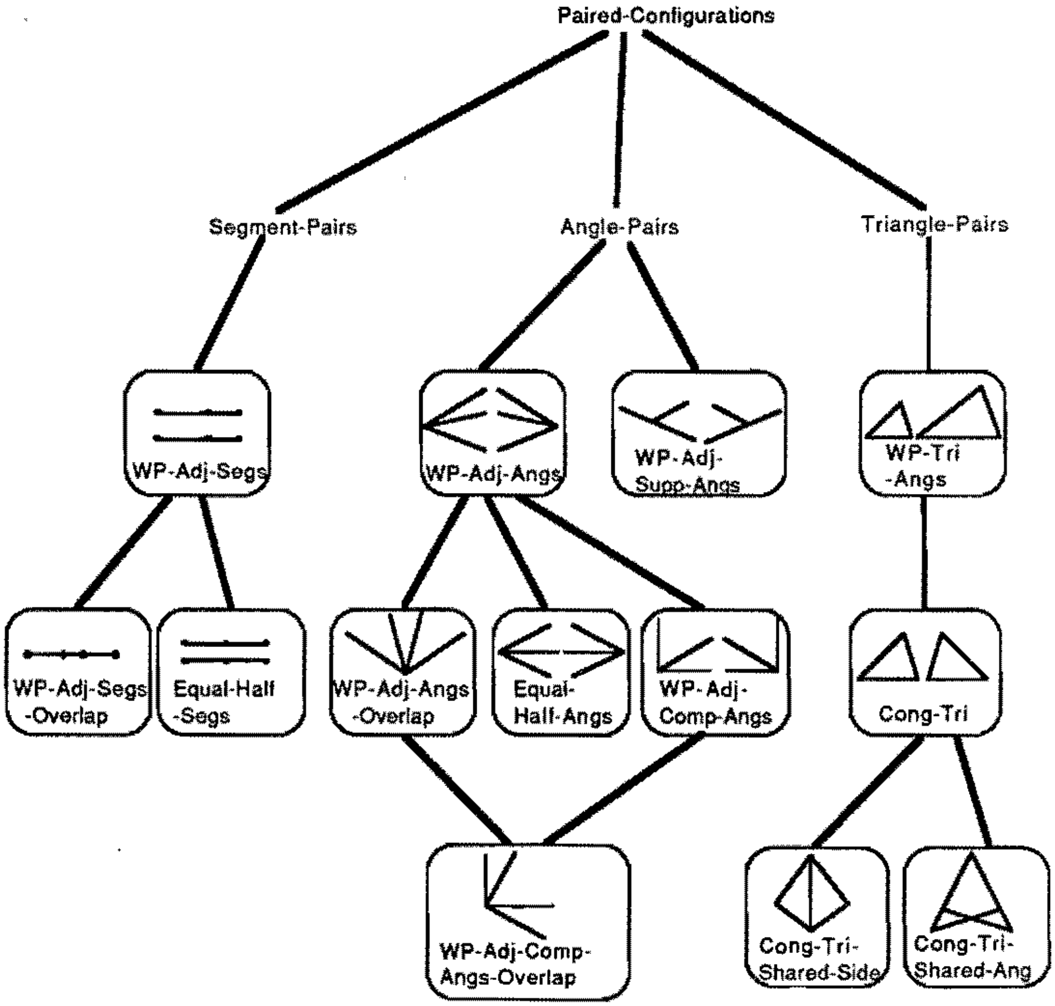


Figure 1.4b. The pairing of basic configurations where relationships hold among the corresponding parts of the configurations paired.

DC's diagram parsing algorithm corresponds with a very powerful visual process in humans. We make no claims that the internal steps of this algorithm match those 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.

The final result of diagram parsing is a network of instantiated schemas and part-statements. Figure 1.5 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.

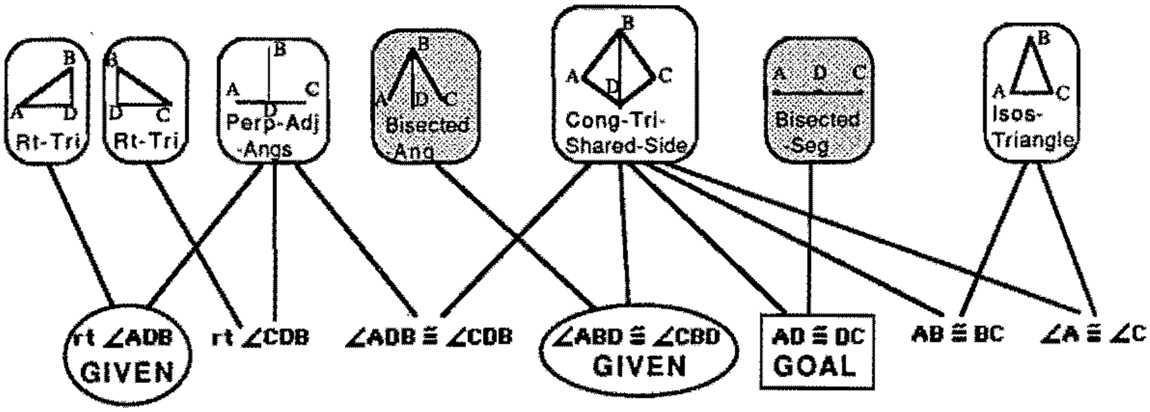


Figure 1.5. DC's solution space for Problem 3. The schemas DC recognizes during diagram parsing are shown in the boxes. The lines indicate the part-statements of these schemas. A solution is achieved by finding a path from the givens to the goal satisfying the constraints of the ways-to-prove slot of the schemas used.

1.3.2.2 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.

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 $m\overline{AB} = m\overline{BC}$ and $\overline{AB} \cong \overline{BC}$, 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, $\overline{BD}$ **bisects** $\angle ABC$, is the whole-statement of a BISECTED-ANGLE schema. DC encodes it by establishing its only part-statement $\angle ABD = \angle CBD$ as known (see Figure 1.5). Similarly, DC encodes the goal statement of Problem 3 by tagging the part-statement $\overline{AD} \cong \overline{CD}$ as desired.

1.3.2.3 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 that they may be concluded. Thus, it was only after the PERPENDICULAR-ADJACENT-ANGLES schema is established in the example problem that the CONGRUENT-TRIANGLES-SHARED-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 which is one statement away from satisfying ones 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 CONGRUENT-TRIANGLES-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 1.5.4.2, we relate this phenomenon to an "indefinite subgoaling" phenomenon identified by Greeno (1976).

1.3.3 Avoiding Algebra in the Diagram Configuration Space

One of the places where the Geometry tutor expert (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 1.4.1). Thus, it is worth discussing how DC avoids the black hole of algebra.

DC avoids the algebra sub-space 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.[1] These schemas are not ad hoc additions to remedy the difficulty with algebra sub-proofs. 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 sub-proofs. 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 1.4b that begin with WP. Our WHOLE-PART schemas are essentially the same as the WHOLE/PART schemas discussed in Anderson, Greeno, Kline and Neves (1981) and Greeno (1983).

A great variety of WHOLE-PART schemas can be formed by pairing any two component configurations which have corresponding parts (see Figure 1.4b). However, it would be misleading to suggest that all algebra sub-proofs can be solved using some WHOLE-PART schema. For example, the geometric proof of the Pythagorean theorem requires an algebra sub-proof 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 sub-proofs fall within the scope of WHOLE-PART schemas.

1.4 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 regularity in expert step-skipping that is contrary to straight-forward 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.

1.4.1 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 are multiple sources of

[1] While geometry textbooks have lots of theorems to skip commonly occurring steps, they do not have any theorems equivalent to the algebra schemas we are proposing (at least none of the textbooks we've seen do). There are two possible reasons for why they are absent. First, the utility of such theorems has been overlooked by textbook writers. I 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.

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.

1.4.1.1 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 1.1. 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 1.4a and 1.4b).

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 can only be acted on 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.

1.4.1.2 Results and Discussion. Table 1.2 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 doesn't suggest that we can just throw out algebra. Many problems require algebra sub-proofs 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.

TABLE 1.2
The Size of Three Different Problem Spaces on Problem 7.

	Execution Space	Execution Space without Algebra	Diagram Configuration Space
1st ply*	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 ply is all the operator instantiations that apply to the known statements produced by the previous ply.

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 effectively search the execution space. 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 1.1, where algebra is required, DC can solve the majority of these problems.

1.4.2 Accounting for Experts' Step-Skipping Behavior

In the process of planning a solution, our expert subjects made inferences that skipped more than 50 percent of the steps necessary for a complete solution in the execution space. In addition, we found that out 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.

1.4.2.1 Experimental Procedure. The data used for this analysis comes from four subjects' (B, K, J and F) verbal reports on one problem and one subject's (R) verbal reports on eight problems. Two of the single-problem subjects (B and K) were mathematics graduate students while the other two (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 problem solve. The four single-problem subjects were audio-taped as they entered their solutions using the interface of the Geometry tutor, while Subject R was video-taped 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".

1.4.2.2 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[1]. 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 1.2 illustrates the result of this analysis for the protocol of Subject R in Table 1.1.

1.4.2.3 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, e.g., 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, i.e., 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.

1.4.2.4 Results and Discussion. In the twelve subject-problem pairs, less 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 1.3 and 1.4 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 ($X^2(1) = 30.3$) 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.

[1] 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 1.3

Model-data Fit for All Subjects Solving the Same Problem.

		Predicted Mention		Predicted Skip	
Sbj	Prob#	Actually Mention	Actually Skip	Actually Mention	Actually Skip
R	7	3	0	3	2
B	7	2	0	1	3
K	7	3	0	1	6
J	7	2	0	1	3
F	7	3	2	3	9
Total		13	2	9	23

Table 1.3 shows the data for all five 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 $X^2(1) = 14.1$, $p<.001$. Table 1.4 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 $X^2 (1) = 22.0$, $p<.001$.

TABLE 1.4

Model-data Fit for One Subject Solving Eight Problems.

		Predicted Mention		Predicted Skip	
Sbj	Prob#	Actually Mention	Actually Skip	Actually Mention	Actually Skip
R	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

If the model fit perfectly, the totals for columns two and three in the Tables would be zero. The predictions are most deviant from the data in column three – the subjects mentioned fourteen[1] steps 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 $\angle ADB = \angle CDB$ and/or $\angle ABD=\angle CBD$ might be mentioned because both are needed to prove the CONGRUENT-TRIANGLES-SHARED-SIDE schema. To account for such situations our

[1] Adding the third column totals from Tables 3 and 4 yields seventeen. However, since subject R, problem 7 appears in both tables, we need to subtract three from seventeen to get the proper overall total of fourteen.

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 eleven, the number of false alarms (column 2) would be increased by six. 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 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 2 steps in Subject R's solution to Problem 8 (see Figure 1.1) that he skipped though we predicted he would mention them (see column two of Table 1.4). In this case, it appeared that the subject used a diagram configuration that combined two of DC's and thus was able to skip extra steps that the current version of DC cannot.

1.4.3 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 1.1), one by a forward search that was guided by the goal (Problem 2), and two using some backward inferences (problems 6 and 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 1.1). 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, et. al. (1980a) describes 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 purely work forward on relatively difficult problems as well as his ability to recognize he is done before reading the problem goal are further evidence for the DC model.

1.4.4 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 complete 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.

1.4.4.1 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 1.4a) can be paired to form an equation between the two non-right angles of the right triangle and the two adjacent complementary angles. We could supplement DC with such kinds of paired-configurations (as in Figure 1.4b) or, alternatively, the execution space search component could be used to discover such pairings.

1.4.4.2 Constructions. The computer simulation could also be made more complete is 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 require them. However, we feel that DC is particularly well-suited for adding a construction capability. The major decision points in solving proof problems which 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 definitively 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.

1.4.4.3 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 hasn't 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 over-specialized 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 $\triangle GCK \cong \triangle HCK$, **ISOS** $\triangle CGH$, $AB \parallel GH$, and $GH \perp CD$. 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 was done on demand, however, this extra work would not be necessary.

1.4.4.4 Diagram Drawing. While over-specialized 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, 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 accurately reflect a problem's givens.

1.5 COMPARISON WITH PREVIOUS GEOMETRY EXPERT SYSTEMS

Geometry theorem proving models have been developed by numerous researchers, most with primarily 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, Nevin'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.

1.5.1 Gelernter's Geometry Theorem Proving Machine

Gelernter's 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 the other subjects inferences were forward rather than backward.

1.5.2 Nevins' Model

Nevins (1975) presents a geometry theorem-proving program which 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. He claimed that human experts engage in much more forward inferencing than backward inferencing. Although he provided no evidence and was probably reacting to the purely backward reasoning strategy of most expert systems at that time, it is interestingly 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' 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 his 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, i.e., capable of inferences which require multiple steps in the execution space. However, they are embedded in complex procedures within Nevins' model and are not clearly and uniformly represented like diagram configuration schemas are. Nevins' 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' 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' 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. Also, in addition to DC's computational advantages, we have provided empirical evidence that human experts solve problems like DC.

1.5.3 The Geometry Tutor Expert System

The Geometry tutor expert system (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. 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 1.1. 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 actually prove the two triangles congruent has a higher aptness rating than the rule above which in turn has a higher aptness rating than a rule which simply suggests 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, i.e., ACT*, that 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 that we observed experts using.

1.5.4 Greeno's Perdix

Greeno used verbal report data from geometry students as the basis for the design of a geometry theorem proving model called Perdix (Greeno, 1978). 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 so as 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-operator-like operators. With respect to algebraic reasoning, Perdix contained operators which 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 non-deductive or "indefinite" reasoning that both students and experts appear to engage in.

1.5.4.1 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 else 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 it) 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 which 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.

1.5.4.2 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, e.g., side-angle-side, subjects attempt to prove any of the corresponding parts statements they can. These statements are indefinite goals because they are not associated with any definite rule. DC accounts for indefinite goals as 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 ▲ACD ≡ ▲BCD and/or ▲AKD ≡ ▲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 the Nevin's model and Perdix (Greeno, Magone, and Chaiklin, 1979) are capable of introducing constructions into the geometry diagram allowing them to solve a class of problems that DC cannot as it currently does not have a construction capability. However, as noted in Section 1.4.4, we feel that DC is particularly well-suited for adding a construction capability.

1.6 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 that 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, and 4) how these findings might be applied to improve geometry instruction. Chapter 2 follows up on this last point by focussing on how these findings have been used in the development of a 2nd generation intelligent tutoring system for geometry.

1.6.1 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 below in order 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 which 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 Rips, 1983; Braine, 1978).

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 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 our 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 by simply 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 which 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 – i.e., the mapping of a schema to a situation where the schema-based inference is incorrect (Holland, et. al., 1986).

DC has similarities with 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 lots of 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[1]. According to the mental model approach, what is left for the problem solver to do is properly annotate the model and read-off the conclusion. This is essentially what we propose experts do – they annotate the diagram, on paper or in their mind's eye, by noting established relationships.

However, the annotation process is not as straight-forward 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 it 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.

1.6.2 Contributions to the Study of Human Expertise

1.6.2.1 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, McDermott, Simon, & Simon, 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 done between different subjects, the invited inference is that as a person acquires skill

[1] 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 altitude 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.

in one of these domains their 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 have observed an expert (Subject R) exclusively working forward on a number of the simpler problems we asked him to solve. This ability to essentially solve certain problems 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 successfully work forward.

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 successfully solve problems 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 successfully performed, it should not surprise us 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 successfully work forward, 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 successfully work forward. He suggests that experts use *schemas* to classify problems into categories that carry implications for appropriate moves to make. He 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 sub-problems based on recognizing prototypical images in the problem diagram and the retrieval of the relevant sub-proof.

The key point is not so much that experts will necessarily prefer working forward. Rather, it is that as a result of the their superior skill, experts are *capable* of successfully working forward without recourse to backward reasoning. Knowledge in the form of schemas is what allows 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 8 schemas (see Table 1.2). Still, all of our experts did some backward reasoning on

Problem 7. It was only on simpler problems, like 3 and 5 with only 3 relevant schemas, that Subject R performed a purely forward search.

1.6.2.2 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, Turner, Polson, & Atwood, 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 no better than novices at remembering boards with randomly placed pieces.

While these recall abilities are correlated with game playing skill, it has yet to be decisively established whether they are a necessary part of game playing skill or whether they are merely a side-effect of spending lots 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 which is 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 towards 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 findings in the domain of 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 which provide the basis for an abstract problem space in which players can plan and evaluate multiple-move strategies.

1.6.3 DC's Relation to Comprehensive Theories of Cognition

In 1972 Allen Newell gave his well known "20 questions" talk (Newell, 1973) in which he argued that to avoid spinning our wheels in cognitive science research we need to begin to integrate local hypotheses and domain 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; Newell, 1990; Johnson-Laird, 1983; Holland, et. al., 1986). In this section we try to place DC in terms of two of these theories, ACT* (Anderson, 1983) and Soar (Newell, 1990). 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 to not change the resulting behavioral predictions. Consider the TRIANGLE-CONGRUENCE-SHARED-SIDE schema in Figure 1.3. This schema can be represented as 6 production rules whose left-hand sides correspond to the 6 ways-to-prove of the schema and whose right-hand sides contain 5 actions which correspond with the 5 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 to productions? In terms of problem solving behavior the answer is probably no. However, another question we need to ask with respect to the 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, e.g., problem descriptions, problem constraints, example solutions, etc., 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[1].

There are both empirical and computational reasons to doubt that 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 that indicates 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

[1] To cut off a potential confusion based 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.

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 marco-operator-like 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 bare 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 in the process of executing an abstract plan, subjects could not always immediately fill in the steps they had skipped during planning. For example, in Problem 7 subjects would plan to prove the goal from $\angle ADC \cong \angle BDC$, 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 that 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-operator-like 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 which could make inferences between statements which are whole-statements of schemas, for example, it doesn't contain an operator 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?

1.6.4 Implications for Geometry Instruction

While the main focus of Chapter 2 is on how DC can provide the basis for an improved intelligent tutoring system, 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 new method for doing geometry proofs which 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 1.4a and 1.4b. The formal rules, then, could be taught in context of how they are used to prove schemas. Such a task-adapted curriculum organization can help students to remember rules and access them in the appropriate situations (Eylon & Reif, 1984).

CHAPTER 2.
ANGLE: A NEW GEOMETRY LEARNING ENVIRONMENT

Tutor building is a time consuming task often involving as many as 10 person years of effort. This chapter reports on ANGLE in a fairly early stage of its development – after about 1.5 person years of effort. Our development pace is certainly faster as a result of the previous research on the Geometry Proof Tutor (GPT) and the LISP tutor (Anderson, Boyle, Corbett, & Lewis, 1990). In particular, ANGLE's tutoring scheme is a version of the *model tracing* approach developed for these tutors. In model tracing students are tutored by matching their behavior against a cognitive model of successful and buggy performance in the domain. The most essential difference between ANGLE and GPT is that the cognitive model has changed: ANGLE traces students relative to the diagram configuration space, while GPT traces students relative to the execution space.

Yet despite the general contributions of prior research, we basically started from scratch when it came to designing the specifics of the interface and of the tutoring strategies and messages. By taking the DC problem solving theory seriously as a driving force in tutor design, we arrived at a significantly different kind of interface and a different set of specific tutoring strategies and messages.

2.1 MOTIVATIONS FOR TUTOR DESIGN: FROM DC TO ANGLE

2.1.1 The Implicit Plan Problem

One of the difficulties involved in building an intelligent tutoring system (ITS) is finding a way to communicate about the thinking that students do between their observable problem solving actions. We call this the "implicit plan" problem. If aspects of student planning cannot be reified or articulated by the ITS, it is not possible for the system to monitor this planning nor provide relevant advice. For example, in data collected from GPT, it was found that the better performing students were taking significantly more time at the beginning of each problem solving step. It appears they were doing some planning that cannot be represented in the GPT interface and thus, cannot be tutored by GPT. In contrast, the poorer students were more quickly jumping into a step indicating, perhaps, that they were not performing the kind of implicit planning that made the better students' successful. Because this implicit planning is not represented in GPT, the system cannot tutor it and thus, the poorer students are left on their own to discover it.

In support of this interpretation, a common complaint about GPT is that it does not provide very good global feedback. The feedback focuses locally on the *next* proof step the student might take rather than more globally at the next few steps or an overall plan. Many critics have 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. Thus, ANGLE can address the implicit plan problem – we can reify the planning process in the interface and so, open it up for discussion and instruction.

2.1.2 Reifying Planning: Advantages of a Diagram-Based Method

ANGLE follows from the following instructional philosophy: 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 tractable method for humans. Second, there must be an effective way of communicating the method that avoids being so complicated that students spend too much time trying to understand the method itself and too little time actually learning to solve problems in the domain.

We can be reasonably certain that DC's problem solving method is humanly tractable because it appears to be the method skilled problem solvers are using. The next question is whether we can effectively communicate the method to students. Some ITS designers have addressed the problem of communicating about the abstract planning occurring above the level at which solution steps are executed. Examples of the resulting tutoring systems include Bridge (Bonar & Cunningham, 1988), GIL (Reiser, et. al., 1988), and Sherlock (Lesgold, et. al., 1988). The basic approach is to develop a command language, usually menu-based and possibly graphical, which makes this planning level concrete. While such languages clearly provide a means for communicating about planning, they are at risk of violating the second caveat noted above. In other words, it is not always clear whether these command languages pay off or whether they side-track the student with the extra burden of learning a complicated language.

For the most part, 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. Rather than choosing an operator from a list of geometry rules as in GPT, in ANGLE students select an operator from a list of diagram configuration icons. These icons are the building blocks for proofs in ANGLE just as geometry rules were the building blocks for proofs in GPT.

2.1.3 A Methodology for Theory-Based Tutor Design

In this chapter, we discuss the design of ANGLE and in particular, how its major components, the expert, interface, and tutoring component have been influenced by the cognitive model of problem solving laid out in Chapter 1. The expert component is the subject-matter knowledge of the tutor, the interface component determines how the student interacts with the system, and the tutoring component provides the verbal advice the system gives the student.

TABLE 2.1
Ways to Communicate a Problem Solving Model in an ITS.

	Representations	Processes
Interface	*Notations*	*Actions*
Tutor Advice	*Vocabulary*	*Hints and Explanations*

Table 2.1 summarizes a methodology for theory-based tutor design that indicates a way to translate from a cognitive model of problem solving into design specifications for the interface and tutoring components of an ITS. Given a model of successful

students' implicit planning, one needs to find ways to communicate the knowledge representations and cognitive processes that make up this model.

The interface component provides an implicit form of instruction in that students can learn through their perception and interaction with it. Thus, one avenue to teaching the problem solving model is to invent interface *notations* and *actions* that reify the underlying representations and processes of the model. Interface notations should mirror the model representations such that students can begin to internalize these representations through repeated perception and use of the notations. Similarly, interface actions should mirror the model processes such that students can learn these processes through repeated performance of these actions.

While the interface component provides an implicit form of instruction, the tutor component provides a direct way to instruct students on the representations and processes of the problem solving model. Both ANGLE's tutoring messages and the off-line text materials use specific *vocabulary* to directly articulate the schema representations used in DC. Similarly, the content and order of ANGLE's *hints* and *explanations* have been designed to articulate the problem solving processes (e.g., diagram parsing, abstract planning) incorporated by DC.

The following three sections describe the three major components of ANGLE: the expert, interface, and tutor components.

2.2 THE EXPERT COMPONENT

2.2.1 Summary of DC: ANGLE's Expert Component

The expert component is the core of an ITS as its capabilities constrain what can and cannot be done in the interface and tutor components. Essentially ANGLE's expert component is the computer simulation of DC described in Chapter 1. Some differences in implementation details are described in the following section. Here we simply summarize the important differences between DC and the expert component of the GPT.

TABLE 2.2
Differences Between GPT's and ANGLE's Expert Components

	Knowledge	Strategy
GPT Expert	*Formal Rules*	*Local Heuristic Search*
ANGLE Expert	*Percepts & Concepts*	*Global Planning*

See Table 2.2. In GPT's expert component, the knowledge was organized around the *formal rules* of geometry. These logical rules appeared as parts of production rules that also contained context cues to indicate likely situations in which the logical rule would be usefully applied. These productions constitute a bidirectional heuristic search strategy that has a *local* view in that decisions are only made about next logical step in the proof (either forward or backward). In contrast, ANGLE's expert component is organized around geometric *percepts* and associated *conceptual* knowledge of the logical properties (part-statements) of these percepts. It searches a different problem space that takes a more *global* view in that planning decisions are made that typically involve many steps in the proof.

2.2.2 Efficiency Considerations

This section provides some efficiency related details about how DC is used as the expert component of ANGLE.

2.2.2.1 Off-line Problem Model Creation. The efficiency of ANGLE is improved by running DC off-line and producing a *problem file* for each problem ANGLE is going to tutor. ANGLE inputs the problem file just before the student begins working on a particular problem.

It turns out that most of the computation necessary to model student problem solving (on a particular problem) can be done off-line. This can be accomplished because of the way diagram parsing and schema search can be decoupled: all of the diagram parsing is done off-line and only the schema search needs to be done on-line. The result of the off-line diagram parsing is a problem file containing a network of part-statements and schemas. Such a network is illustrated in Figure 1.5. As discussed in Section 1.3.2.1, this network is a *concrete model* of the problem space of a particular problem – no more pattern matching of predicates need be done to find solutions to this problem (or to follow any legal path for that matter). We'll refer to the network for a particular problem as the *problem model.* Only constant to constant matching is necessary for ANGLE to check students' solution steps. This approach allows for fast responses to the student actions (< 1 sec) and is a significant efficiency improvement over GPT as argued below.

There is no loss in generality as a result of this efficiency improvement. ANGLE can still follow the student along any path of legal inferences whether or not they are needed to prove the problem goal.

While the generation of the problem model is done off-line, ANGLE must still do some problem solving search when asked to generate strategic advice. This advice is given when the student is stuck and is designed to hint at a good next step the student might take. In the Tutor Component section we describe the nature of the strategic advice given. The Expert Component's role is to identify the best next step the student could take within the context of what he or she has already done. ANGLE suggests only forward reasoning steps. It determines the best next step by finding a schema which meets to criteria. First, it must be a schema that the student could currently prove using any of the statements he's already proven or any of the part-statements which are consequences of the schemas he's already proven. Second, it must be the (or one of the) schema(s) that is on the closest path to the goal. The Expert does a backward breadth-first search through the problem model to find this schema. As noted above this can be done quite quickly, about 1 second. This is both because problem models are relatively small and because no pattern matching is necessary to search them.

We decided to tutor only forward inferences and not backward ones. This decision was based on classroom experience with an earlier version of GPT which provided backward reasoning hints under certain circumstances. The average student found this very confusing and such hints were eliminated from the tutor so that the current version of GPT only tutors forward inferences. This issue is certainly worth more exploration, but we wanted to avoid this potential roadblock with this first instantiation of ANGLE. (In fact, a new tutoring scheme for ANGLE is currently in the works which tutors backward reasoning.)

2.2.2.2 Evaluating Efficiency Improvements. Identifying which of the possible solutions the student is working on is somewhat problematic for GPT. ANGLE finesses this issue by being able to quickly generate potential solution paths on-line. Instead of trying to match a complex set of student steps against a series of possible solutions, ANGLE simply finds the shortest path to the goal from the steps the student has already generated. It gives advice on the next step in this path.

The MacII hardware is about 3 times faster running Common LISP than the Xerox D-machine running its Interlisp. However, the efficiency of ANGLE's problem model makes it even faster. Average problem loading time for GPT is about 35 seconds for "easy" problems and about 130 seconds for "hard" problems. In contrast, ANGLE doesn't suffer so much on the hard problems: about 10 seconds for the "easy" problems (about the expected 3 times faster), but only about 13 seconds for the "hard" problems -- a ten-fold gain largely attributable to the concrete problem model. GPT response times are about 15 seconds while ANGLE's are about 1 second. Again, a gain well beyond the hardware difference.

2.3 THE INTERFACE COMPONENT

2.3.1 Motivation for Interface Component Design

An obvious design principle for a tutor interface is that it should be easy to use and learn. A less obvious principle has to do with the role of interface as a subtle, yet ever-present, form of instruction. This implicit instruction comes both in the form of the *notations* used in the computer interface and also in the *actions* allowed by the interface. A theory-based approach to tutor design can guide the creation of these interface notations and actions. Notations should be created which mirror the important underlying representations of the problem solving model, while actions should be created which mirror the important underlying processes of the problem solving model. In this way, students can begin to internalize both the desired representations as they use the interface notations and the desired processes as they perform interface actions.

2.3.1.1 ANGLE Interface Notations. The ANGLE interface includes a number of examples of notations which reify representations in the DC model. The most prominent and important are icons for representing the generic schema categories and for representing student-selected instances of these schemas (these are discussed below in Section 2.3.2 and illustrated, for example, in Figures 2.1 and 2.3). These icons provide a concrete image to which students can attach the related conceptual knowledge about the part-statements and ways-to-prove. In particular, they reinforce the perceptual character of schemas.

Following GPT, ANGLE incorporates a graph representation of proofs in contrast to the two-column format of traditional geometry instruction. (The construction of a proof graph is described below in Section 2.3.3 and illustrated in Figures 2.3 through 2.12.) This proof graph notation reifies the search process:

1) by explicitly indicating how a correct solution must be a chain of steps linking the givens to the goal,

2) by allowing the posting of subgoals as possible future links in the solution chain, and

3) by explicitly indicating dead end solution attempts which are a common part of problem solving (even for experts).

Following a conventional notation used on paper, ANGLE uses hash marks in the diagram to indicate part-statements that have been proven. (This is described below in Section 2.3.2.1, see, for example, the change in the diagram from Figure 2.5 to 2.6.) These markings reify the equivalence class nature of segment and angle congruence in contrast to the binary relationships of the formal notation. In other words, to indicate that 3 angles are all congruent to each other using the hash mark notation, one can mark all three with the same marking – the marking serves as a token of the equivalence class containing all three angles. In contrast, the formal notation requires three binary statements to represent this situation, for example, $\angle 1 \cong \angle 2$, $\angle 2 \cong \angle 3$, and $\angle 1 \cong \angle 3$.

Finally, as a further aid to the acquisition of schemas, ANGLE highlights a schema within the problem diagram whenever the mouse passes over the corresponding schema instance icon in the proof graph. This is intended to reinforce the relationship between the schema and the rest of the diagram. This is a kind of dynamic notation which cannot be feasibly employed with paper and pencil.

2.3.1.2 ANGLE Interface Actions. The DC theory influenced interface decisions involving both the type of actions allowed and the grain size of these actions. Following the major processes in DC, ANGLE interface actions are broken down into (1) diagram parsing actions, (2) planning actions, and (3) execution actions. The grain size of parsing and planning actions is designed to emphasize schemas, while the focus on statements and rules is left for execution actions.

The diagram parsing actions are those done in order to post schemas. As described below (Section 2.3.3.1), first the student selects a schema type and then indicates the lines within the diagram that make up an instance of this schema (see Figure 2.2). This particular way of constructing a statement, as opposed to the way a student constructs one in GPT or on paper, is meant to reinforce the relationship between the schema instance and the problem diagram in which it is embedded.

The planning actions are those done in order to justify schemas and part-statements. To some extent these actions are more elaborate than those made in a two-column proof on paper: In ANGLE, the student must explicitly indicate the premises that lead to a conclusion (that is, by drawing the lines between them), while in the typical two column proof these links are only implicit. On the other hand, the planning actions are for the most part less elaborate than those required for a two column proof. Certain details required in a two column proof can be left out while constructing a plan. Students can omit a) certain statements usually required in a complete proof, for example, the reflexive statement $\overline{CK} \cong \overline{CK}$ in the first inference in Figure 2.7, and b) the rules or "reasons" that usually appear in the right column of a two-column proof.

Any tutor interface (or notational scheme for that matter) is implicitly taking an instructional stance about what things are hard and/or important to learn and what things are not. Three important aspects of the instructional stance taken by ANGLE's interface are worth making explicit: (1) learning about the logical linkages between proof steps is hard and important, (2) learning the details of proof execution is less important and perhaps less hard than learning proof planning, and (3) learning how to parse geometry diagrams into particular chunks (DC-schemas) is hard but it is important for successful search in a vast problem space. The first point is shared by GPT's interface, but not by the two-column notation used on paper. The last two points are special to ANGLE's interface.

The following sub-sections provide more details on the nature of the ANGLE interface.

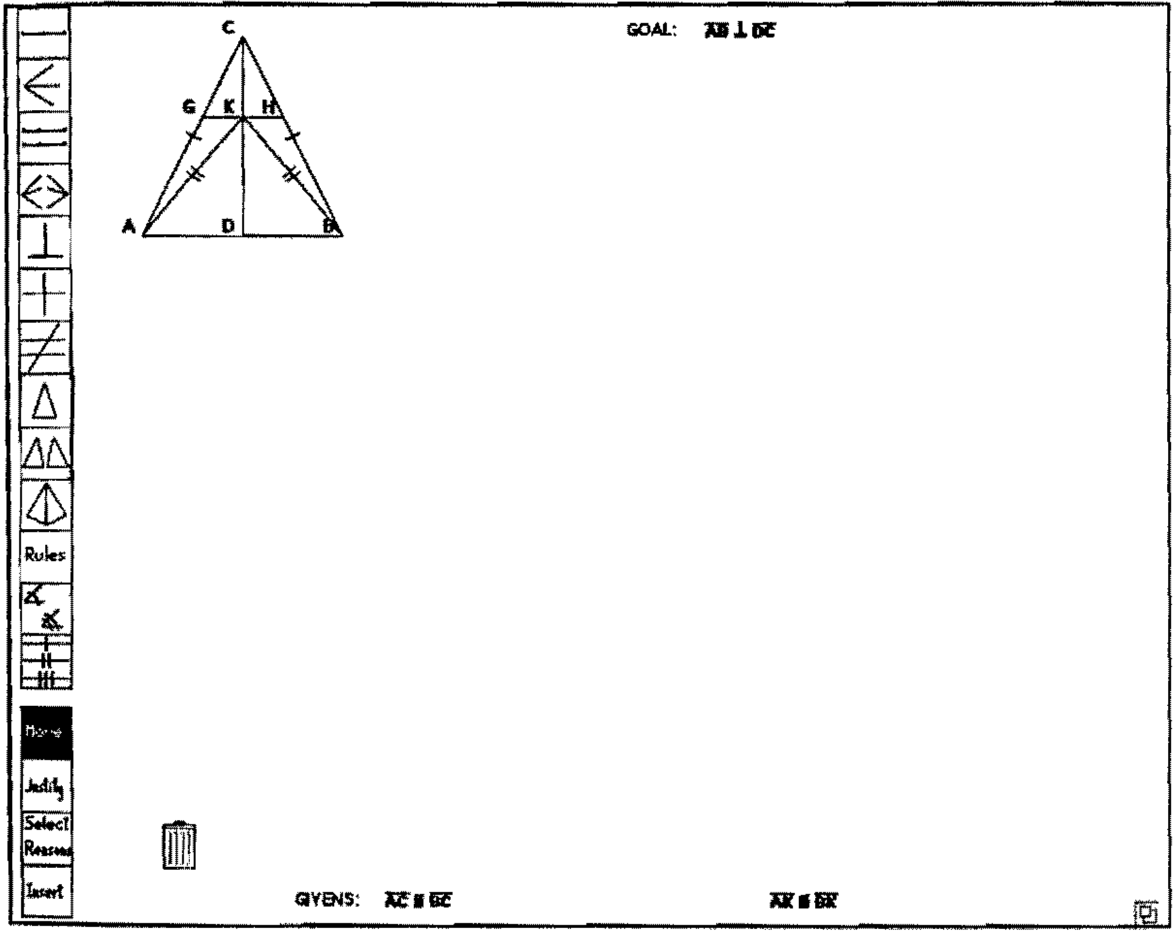

C
G
K
H
A
D
B
GOAL: $\overline{AB} \perp \overline{DC}$
Rules
Move
Justify
Select Reason
Insert
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

Figure 2.1. The ANGLE interface at the start of a problem.

2.3.2 Screen Layout

Figure 2.1 shows the ANGLE screen at the start of a problem. On the upper-left edge of the window is an icon-menu containing icons used to indicate various schema types as well as segment and angle congruence statements (the bottom two). The menu below it is the *mode-menu* where the current mode is always highlighted. The Move mode is the default mode as shown in Figure 2.1. The problem givens appear at the bottom of the window, the problem goal at the top, and the problem diagram at the top-left. Just like in GPT, the proof is represented as a graph linking the problem givens to the problem goal. Figure 2.7 shows a complete proof plan, while Figure 2.12 shows the complete solution after execution.

The elements of the proof graph include two types of statements: 1) schema instances, displayed with both the schema's whole-statement and a miniature picture of how the schema instance appears in the problem diagram, and 2) part-statements, displayed in the standard way.

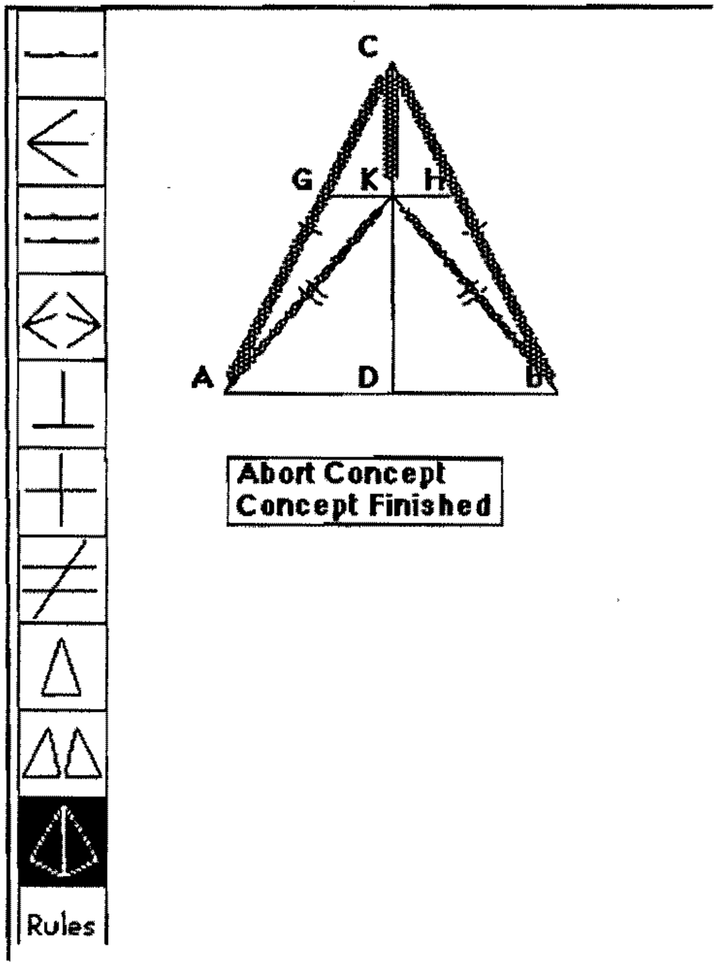
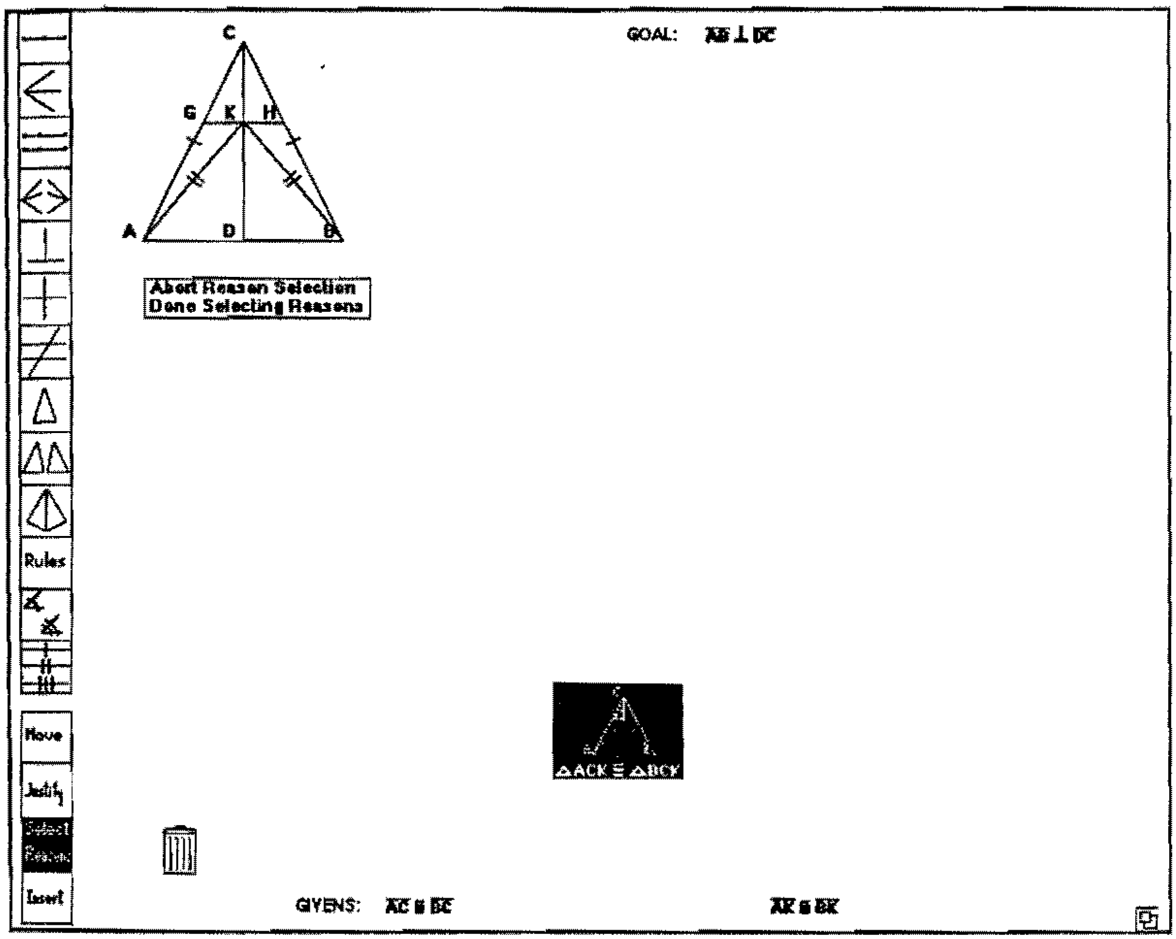


Figure 2.2. Selection of a schema statement or "concept" as they were called to students. This method of schema selection is intended to reify the diagram parsing process.

2.3.3 Interface Actions

2.3.3.1 Planning Actions. Students enter planning steps in two sub-steps, statements must be *posted* first and then can be *justified*. To post a schema statement, the student starts by selecting a schema type from the icon-menu on the left. Next, she mouse-clicks on lines in the problem diagram that make up a particular instance of this schema. These lines are highlighted as shown in Figure 2.2. The figure shows the selection of an instance of the CONGRUENT-TRIANGLES-SHARED-SIDE schema – this particular instance is the first step along the solution path. After the student mouse-clicks on "Concept Finished", the schema statement is *posted* – that is, it appears on the screen. The student is then free to drag the statement (using the mouse) into the proof area.

The student is free to post other statements as they wish. Typically, she will attempt to *justify* the statement just posted. To justify a statement, the student starts by mouse-clicking on `Justify` in the mode-menu. Next, she mouse-clicks the statement she wants to justify, in this case **▲ACK ≅ ▲BCK**. This statement is highlighted and the mode-menu switches to `Select Reasons`, and a "Done-Abort" menu appears below the problem diagram. The new state is shown in Figure 2.3.



GOAL: $\overline{AB} \perp \overline{DC}$

GIVENS: $\overline{AC} \cong \overline{BC}$ $\overline{AK} \cong \overline{BK}$

Figure 2.3. Justifying a schema statement. The student is indicating the intention to justify **▲ACK ≅ ▲BCK**. Next, she will select the given statements as the reasons for this statement.

The student now mouse-clicks on the reasons or statements from which **▲ACK ≅ ▲BCK** is justified. In this case, the student mouse-clicks on the two givens, $\overline{AC} \cong \overline{BC}$ and $\overline{AK} \cong \overline{BK}$. As she does, lines are drawn from these statements to **▲ACK ≅ ▲BCK**. To finish selecting reasons, she mouse-clicks on "Done Selecting Reasons" in the menu below the problem diagram. The result is shown in Figure 2.4. The mode switches back to `Move` and the student is free move any piece of the proof graph they've created.

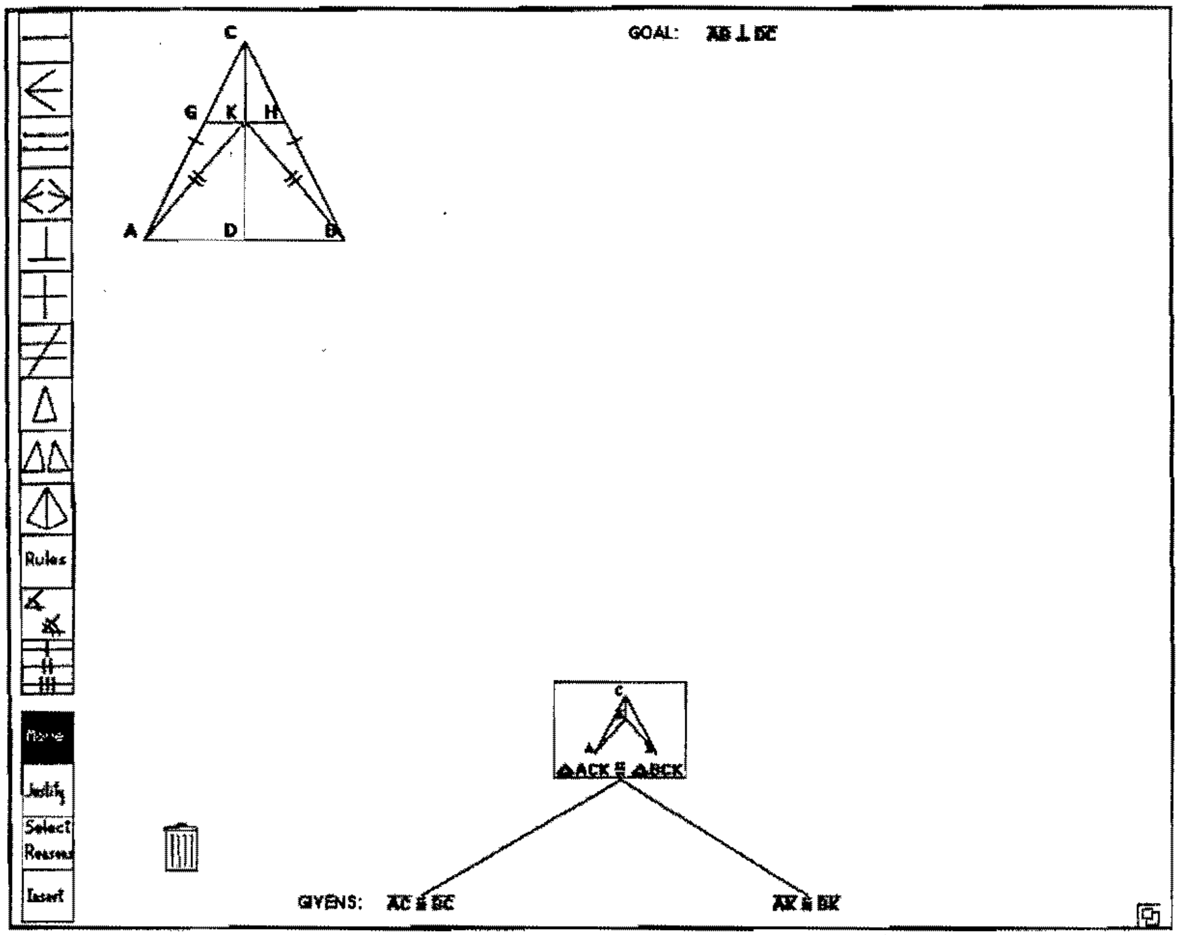


Figure 2.4. Completion of a schema justification action.

Posting a part-statement, for example, **∠BCD ≅ ∠DCA**, is analogous to posting a schema statement. First, the student mouse-clicks on the angle congruence icon in the icon-menu (second from the bottom). Then, she mouse-clicks near the vertex of one of the angles between the two rays that form it, that is, right where one would mark the angle with a pencil on paper. She does the same for the other angle. At this point, the screen appears as shown in Figure 2.5.

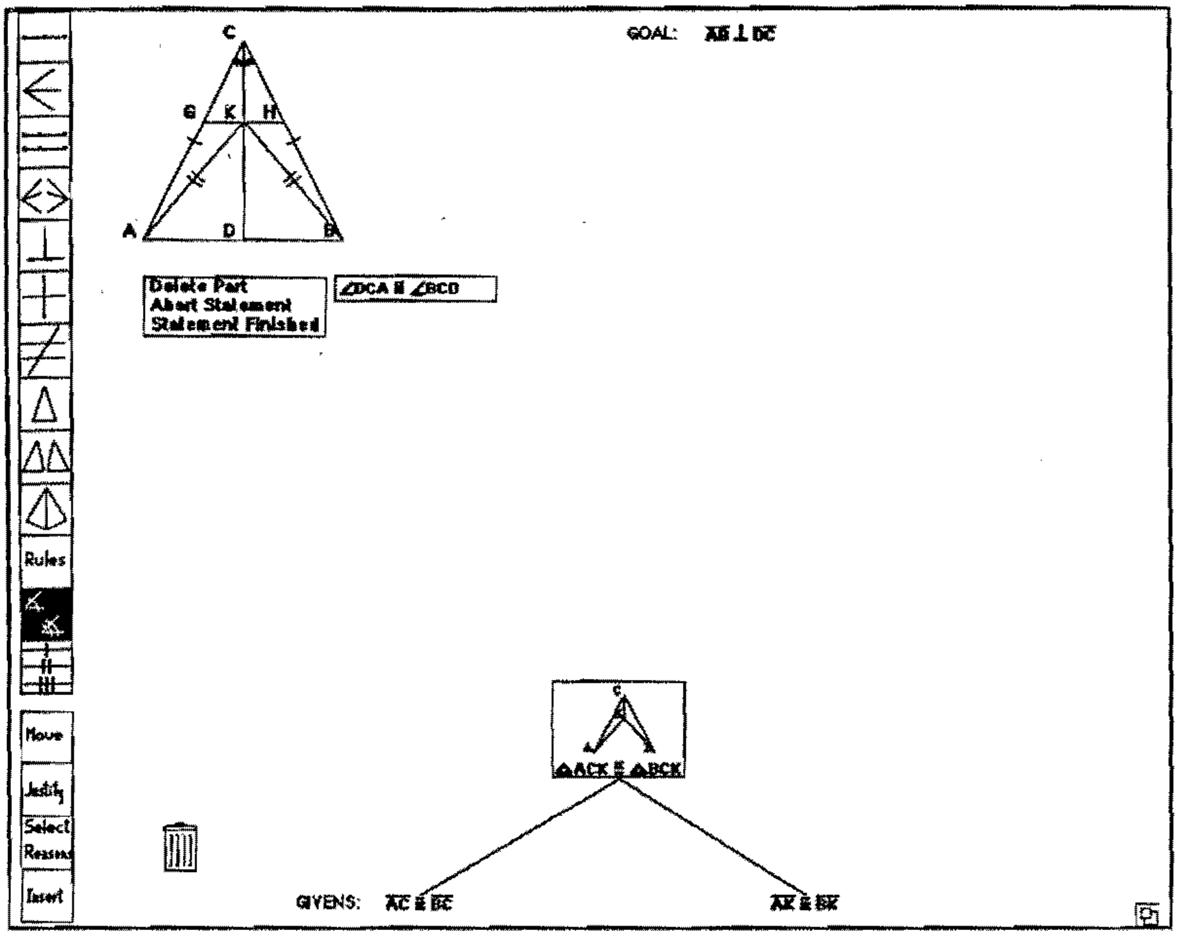

GOAL: $\overline{AB} \perp \overline{DC}$
C
G
K
H
A
D
B
Delete Part
Abort Statement
Statement Finished
∠DCA ≅ ∠BCD
Rules
Move
Justify
Select Reasons
Insert
C
A
B
△ACK ≅ △BCK
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

Figure 2.5. Creating an angle congruence statement. Note that the angles are temporarily highlighted as they are selected – see the markings at vertex C in the diagram.

To finish posting **∠BCD ≅ ∠DCA**, the student selects "Statement Finished" from the menu below the diagram. Now the statement can be dragged into the proof area and justified. Justifying part-statements is exactly the same as justifying schema statements. The student selects Justify from the mode-menu and mouse-clicks on **∠BCD ≅ ∠DCA**. This statement is highlighted, the mode switches to Select Reasons, and the "Done-Abort" menu appears. The student mouse-clicks on **△ACK ≅ △BCK** as the reason and selects "Done Selecting Reasons". The result is shown in Figure 2.6. Notice that as a result of proving **∠BCD ≅ ∠DCA**, these angles are marked congruent in the problem diagram using the conventional hash marks used on paper.

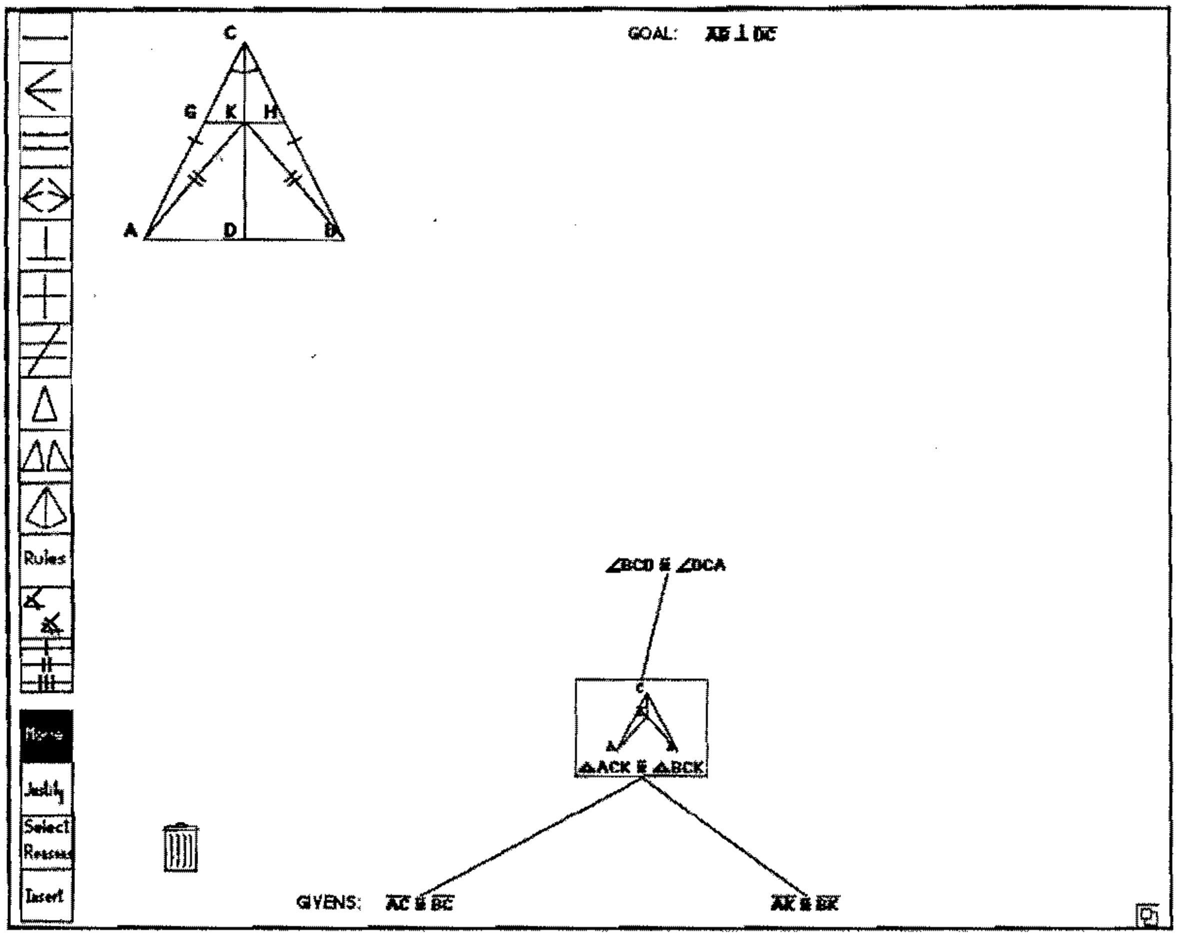

C
G
K
H
A
D
B
GOAL: $\overline{AB} \perp \overline{DC}$
Rules
Move
Justify
Select Reason
Insert
∠BCD ≅ ∠DCA
C
K
A
B
△ACK ≅ △BCK
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

Figure 2.6. Justification of a part-statement. Notice that these angles are marked in the diagram.

Figure 2.7 shows the completion of the proof plan after the addition of schema statement **△ACD ≅ △BCD** and part-statement **∠ADC ≅ ∠BDC**.

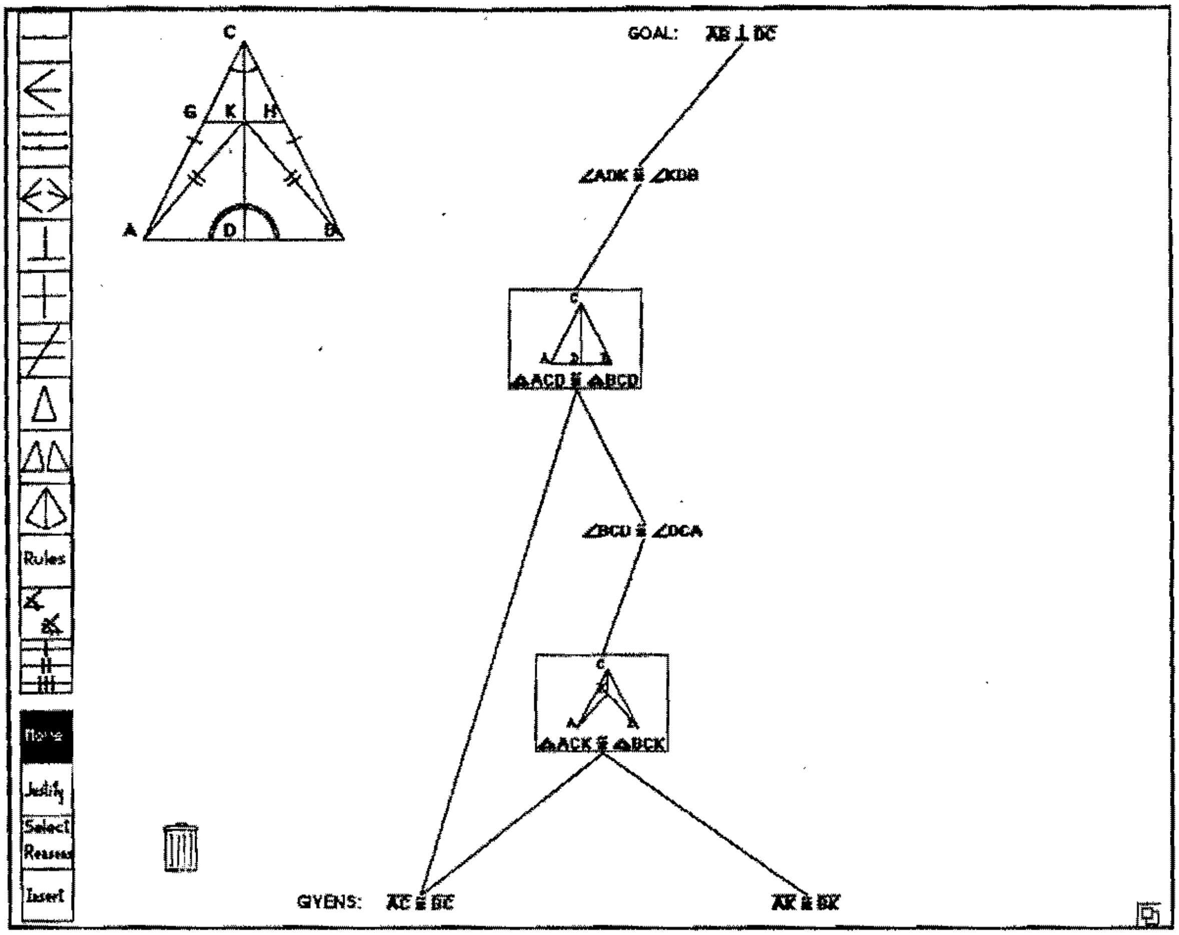


Figure 2.7. A complete proof plan.

2.3.3.2 Execution Actions. At this point it is the task of the student to execute or fill in the details of her proof. Proof execution involves adding any statements that were left out and adding rules to justify the links between statements. To start with a simple and more typical example, I'll illustrate the execution of the second planning step, that is, from **△ACK ≅ △BCK** to **∠BCD ≅ ∠DCA**, and then return to the more involved execution of the first planning step.

Executing the planning step from **△ACK ≅ △BCK** to **∠BCD ≅ ∠DCA** simply involves adding the geometry rule that justifies this inference. This rule is "corresponding parts of congruent triangles are congruent" which we abbreviate **CORRES-PARTS**. First the student needs to create this rule using the "Rules" icon in the icon-menu. Figure 2.8 shows the result of clicking on this icon – a list of rules appears.

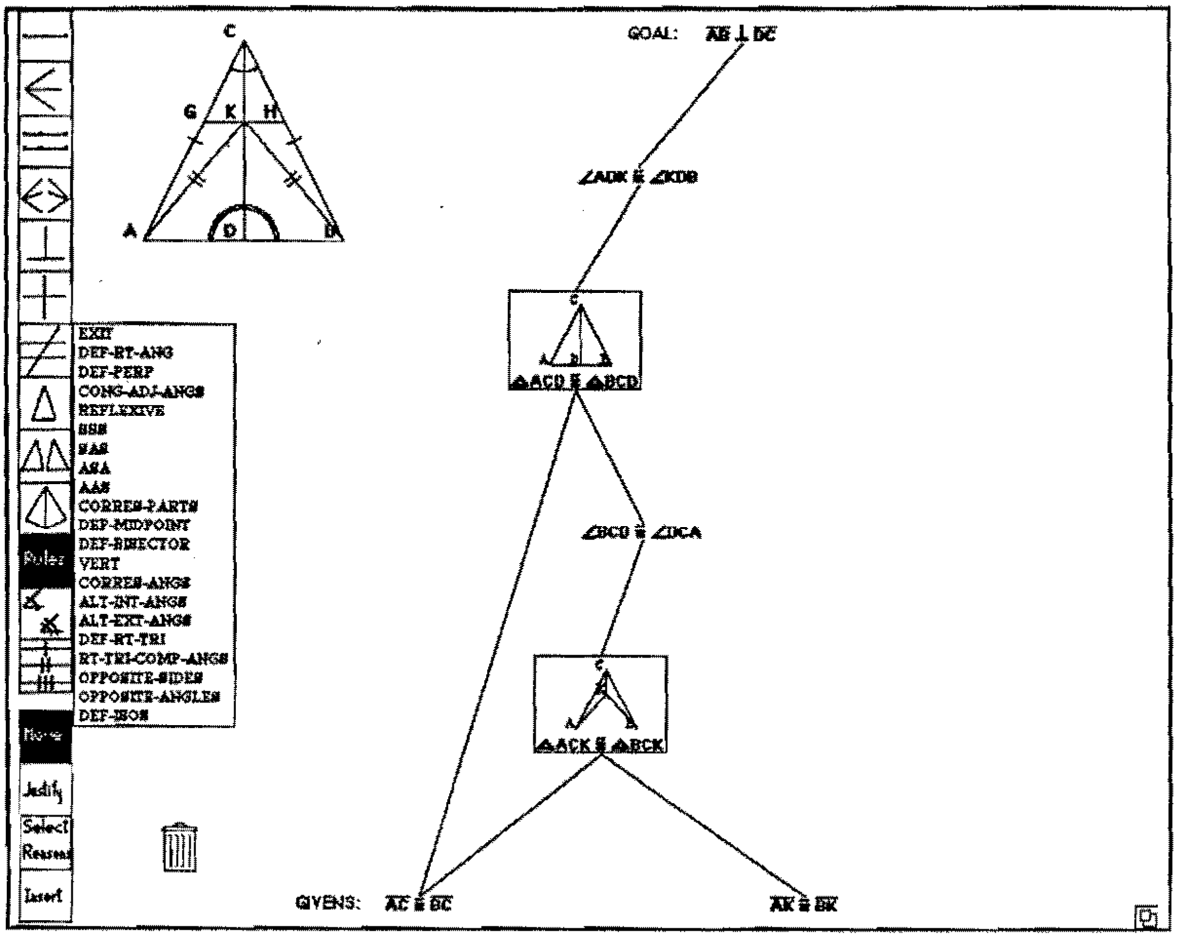

C
G
K
H
A
D
B
GOAL: $\overline{AB} \perp \overline{DC}$
$\angle ADK \cong \angle KDB$
C
A
D
B
$\triangle ACD \cong \triangle BCD$
$\angle BCD \cong \angle DCA$
C
A
B
$\triangle ACK \cong \triangle BCK$
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$
EXIT
DEF-RT-ANG
DEF-PERP
CONG-ADJ-ANGS
REFLEXIVE
SSS
SAS
ASA
AAS
CORRES-PARTS
DEF-MIDPOINT
DEF-BISECTOR
VERT
CORRES-ANGS
ALT-INT-ANGS
ALT-EXT-ANGS
DEF-RT-TRI
RT-TRI-COMP-ANGS
OPPOSITE-SIDES
OPPOSITE-ANGLES
DEF-ISOS
Rules
None
Justify
Select Reason
Insert

Figure 2.8. The rule menu is shown on the left. This menu is used during execution to add rules to justify the links in the proof.

Mouse-clicking on **CORRES-PARTS** causes this rule to appear on the screen. Then the student drags the rule on top of the line connecting **$\triangle$ACK $\cong$ $\triangle$BCK** to **$\angle$BCD $\cong$ $\angle$DCA**. See Figure 2.9.

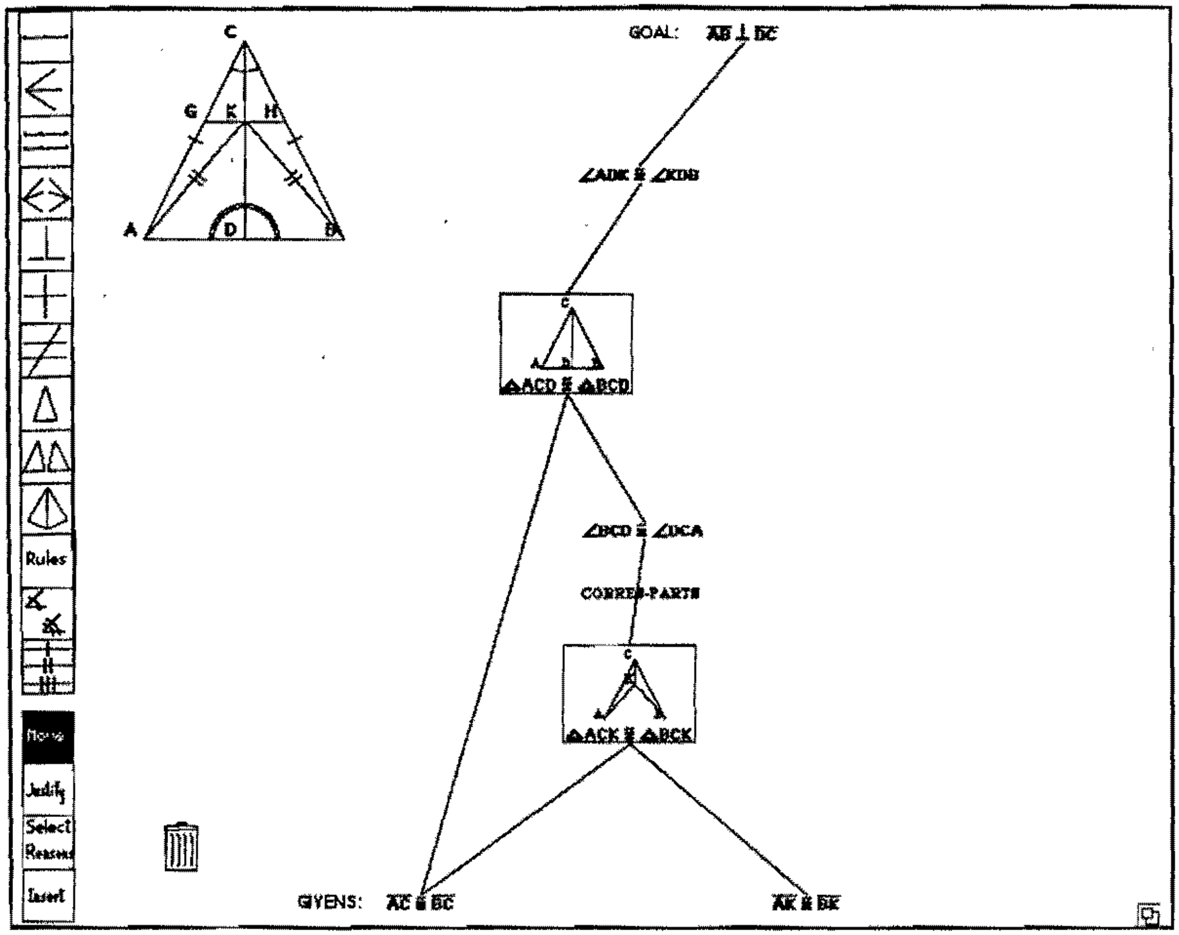

C
G
K
H
A
D
B
GOAL: $\overline{AB} \perp \overline{BC}$
$\angle ADK \cong \angle KDB$
C
A
D
B
$\triangle ACD \cong \triangle BCD$
$\angle BCD \cong \angle DCA$
CORRES-PARTS
C
K
A
B
$\triangle ACK \cong \triangle BCK$
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$
Rules
None
Justify
Select Reasons
Insert

Figure 2.9. A rule is inserted first by placing it over the line in which it is to be inserted.

To insert this rule, the student mouse-clicks on `Insert` in the mode-menu, then mouse-clicks on the rule, and lastly mouse-clicks on "Done Inserting" in the menu underneath the diagram. The rule is inserted by taking the line(s) going into the topmost statement, $\angle BCD \cong \angle DCA$, and having it (them) go into the rule, and then adding a line from the rule to the topmost statement.

To execute the first planning inference, that is, the one from $\overline{AC} \cong \overline{BC}$ and $\overline{AK} \cong \overline{BK}$ to $\triangle ACK \cong \triangle BCK$, the student must add a new statement – the reflexive statement $\overline{CK} \cong \overline{CK}$. This statement is created, just as statements are during planning, and placed somewhere underneath $\triangle ACK \cong \triangle BCK$. Now, rules must be added. Creating the REFLEXIVE rule from the the "Rules" menu, the student drags this rule underneath $\overline{CK} \cong \overline{CK}$. Using `Justify` in the mode-menu, the student creates a link from REFLEXIVE to $\overline{CK} \cong \overline{CK}$. The result is shown in Figure 2.10.

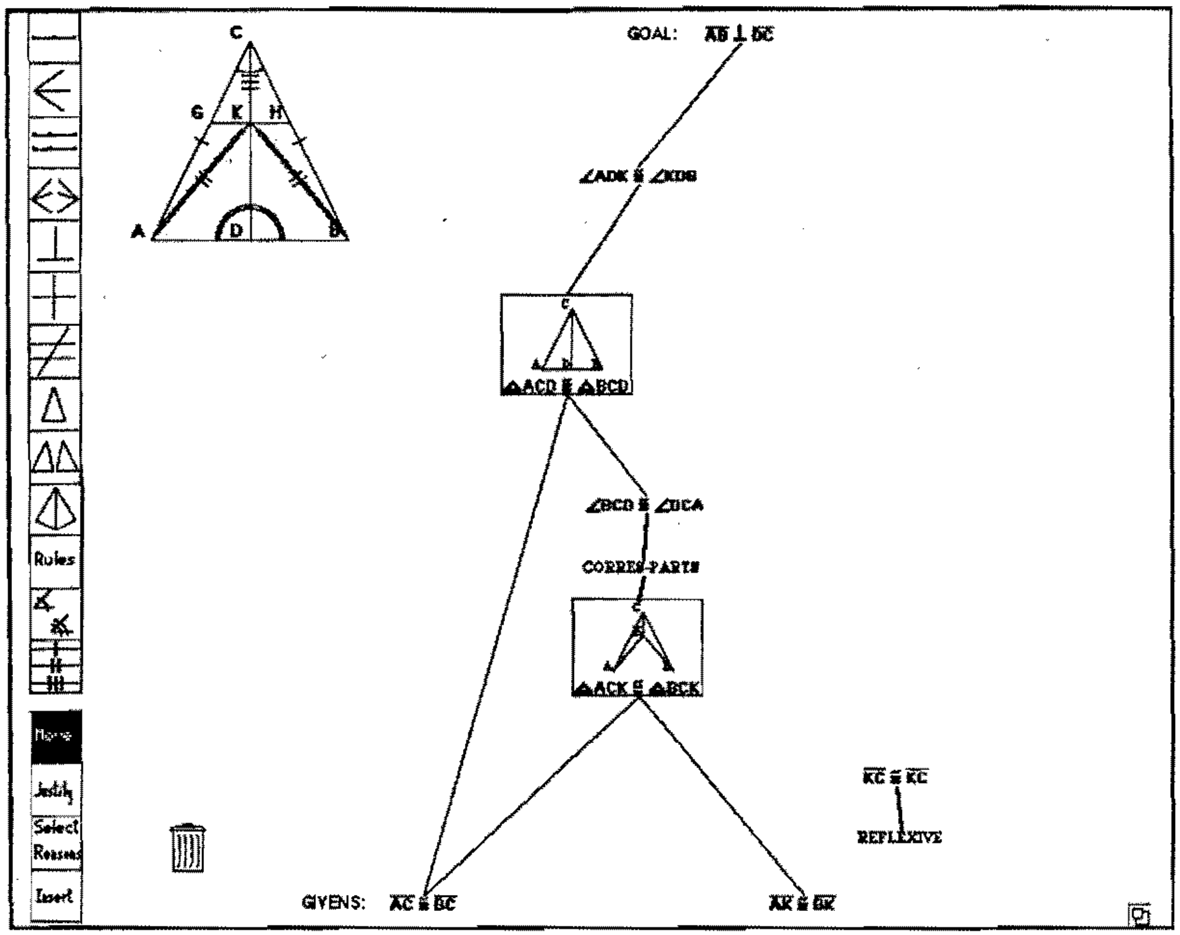

C
G
K
H
A
D
B
GOAL: $\overline{AB} \perp \overline{BC}$
$\angle ADK \cong \angle KDB$
C
A D B
△ACD ≅ △BCD
$\angle BCD \cong \angle DCA$
CORRES-PARTS
C
A B
△ACK ≅ △BCK
$\overline{KC} \cong \overline{KC}$
REFLEXIVE
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$
Rules
Home
Justify
Select Reasons
Insert

Figure 2.10. Adding and executing the reflexive statement $\overline{CK} \cong \overline{CK}$ in service of executing **△ACK ≅ △BCK.**

To complete the execution of this planning inference, the student adds the sss rule and drags it on top of either of the two lines going into **△ACK ≅ △BCK**. She mouse-clicks on `Insert` and then on sss. Figure 2.11 shows this state.

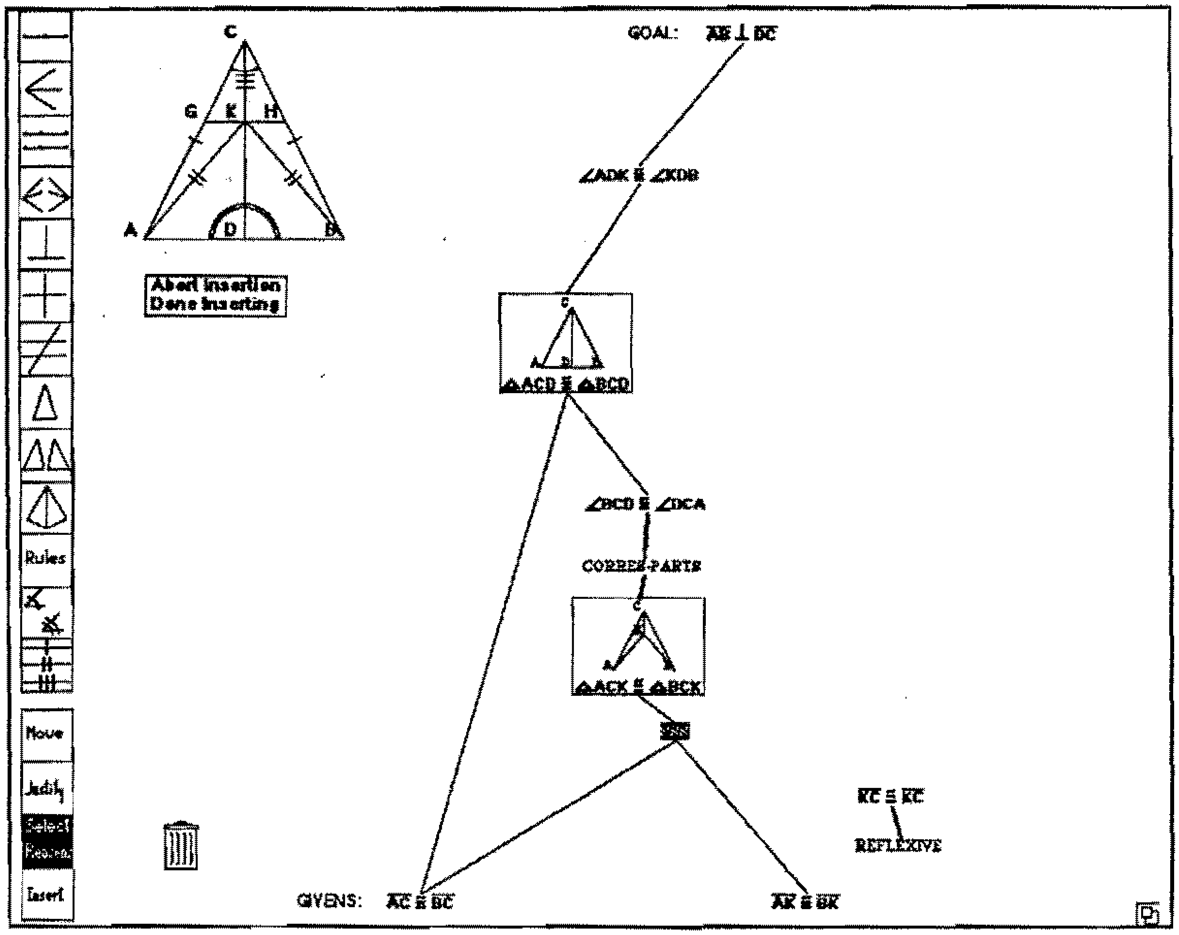

C
G
K
H
A
D
B
Abort insertion
Done inserting
Rules
Move
Justify
Select Reasons
Insert
GOAL: $\overline{AB} \perp \overline{KC}$
∠ADK ≅ ∠KDB
C
A D B
△ACD ≅ △BCD
∠BCD ≅ ∠DCA
CORRES-PARTS
C
K
A B
△ACK ≅ △BCK
SSS
$\overline{KC} \cong \overline{KC}$
REFLEXIVE
GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

Figure 2.11. Inserting a triangle congruence rule – just before adding the extra premise $\overline{CK} \cong \overline{CK}$.

Now, before clicking on "Done Selecting Reasons", the student mouse-clicks on $\overline{CK} \cong \overline{CK}$ to add it as the third premise of SSS. After completing this and executing the remaining planning inferences, the final proof is done as shown in Figure 2.12.

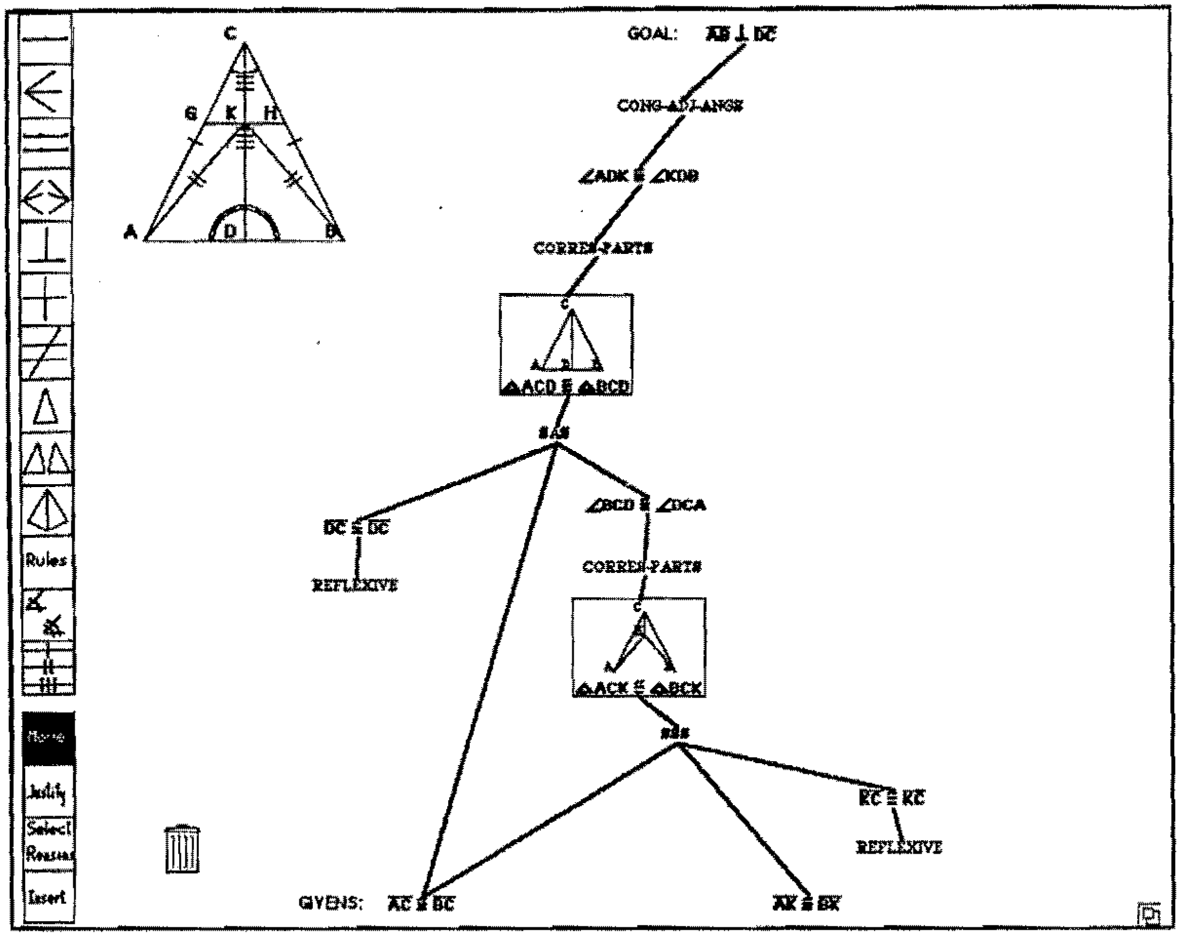


Figure 2.12. The final solution in the execution space.

2.4 THE TUTOR COMPONENT

2.4.1 Motivation for Tutor Component Design

While the interface component was designed to reify model representations with interface notations and reify model processes with interface actions, the tutor component was designed to articulate model representations with the vocabulary used by the tutor and articulate model processes with the explanations given by the tutor.

2.4.1.1 Articulating Model Representations. Schemas were explicitly presented to students in the format shown in Figure 1.3. We decided to call schemas "concepts" as this word is much more familiar to most students. In addition to talking about "concepts", the vocabulary of the schema slot-names, "configuration", "part-statement", and "ways-to-prove" are an explicit part of both the tutor advice and the supplementary text materials.

The content of the feedback ANGLE gives when students make logical errors was also designed to help students learn the schema representation. This logical error feedback first enforces the constraint (1) that only part-statements of a schema can be used to prove it. When this constraint has been satisfied the tutor then checks (2) that the part-statements the student selected match one of the ways-to-prove. In this way, students may learn that when trying to prove a schema, they need not consider any statements besides the part-statements of that schema. Violations of these constraints are called a *wrong-type* error for (1) and either a *too-few* or *too-many* error for (2). These error categories and the corresponding feedback messages are described in more detail below (Section 2.4.3.2).

2.4.1.2 Articulating Model Processes. One straight-forward way of communicating model processes is through the content of the hints the tutor provides. ANGLE's hints for posting a particular schema explicitly direct the student toward diagram parsing (a key process in the cognitive model). The hint to post a CONGRUENT-TRIANGLES schema, for example, encourages careful diagram parsing by asking students to count the triangles in the diagram.

Besides the content of hints, a less obvious way of communicating model processes is through the order in which hints are given. In other words, the order in which hints about appropriate problem solving steps are given should correspond with the order of these steps in the cognitive model. Following through on this design guideline results in one of the biggest differences between ANGLE and the first generation GPT.

Consider the situation in Figure 2.13. If a student were to ask for help from GPT at this point, it would focus on using the corresponding-parts rule to make the inference from $\triangle BCD \cong \triangle YZW$. The message would be something like "Notice two triangles which are congruent . Think about what this tells you." This is not necessarily a bad hint. But, it lacks a strategic context that might help the student decide which of the six corresponding parts it might be useful to prove. We've observed students blindly following this advice, proving non-essential parts congruent (e.g., $\angle CBD \cong \angle ZYW$), getting the advice again, proving another set of non-essential parts congruent (e.g., $\angle CDB \cong \angle ZWY$), and so on. Eventually, they stumble upon the right corresponding parts, in this problem both $\overline{DC} \cong \overline{WZ}$ and $\angle DCA \cong \angle WZX$ are useful, and only then will GPT start giving advice towards proving $\triangle ACD \cong \triangle XZW$.

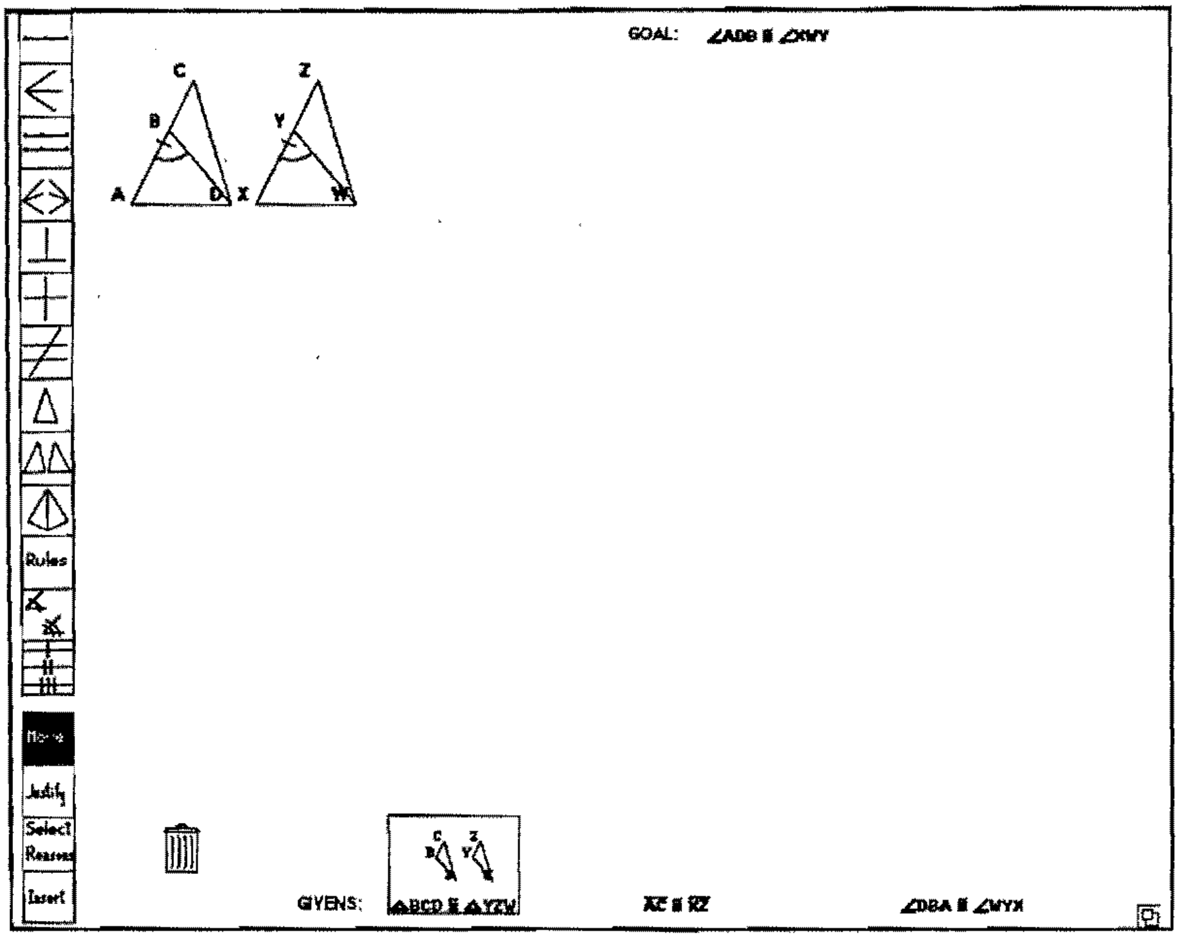


Figure 2.13. The context for a strategic hint. The next hint will focus on the selection of the △ACD ≅ △XZW schema and only then, will the hints focus on part-statements that follow from △BCD ≅ △YZW. These hints suggest to select those part-statements which will help to prove △ACD ≅ △XZW.

ANGLE approaches this situation in more or less the opposite direction, that is, from △ACD ≅ △XZW to the part-statements $\overline{DC} \cong \overline{WZ}$ and ∠DCA ≅ ∠WZX. If the student needs help, ANGLE would first provide *schema selection hints* (described in Section 2.4.4.1) to encourage her to look in the diagram for triangles that she might prove congruent. The student is advised toward posting △ACD ≅ △XZW as an island subgoal. See Figure 2.14.

Next, she should work on finding part-statements which she can prove from △BCD ≅ △YZW and which are relevant to this goal. If she has trouble, ANGLE provides *part-statement justification hints* (described in Section 2.4.4.3). For example, the most general such hint would suggest in this situation: "Look for OVERLAPPING concepts. That is, look for a part-statement which appears both in △ACD ≅ △XZW and in a concept you've already proven.".

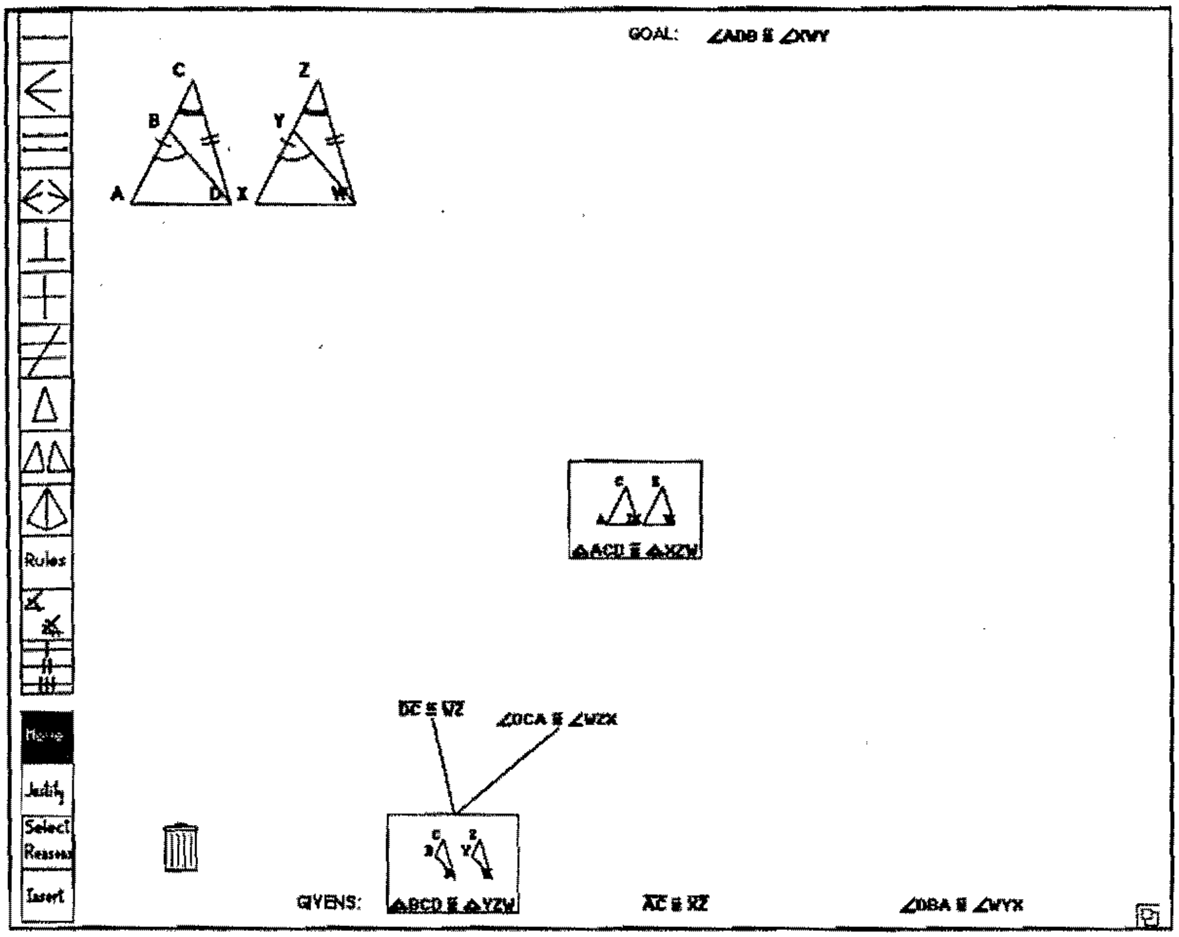


Figure 2.14. The needed part-statements have been added. The next hint will focus on justifying △ACD ≅ △XZW using the proven part-statements.

Figure 2.14 shows the addition of the necessary part-statements. If the student has trouble here, ANGLE would provide a *schema justification hint* (described in Section 2.4.4.2) that would work her toward finding the appropriate premises ($\overline{DC} \cong \overline{WZ}$, ∠DCA ≅ ∠WZX, and the given $\overline{AC} \cong \overline{XZ}$) with which to justify △ACD ≅ △XZW.

2.4.2 Tutor Description

ANGLE's tutor component provides two types of advice: 1) *feedback* on student's logical errors, and 2) strategic *hints* for what to do next in the case the student is stuck or appears to be floundering. The tutor component calls upon the expert to provide the basic information from which to generate advice. To help generate feedback, the expert is used to check the correctness of student actions. If the action is incorrect, the error is categorized and the tutor gives feedback with respect to the error category. To help generate hints, the expert is used to find a good *next-step* that the student could take in the context of their current solution. The tutor provides advice by hinting at this next-step.

Within a particular context, the advice always starts out general, so as to maximize the cognitive involvement of the student, and becomes increasingly more specific if the student continues to make errors within this context. If the student makes enough errors, at some point the advice bottoms-out by telling the student what they should do next.

Different feedback and hints are required for the three major kinds of action in ANGLE: statement selection (diagram parsing), statement justification, and execution-justification.

2.4.3 Logical Feedback

2.4.3.1 Feedback for Statement Selection. As mentioned above, feedback is generated in response to student errors. A statement selection action is in error if the statement is clearly not a consequence of the problem givens. Given a properly drawn diagram, that is, one in which the problem givens are true, a student can tell whether a statement is plausible just as DC does by seeing if it looks true in the problem diagram. If a student selects a part-statement which is implausible, for example, $\angle GAK \cong \angle KAD$ in the problem in Figure 2.12 above, the tutor provides feedback: "If the diagram is drawn accurately, angles which don't look equal cannot be proven equal."

When a student selects a schema-statement, it is checked for types of errors. The more serious error is the selection of a schema-type which does not appear in the diagram. In this case, ANGLE responds "This concept does not appear in the diagram." A less serious error occurs when the student selects a schema-type which appears in the diagram, but the lines he picks does not constitute an instance of this schema.The tutor responds to this error with "The lines you picked do not form this concept".

Occasionally a student might select lines which are instance of a schema other than the schema he selected. ANGLE does not currently recognize such a bug, but it might be useful if it did, so that the tutor could give more reasonable feedback, e.g., "It looks like you are trying to make $\triangle ACK \cong \triangle BCK$. Since these triangles share a side, you should select the CONGRUENT-TRIANGLE-SHARED-SIDE concept, not the CONGRUENT-TRIANGLE concept."

2.4.3.2 Feedback for Statement Plan-Justification. Consider problem 7 shown in Figures 2.1 through 2.12 above. Imagine a student selected $\triangle ACD \cong \triangle BCD$ first and tried to justify it with the two givens. This is clearly wrong since $\overline{AK} \cong \overline{BK}$ are not corresponding parts of these triangles. The tutor responds first with a generic message, "The premises you chose are not a legal justification of this concept. Review the TRIANGLE CONGRUENCE SHARED SIDE concept". If the student were to make this error a second time the tutor would respond with more specific feedback about the error: "The premise $\overline{AK} \cong \overline{BK}$ is not a part-statement of the concept you are trying to prove".

This example illustrates the two levels of logical feedback. The generic feedback message always appears in response to the first logical error a student makes when attempting to justifying a particular statement. The response to the second and subsequent logical errors depends on the type of the error. Logical errors are categorized into three types, ordered in terms of severity: 1) *wrong-type*: one of the premises is of the wrong type, e.g., not a part-statement of the schema being justified, 2) *too-few*: all the premises are of the right type, but there are not enough of them to satisfy a way-to-prove, and 3) *too-many*: they're the right type and there's enough, but there are extra unnecessary ones as well. The tutor generates a message to address the most severe error the student has made. The example above was a wrong-type feedback message. Too-few feedback says "The statements you selected are part-statements of <a schema statement> but they do not match any of the ways-to-prove", while too-many feedback says "You chose more premises than you need.

2.4.3.3 Feedback for Execution-Justification. This feedback is quite simple in this version of ANGLE. When a student makes an error in attempting to add a rule, the tutor responds by indicating the what type of statements the premises should be. For example, if a student makes a mistake on a triangle rule, the tutor responds "The premises of <a triangle rule, like SAS> should be 3 segment or angle congruence statements which are corresponding parts of congruent triangles."

2.4.3.4 Miscellaneous Feedback Situations. It is possible to create a loop in ANGLE's proof graph by indirectly using a statement to justify itself. ANGLE detects this and responds "You are trying to use <some statement> to prove itself. That line of reasoning is circular."

Other special case message situations include (1) trying to justify a given and (2) trying to use the goal as a premise. In the former case, the tutor responds "It doesn't make sense to justify a given. Givens are already proven by definition." In the latter case, "It doesn't make sense to use the goal as a reason. The goal is what you're try to prove."

2.4.4 Strategic Hints

Strategic hints are given under two circumstances. Either because students request a hint using the `Info` menu at the top of the screen or because they appear to be floundering. In a similar fashion to the Geometry Proof Tutor, students are judged to be foundering if they commit three logical errors within a particular *hinting context.* (In GPT, strategic hints start sooner – after *two* logical errors.) Each hinting context starts either at the beginning of the problem or after a successful justification action.

The first time a student gets a hint within a hinting context, either because he's requested one or because he's made three logical errors, it is the most general hint associated the with a good next move as determined by the expert component (as described above). Until the student successfully justifies some statement and as long as they continue to make more logical errors or request hints, they will get ever more specific hints which bottom-out by telling the student what he should do. We call this last type of hint a *bottom-out hint.* Typically, there are only one or two less general hints between the most general hint and the bottom-out hint. Thus, if the student is not asking for help it will take three logical errors before they get their first general hint and two or three more errors before they get the bottom-out hint – a total of five to six errors. Alternatively, they will get the bottom-out hint if they are not making errors but request a hint three or four times within a hinting context.

Planning hints are associated with schemas and are retrieved from the particular schema that the expert suggests as a good next move. The most general schema hints are applicable to all or most schema classes, while the more specific hints may be only associated with a particular schema. Planning hints for a particular schema statement are selected from one of three categories depending on the situation: (1) a *schema selection hint* is given if the student has not yet posted this schema statement, (2) a *schema justification hint* is given if the schema has been posted and the part-statements necessary to prove it have been posted *and* proven, or (3) a *part-statement justification hint* is given if the schema has been posted but the part-statements necessary to prove it either have not been posted or have not been proven.

(Examples of these hint categories are given below.) Recall that because of the way the suggested next schema is generated, in case (3) the needed part-statements will be necessarily provable from already proven schemas.

Planning hints are given until planning is complete. Then execution hints are given. In other words, the cognitive model suggests that a complete plan should be found before potentially wasting time executing something that will not contribute to the final proof.

2.4.4.1 Hints for Schema Selection. The most general schema selection hint, "Look in the diagram and see if you can find something that looks like one of the concepts you've learned", appears for any schema except the triangle-related schemas. In the case of triangle schemas, the general statement selection hint is "Sometimes triangle concepts are hard too see. If you can't find all <N> triangles in this diagram, you may not have noticed some that are useful." where N is the number of triangles in the diagram. This hint is intended to encourage students to more carefully parse the problem diagram in situations where they are apparently not seeing an appropriate triangle congruence inference. The more specific hints essentially indicate the type of schema the student should be looking for, for example, "Find lines in the diagram which look perpendicular. Try to prove that they are." The schema selection bottom-out hint tells them which particular instance to pick, for example, "Pick the CONGRUENT TRIANGLES SHARED SIDE concept and indicate ▲ACD and ▲BCD in the diagram."

2.4.4.2 Hints for Schema Justification. The most general schema justification hint attempts to focus students on looking for any proven part-statements of the schema they are trying to prove, "Find proven part-statements of <the desired schema> and use them to justify it." Essentially, this is a generalization of the indirect subgoaling that Greeno identified (see Section 1.5.4.2) in that it applies to any schema not just triangle congruence schemas. The bottom-out hint is "Justify <the desired schema> using statements: <the necessary statements>."

2.4.4.3 Hints for Part-Statement Justification. Schema justification hints are given when the expert's suggested schema can be proven directly with part-statements that are already posted and proven in the proof graph. In the case that one of these part-statements has not been proven or posted, a part-statement justification hint is given. The most general hint is "Look for OVERLAPPING concepts. That is, look for a part-statement which appears both in <desired schema> and in a concept you've already proven.". The less-general hint points out which part-statement this is and the bottom-out hint explicitly states how it should be proven: "Justify <needed part-statement> using concept <already proven schema>."

2.4.4.4 Hints for Execution-Justification. The execution hinting scheme is quite simple in this version of ANGLE. The most general hint, makes sure the student knows he is executing and no longer planning: "Add rules to indicate the reason for each statement or concept. All concepts and statements, except the givens, should have thick lines going to them". The bottom-out hint is "Prove <statement> using the <rule> rule."

2.5 WHAT'S ON THE HORIZON FOR ANGLE?

We have two good milestones of tutoring success to which ANGLE can be compared. The first is GPT which, in a field test in a Pittsburgh high school, led to about one grade level or one standard deviation improvement in students who had the tutor in their classroom over students in a normal classroom. The second milestone is a generic one for all computer tutors. Bloom (1984) showed that human tutors can improve student performance by about 2 standard deviations over students in a normal classroom.

Why might ANGLE be better than GPT? We can think of a tutor as a model of problem solving which students can emulate. To the extent that the tutor's problem solving method is a good one and students successfully emulate it, then they will be good problem solvers as well. Because DC is a more powerful method than the one in the previous tutor, we believe that students who successfully emulate it will be even better problem solvers than those who successfully emulated the problem solving method taught by GPT.

Besides being a more powerful problem solving method than one that focuses on single steps, our basic research suggests that the DC method may be easier to learn. Koedinger and Anderson (1989) argue that DC-schema's are not learned, in any direct way, from the formal rules of the domain – what we call the execution space. Instead, they appear to be learned by identifying useful categories of domain objects (i.e., the configuration) and learning their properties (i.e., the part-statements and ways-to-prove). These categories carry the load of recognition: indicating *when* particular knowledge should be brought to bear and having the effect of drastically reducing the search space. The properties carry the load of inferencing – indicating what can be concluded (part-statements) and under what conditions (ways-to-prove).

We believe this type of learning, that is, abstraction from domain objects, is a more natural extension of students' prior knowledge. Thus, contrary to the concern that DC represents an expert method that will be too difficult for students to understand, we believe the DC method may be easier to learn. The instruction in GPT is focussed on the formal rules of geometry which are totally new and unfamiliar to students. In contrast, the focus of instruction in ANGLE would be on diagram configurations and their properties. Pre-geometry students already have some prior perceptual intuitions about geometric figures. Diagram configuration schemas can be taught by building off this familiar ground.

CHAPTER 3. INITIAL EVALUATION OF ANGLE

3.1 INTRODUCTION

This chapter reports on our first empirical study of ANGLE. The central goal is to test the hypothesis that the development of more accurate and powerful cognitive models of problem solving can lead to major improvements in the instruction of problem solving, particularly within the context of intelligent tutoring systems. In this study, we compared GPT and ANGLE – two tutors for the same domain whose primary difference is the cognitive model that underlies them. While the cognitive model underlying GPT is a reasonably good one, it has some flaws. It does not do a good job capturing the more strategic abilities of effective problem solvers. The DC model improves on GPT's model by providing a sound explanation of these abilities. The question is: can this improvement lead to more effective instruction?

In addition to this big goal, this study allows us to (1) check the feasibility of teaching proof planning separate from execution, (2) gather further data on the roles of conceptual, perceptual, and formal rule knowledge in a formal reasoning domain, and (3) gather data with which to tune ANGLE's interface and tutor messages.

ANGLE is still in an early development stage. In particular, the current implementation of the feedback scheme and the wording of the feedback messages are only a reasonable first pass attempt. It takes iterations of usage and redesign to get a good sense for how to make them most effective. As mentioned in Chapter 2, ANGLE represents about 1.5 person-years of effort. This study was its first extensive trial. In contrast, GPT represents about 10 person-years of effort and has been through at least 2 major iterations of study and redesign.

3.2 METHOD

3.2.1 Subjects

The subjects in this study were high school students recruited from the Pittsburgh Public School district. To reduce the amount of declarative instruction in geometry and focus on proof problem solving, we required these students to have just completed a high school geometry course. 30 students participated for pay: $54 total for completing the experiment. Subjects were told they would receive $2 per hour plus a completion bonus of $20 and a performance bonus of $10. The performance bonus was intended to encourage subjects to take the task seriously. All subjects were actually paid the whole amount.

3.2.2 Materials

The following materials were used in this study. The handouts and tests appear in Appendix A.

Handouts: Rule Summary Sheet, Tracking Sheet, Concept Summary Sheet (ANGLE group only), ANGLE Text (ANGLE group only), GPT Text (GPT group only).

Tests: Proof Construction A, Hidden Figures A, Truth Judgment A, Proof Checking A, Proof Construction B, Hidden Figures B, Truth Judgment B, and Proof Checking B.

Tutors: ANGLE written in Macintosh Allegro Common LISP running on a Mac IIci, GPT written INTERLISP-D running on a Xerox 1109.

3.2.3 Design

The design of the study is straight-forward. Half the subjects used ANGLE while the other half used the Geometry Proof Tutor (GPT). There were two versions of each test, the A version and the B version, designed to be equally difficult. Half the subjects in each group took the A version tests as pre-tests and the B versions as post-tests, while the other half did the opposite: B versions as pre-tests and A versions as post-tests.

Here is the design, with number of subjects in each cell:

	Order:		
Tutor:	A->B	B->A	
ANGLE	8	7	15
GPT	7	8	15
	15	15	30

3.2.4 Procedure

Subjects participated in six two-hour sessions over the period of two weeks. The first session was devoted to pre-testing and if time permitted, an introduction to the one of the two tutors depending on which condition the subject was in. During the next four days, the subject would work on the tutor until they had spend eight full hours. On the sixth day, subjects would take the post-test.

PRE-TESTING ==> 8 HRS OF TUTORING ==> POST-TESTING

During both the pre-test and post-test, subjects were given the Rule Summary Sheet for the Proof Construction and Proof Checking tests. For the proof construction test they were told that their proofs must not contain any geometry rules other than the eight appearing on the Rule Summary Sheet. They were given 35 minutes for the Proof Construction test, 6 minutes for each of the two parts of the Hidden Figures test, 15 minutes for each of the Truth Judgment and Proof Checking tests. The eight test handouts appear in Appendix A.

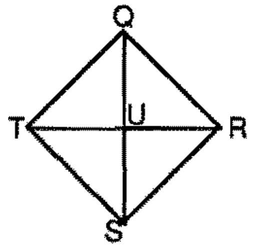
3.2.5 Motivation for Tests

The Proof Construction test was the main dependent measure. We wanted to know how the tutors effected the acquisition of proof construction skill. The other tests were used to try to capture sub-components of proof construction skill.

The Hidden Figures test is a variation of the Embedded Figures Test (Witkin et. al., 1971). The test involves finding one of set of figures within a complicated line drawing. This type of perceptual disembedding is similar to the kind of disembedding geometry problem solvers have to do, for example, to find two congruent triangles in a complicated geometry diagram.

The Truth Judgment and Proof Checking tests were an attempt to decouple the planning and execution portions of geometry proof skill. The Truth Judgment test was designed to capture planning skill, but not require execution. Here's an example problem:

9a. If $\angle TQU \cong \angle RQU$ and $\angle QTU \cong \angle QRU$, must $\angle STU \cong \angle SRU$?



YES

CAN'T TELL

The student was asked to circle either YES or CAN'T TELL. One way to approach these problems is to attempt to construct a proof plan of the conclusion: $\angle STU \cong \angle SRU$ in the example above. If a plan is found, the answer is YES. Note, it is not necessary to fill in the details of this plan in order to correctly answer these problems — thus, execution skills are not required.

A second motivation for the Truth Judgment test was an attempt to identify biases in geometric reasoning in the spirit of the biases which have been found in syllogistic reasoning (Johnson-Laird, 1983). In particular, the test included four types of problems varying on whether the correct answer was YES or CAN'T TELL and whether the problem diagram was "good", meaning the givens and goal of the problem looked true, or "bad", meaning the givens and goal did not look true:

	Diagram:	
Answer	Good	Bad
YES	a	b
CAN'T TELL	c	d

Problem types b and d differed from matched items a and c only in that the diagram was distorted. Problem types c and d had the same diagram as a and b, but the point labels were different and the problem givens and goal were different. Problems 1a, 1d, 2b, 2c, etc., appeared on Truth Judgment test A, while problems 1b, 1c, 2a, 2d, etc., appeared on Truth Judgment test B.

In contrast to the Truth Judgment test, the Proof Checking test focuses on execution skills but does not require planning skills. The problems contain two-column proof solutions with errors in them. The task for the student is to identify the errors. These solutions are essentially correct — they contain all the elements of a complete proof plan. However, they contain two types of execution errors: 1) skipped execution steps or 2) a correct step justified with the wrong rule.

3.2.6 Tutoring Details: Slowing Down ANGLE

In both groups students worked with a computer tutor for 8 hours. Each student read through their respective text handout. Both texts start by illustrating how to solve problem PROB150 using the tutor — indicating precisely what mouse-clicking and keyboard actions are required. The remainder of the texts alternatively review the needed geometry content and then indicate the next problem the student should work on with the tutor. To keep track of their progress students were asked to use the Tracking Sheet to check off the problems as they completed them.

The tutors were used as designed with one exception. We tried to compensate for the fact that loading a problem on GPT takes significantly longer (a minute or two) than loading a problem on ANGLE (about 10 seconds). Even though this speed-up is partially due to the greater efficiency of DC over GPT's expert (the other part is the Mac II hardware is faster than the Xerox 1109 hardware), we didn't want to get an effect simply because students were able to spend more time solving problems with ANGLE. Thus, we "brain-damaged" ANGLE by putting a pause in the problem loading procedure such that loading a problem on ANGLE would take about as long as loading it on GPT.

3.2.7 Curriculum

Given the relatively short training time (8 hours), we focussed on a limited portion of the curriculum, in particular, the topics of perpendicularity and triangle congruence. Traditional geometry textbooks are organized around stating, explaining and giving examples of geometry postulates, definitions, and theorems. All three of these can be stated conveniently as if-then or if-and-only-if rules and we refer to them collectively as *rules* or *logical rules* to distinguish them from production rules. The curriculum for this experiment included the 8 rules: DEFINITION-OF-PERPENDICULARITY, CONGRUENT-ADJACENT-ANGLES, CORRESPONDING-PARTS, SSS, SAS, ASA, AAS, and REFLEXIVE. These rules are abbreviated and defined in the Rule Summary Sheet which was given to all students and appears in Appendix A.

Four DC schemas cover the same territory as these eight rules. These schemas are: PERPENDICULAR-ADJACENT-ANGLES, PERPENDICULAR-CROSS, TRIANGLE-CONGRUENCE, and TRIANGLE-CONGRUENCE-SHARED-SIDE and they are shown in the Concept Summary Sheet (Appendix A).

The 30 problems used by both groups during tutoring appear on pages 3-5 of the Tracking Sheet (Appendix A). Subjects were allowed to refer to them during tutoring.

3.2.8 Test Grading

Proof Construction. We graded the problems on the Proof Construction in two different ways: (1) an overall planning and execution measure and (2) a planning only measure. The overall measure gives one point for each student step which is correct and a part of a solution. Note, that in the case of multiple solutions to a problem, the maximum number of points for that problem is still the number of steps in the shortest solution. To prevent a student from getting more points for pursuing a longer solution, such solutions were scored by subtracting one from the maximum score for every missing step. Students lost half a point for correct steps which were justified by the wrong rule. The maximum score on this measure was 32 for Proof Construction test A and 31 for Proof Construction test B.

For the planning measure, one point was given for each correct planning step that the subject made. The maximum score on this measure is 12 for both version A and B. Correct planning steps correspond with the diagram configuration schemas in a correct solution. Subjects were credited for a planning step if:

a) one of the written steps is the whole-statement of the corresponding schema (and in the right position of the proof), or

b) one of the written steps is a part-statement which is unique to the schema, or

c) one of the written steps is a part-statement of the schema and is justified by a rule which is unique to that schema, or

d) the diagram is marked so as to uniquely indicate the schema.

Other statistics were collected including: 1) the number of illegal moves, 2) the number of legal but off solution path moves, 3) the number of "rule errors" defined as steps which had the right statement, but were justified by the wrong rule, and 4) the number of "skipped steps" defined as the number of execution steps left out in the context of a correct planning step.

Truth Judgment. Subjects received one point for each correct item. There were 17 total points on both tests.

Proof Checking. Subjects received 1 point for each incorrect step correctly identified as "doesn't follow" plus 2 more points if they correctly explained what was wrong with the step. They received 1 point for each correct step identified as "OK". Both test versions had 8 incorrect steps and 8 correct steps, so that the maximum score was 32 points (i.e., 8x3 + 8x1).

Hidden Figures. Subjects received 5 points for each correct item and lost 1 point for each incorrect item. With 16 items on each test version, scores could range from -16 to 80.

3.3 RESULTS AND DISCUSSION

3.3.1 Overall Results of the Pre-Tests and Post-Tests

Figure 3.1 shows the percent correct on all four *pre-tests* for both tutor groups. The trends in favor of GPT on all four tests would suggest a slight advantage for the GPT students. These differences are not statistically significant, $p > .6$, $p > .19$, $p > .9$, and $p > .4$ respectively for the Proof Construction, Truth Judgment, Proof Checking, and Hidden Figures pre-tests.

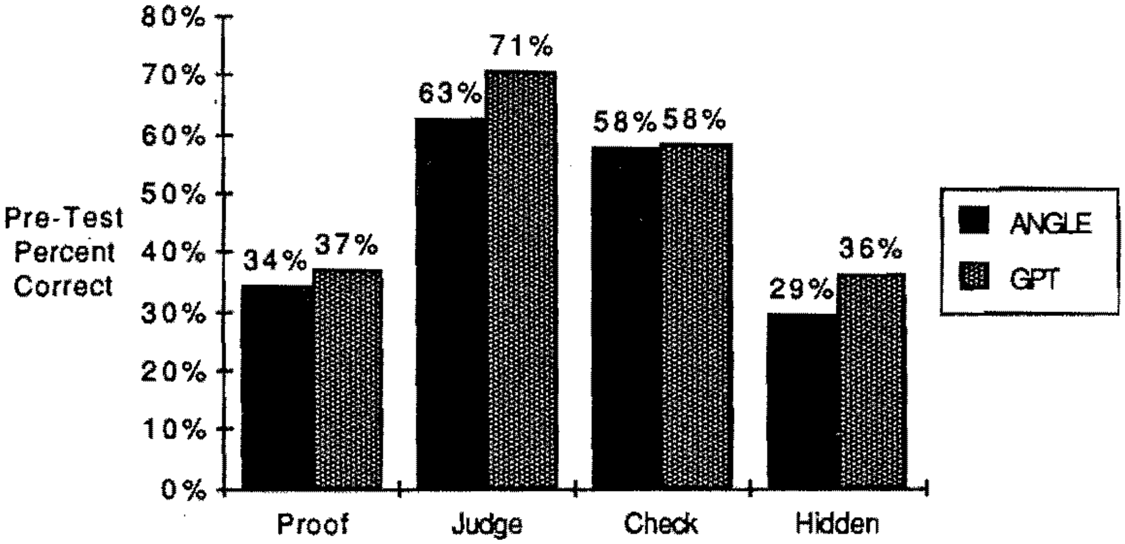


Figure 3.1. Percent correct on pre-tests.

Figure 3.2 shows the percent correct on all four *post-tests* for both tutors. We see essentially the same trends in favor of GPT as we saw in the pre-tests. But again, none of these differences are significant, p > .13, p > .28, p > .33, and p > .71 respectively. The difference on the Proof Construction test is somewhat large and below we discuss what might be going on here. However, the overall point is that in the first major test of ANGLE we have achieved a level of tutoring equivalent to GPT. In addition, we have good reason to believe that the current version of ANGLE has particular weaknesses that can be remedied. We provide evidence for these limitations below and discuss possible remedies.

It is important to note that students did show significant learning in both groups. Comparing subjects' pre-test scores to their post-test scores, across both groups, we find that subjects show significant learning on all but the Truth Judgment test (p-values, .0001, .09, .0001, and .02 respectively). If subjects had not learned, one might question the adequacy of the experimental procedure, for example, was there enough training time? Was the curriculum adequate? Was the subject population appropriate? However, consistent with the established success of GPT, subjects learned quite a lot. On the proof test, subjects went from about one third correct on average on the pre-test to more than two-thirds correct on the post-test[1]. These results provide evidence that ANGLE is an effective tutor, though we clearly can't say that is more effective than GPT.

[1] Note, to avoid ceiling effects, these tests were designed to be difficult. The Proof Construction tests, for example, contained items that are considerably more difficult than the ones typically encountered in a high school class.

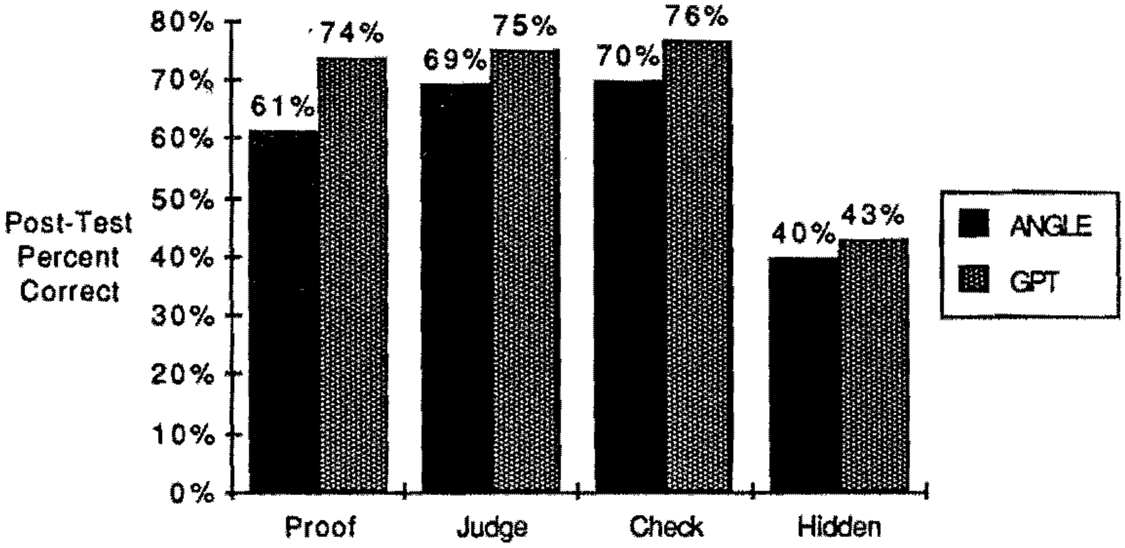
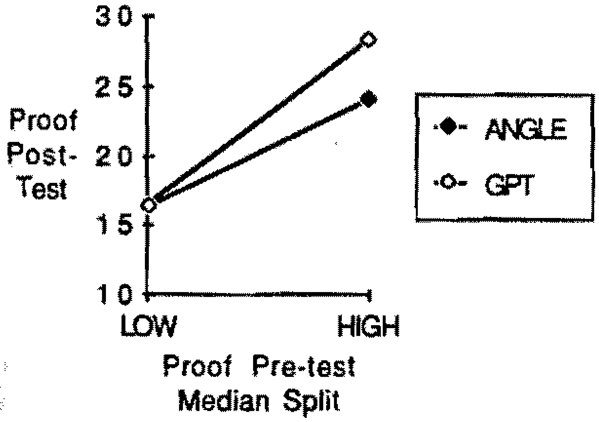


Figure 3.2. Percent correct on the post-tests. None of the differences are significant, p > .13, p > .28, p > .33, and p > .71, respectively.

Adjusting for Pre-Test Differences. We can try to adjust for the pre-test differences in a couple of ways. First, we can analyze subjects' pre- to post-test improvement in terms of the difference between the two scores. Comparing the improvements of the two groups, we get p-values of .13, .71, .23, and .61 respectively – again suggesting no significant difference between the tutors. Second, given the high correlation between pre-test and post-test scores (.66, .33, .64, and .64, respectively), we can do an analysis of covariance using the pre-test score as a covariate. Here we get p-values of .55, .15, .42, and .62 respectively for the four tests.



	Pre-test Split:		
Tutor:	LOW	HI	
ANGLE	16.5 N=9	24.2 6	19.6 15
GPT	16.6 6	28.4 9	23.7 15
	16.5 15	26.7 15	21.6 30

Figure 3.3. Hint of possible aptitude-treatment interaction on the Proof Construction test. It is not significant, p > .3. There is, however, a clear effect of aptitude, p < .001. The table on the right shows the post-test means and number of subjects in each cell.

There was some hint of an aptitude-by-treatment interaction as is shown in Figure 3.3. Subjects were split into HIGHs, the 15 highest scoring subjects on the proof pre-test, and LOWs, the 15 lowest scoring subjects. A two-way ANOVA yields a significant effect of aptitude ($p < .001$), but no significant effect of tutor ($p = .31$) and no significant interaction ($p=.33$).

There is virtually no difference between the tutors in the LOWs. Notice that to the extent there is a difference between the tutors it appears only in the HIGHs. In other words, perhaps the HIGHs are learning something from GPT for which the LOWs are not prepared and ANGLE does not provide. Below, we argue that ANGLE is teaching proof *planning* as well as GPT, however, it is perhaps at a disadvantage in teaching proof *execution*. To the extent that learning proof planning precedes the acquisition of execution skill, it may be that the HIGHs have begun to master planning and are beginning to focus on execution, while the LOWs are still engaged in mastering planning. Thus, the LOWs are perhaps unable to take advantage of GPT's slightly better instruction of execution skill.

3.3.2 Differences in Planning vs. Execution

As described in the Method section, we graded the Proof Construction test in more than one way, the goal being to try to access student's proof planning skills separately from their overall proof writing skills. The graph on the left in Figure 3.4 shows that both tutor groups started out with the same proof skills, with a slight trend in favor of GPT.

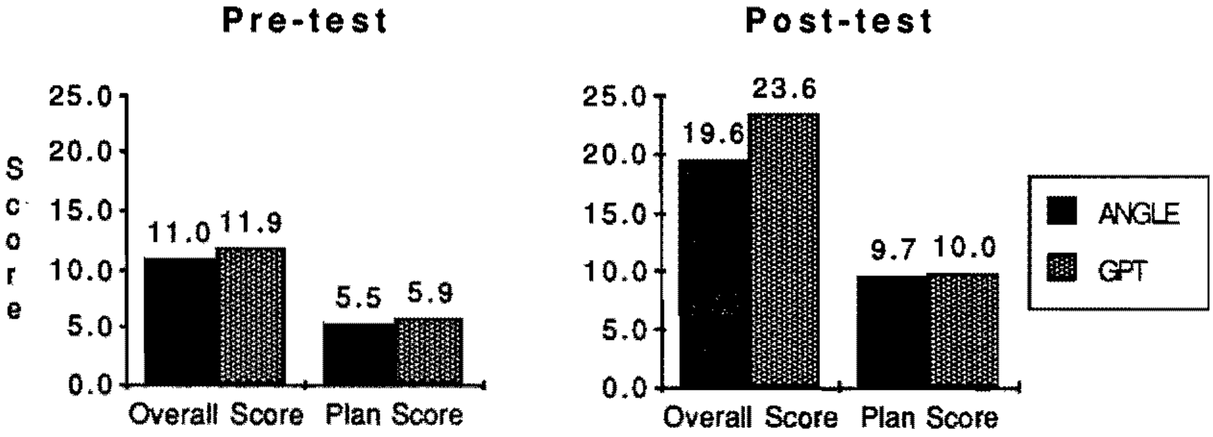


Figure 3.4. Proof execution and planning on the Proof Construction pre- and post-tests. None of the differences are significant. (The trend in the overall score is the raw score corresponding with the percentages shown in the "Proof" column of Figure 3.2, $p > .13$.)

The post-test scores are shown on the right in Figure 3.4. We see the trend in favor of GPT in the overall score, 23.6 vs. 19.6, disappears in the planning score, 10.0 vs. 9.7. In other words, while it may be possible that the current version of ANGLE is not as good as GPT at teaching execution skills, it appears to be doing equally well teaching planning.

From the perspective of the problem solving theory developed above, this result is quite interesting as it adds further evidence in favor of the psychological reality of the abstract planning space. In order to pin this result down further, we took some other measures from students' proof solutions as described in the method section.

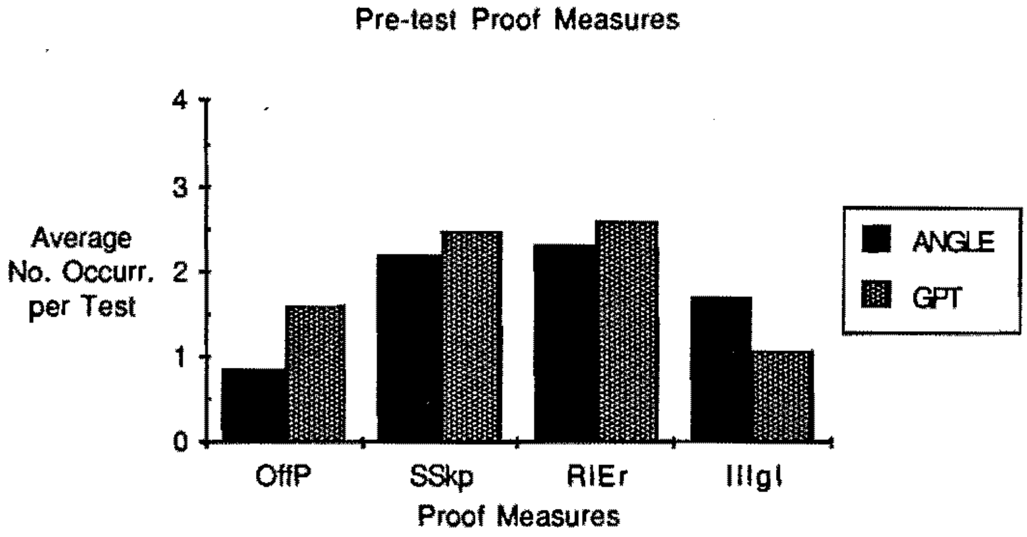


Figure 3.5. Proof measures on the Proof Construction *pre-test.* None of the differences are significant.

Figure 3.5 shows these measures taken from the proof pre-tests. None of these differences are significant (though the positive trend on OffPath moves and the negative trend on Illegal moves are both consistent with the conjecture that the GPT students were better to start).

Figure 3.6 shows the proof measures taken from the proof post-test. Here we find a significant difference in the number of steps ANGLE subjects are skipping relative to GPT subjects. This difference explains why the GPT students might be doing better on execution, but not on planning. In other words, step-skipping (along with rule errors) measure the number of execution errors subjects are making on the tests. Basically, ANGLE students' plans are just as good, but their execution is lacking relative to GPT students. While they get as many of the key steps in a proof, when it comes to translating them into the formal language of geometry they leave out some of the details.

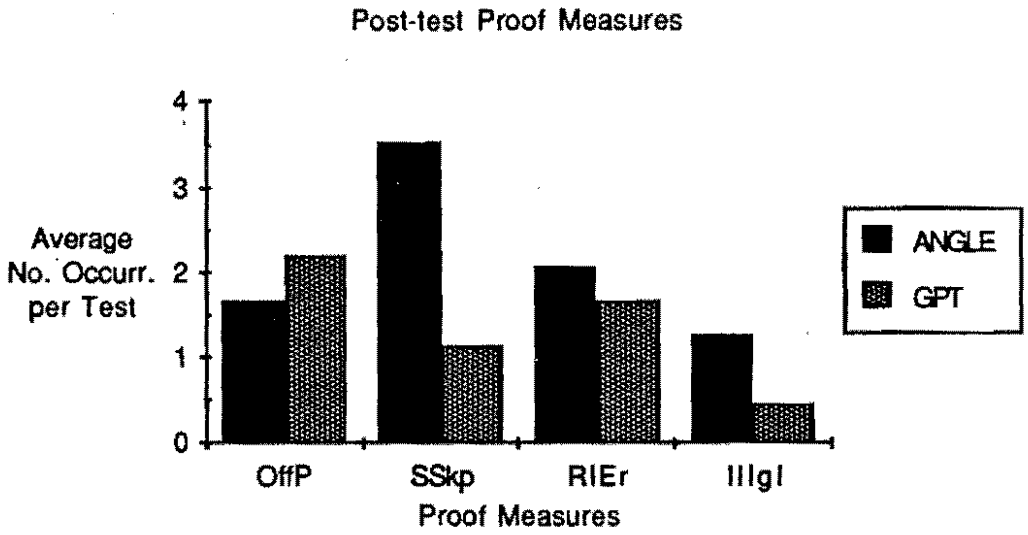


Figure 3.6. Proof measures on the Proof Construction *post-test*. The SSkp (steps skipped) difference is significant (p < .001). Note: While the step-skipping per test goes up from the pre-test to the post-test, this does not indicate Ss are getting worse, rather, it reflects the increased opportunities to step-skip that Ss are getting by making more planning inferences on the post-test.

3.3.3 Truth Judgment Results

As described in Section 3.2.5, it was hoped that the Truth Judgment test would measure students' planning abilities independently of their execution skills and that improvements in planning due to tutoring would show up in pre-to-post test differences. However, there was little if any transfer from the proof instruction (from either tutor) to this test. While students scores improved from 67% correct on the Truth Judgment pre-test to 72% correct on the post-test, this trend was only marginally significant (p=.09)[1].

Given the relative lack of improvement on this test, the invited conclusion is that the tutoring did not improve students' planning. However, this conclusion contradicts the results of the Proof Construction test which indicated students' planning abilities did improve. In addition, there is evidence from the Truth Judgment results that supports a claim that students did not fully appreciate (nor implement) the relevance of proof planning to solving these problems:

1) While planning improved due to proof instruction, students did not show significant improvement in the Truth Judgment test indicating they were not using what they had learned.

2) Students were given 15 minutes to perform this test and most finished in 10 indicating they were not spending the time necessary to come up with proof plans for the more difficult items on this test.

[1] Note since this test had yes-no items on it, these scores should be thought about relative to the fact that someone knowing nothing is likely to score about 50% by chance.

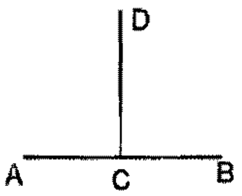
3) There is evidence that other, non-proof related, factors played a significant role in students' answer selection. This evidence is discussed below.

3.3.3.1 The Misleading Model Effect. Instead of proof planning, it appears subjects were using a less-sophisticated approach. This approach involves using the provided diagram as a model of the given statements and simply reading-off whether or not the goal statement appears true. The possibility of such a strategy was anticipated and the items were designed to test for it (see Section 3.2.5). In other words, the items were designed such that reading-off the diagram yields the right answer for half of the items, ones in which the answer and diagram are *consistent*, but yields the wrong answer for the other half of the items, ones in which the answer and diagram are *inconsistent*. Items are consistent when either the goal looks true in the diagram and the answer is YES or the goal looks false and the answer is CAN'T TELL; items are inconsistent when either the goal looks true and the answer is CAN'T TELL or the goal looks false and the answer is YES. Table 3.1 shows two of the items. (I'll discuss the easy-hard dimension below.) The correct answer for the first of these, problem 1b, is CAN'T TELL and since the goal looks true in the diagram (suggesting the YES answer), it is an inconsistent item. The correct answer for the second item, problem 9a, is YES and since the goal looks true (suggesting the YES answer), this is a consistent item.

Table 3.1. Examples of Truth Judgment items.

Easy, CAN'T TELL, inconsistent item:

1b. If $\overline{AC} \cong \overline{CB}$, must $\overline{AB} \perp \overline{CD}$?

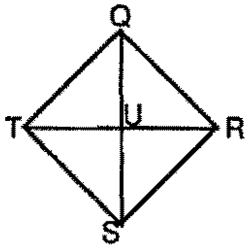


YES

CAN'T TELL

Hard, YES, consistent item:

9a. If $\angle TQU \cong \angle RQU$ and $\angle QTU \cong \angle QRU$, must $\angle STU \cong \angle SRU$?



YES

CAN'T TELL

In fact, subjects were often fooled by the inconsistent items indicating a significant tendency to use the given diagram as a model. Figure 3.7 shows this result. The horizontal axis indicates whether the correct answer is YES or CAN'T TELL, while the graphed lines indicate whether the item was consistent or inconsistent. The vertical axis is the average percent correct for all students on both pre- and post-tests. Recall that if students simply guess randomly they are likely to get about 50% correct by chance.

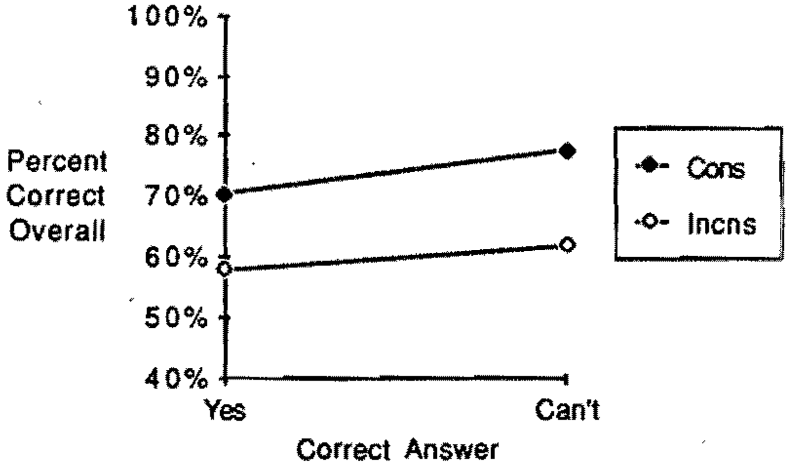


Figure 3.7. Percent correct on the Truth Judgment items broken down according to the correct answer and diagram consistency. Students perform significantly better on the consistent diagrams (p < .001).

The difference between the consistent and inconsistent items is significant (p < .001). There is no overall difference between the YES and the CAN'T TELL items (p = .12) nor is there a difference between *good diagram* items, where the givens and goal look true in the diagram (independent of the answer), and *bad diagram* items, where the givens and goal look false (p = .78).

3.3.3.2 The Plan Difficulty Effect. In addition to the bias to read-off the conclusion from the diagram, there appears to be another strong bias in students' reasoning strategy. Figure 3.8 shows the results splitting the items into *easy* items, those corresponding with one planning step, and *hard* items, those corresponding with multiple planning steps. Again, consider the example items in Table 3.1. Problem 1b is easy because it corresponds with one planning step, that is, the corresponding YES item can be proven with one DC schema (the PERPENDICULAR-ADJACENT-ANGLES schema in this case). Problem 9a is hard because proving **∠STU ≅ ∠SRU** from the givens involves multiple schemas (either 3 TRI-CONG-SHARED-SIDE schemas or 2 TRI-CONG-SHARED-SIDE and 1 PERPENDICULAR-CROSS schema).

On the easy items, students are biased to answer YES (57% of the time) indicated by the fact that they get more YES items correct than CAN'T TELL items. In contrast, on the hard items students are biased to answer CAN'T TELL (62% of the time) indicated by the fact that they get more CAN'T TELL items correct than YES items. The interaction between item difficulty and answer type is significant (p < .001). Looking at this interaction more directly, within the easy items the difference between the YES and CAN'T TELL items is significant (p < .002), and within the hard items this difference is also significant (p < .001).

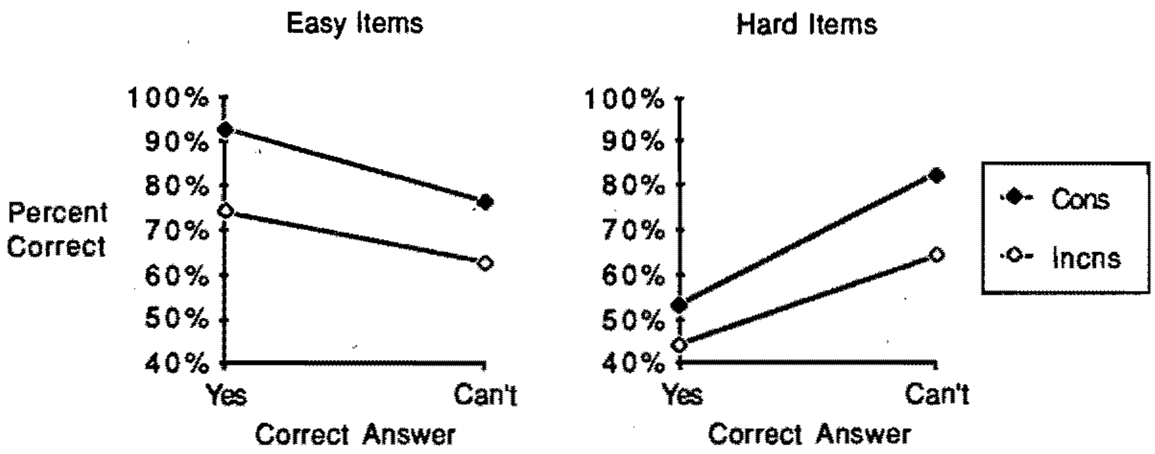


Figure 3.8. Percent correct on *easy* items and *hard* items. In addition to the bias to be deceived by the diagram, students were biased to answer YES on easy items and CAN'T TELL on hard items. The interaction between item difficulty and answer type is significant (p < .001).

There appear to be two possible interpretations for this plan difficulty effect: (1) a *locality* hypothesis or (2) a *complexity* hypothesis. The locality hypothesis is that subjects are applying a type of *locality heuristic*: "constraints on a part (or parts) of an object are more likely to constrain nearby parts of the same object than more distant parts of other objects". (Note, other uses of spatial locality as a heuristic for reasoning were discussed in Sections 1.2.2 and 1.5.3.1.) In problem 1b, for example, the given and goal statement apply to nearby parts of the same object, that is, the T-shaped configuration of perpendicular lines. So, following the heuristic, one would tend to (incorrectly) answer YES as 8 of the 30 subjects did. The other easy items can be characterized as containing one "object" (corresponding to one of DC's diagram configurations) and the givens and goals always refer to parts of this object. Thus, following the locality heuristic one would tend to say YES on these items, all other things equal. In contrast, consider problem 9a where the givens constrain parts of the triangles above the line $\overline{TR}$ while the goal refers to parts of the triangles below this line. The heuristic would lead one to conclude that the givens and goal seem unrelated and so, (incorrectly) answer CAN'T TELL as 17 of the subjects did. The other hard items contain many "objects" and the givens always refer to a different object than the goal. Thus, all other things equal, the heuristic suggests to answer CAN'T TELL.

A second hypothesis, the complexity hypothesis, is simply that for complex diagrams, subjects will tend to answer CAN'T TELL while for simple diagrams, they will tend to answer YES. Since the diagrams for the hard problems were more complex than the diagrams for the easy problems, this hypothesis is consistent with the data. This confound can be alleviated by designing easy problems with diagrams that are equally as complex as the hard problems – that is, some irrelevant lines can be added to the diagrams of the easy problems to make them more complex.

3.3.3.3 Implications for the Mental Models Theory of Reasoning. These Geometry Truth Judgment problems are much like logical syllogisms. While they are not limited to two premises as logical syllogisms are and they require specific geometry knowledge as well as general logical knowledge, they do have the same structure as logical syllogisms: In both cases subjects are given certain premises and asked if a certain conclusion follows from those premises. Perhaps the leading theory on how humans solve syllogisms is Johnson-Laird's (1983) mental models theory. In this context, *model* is a specific instance of the general statements made in the premises of the syllogism. The mental model theory proposes that instead of using logical rules to solve syllogisms, people imagine a model (sometimes more than one) which is consistent with the premises. Then they check the problem conclusion to see if it is consistent with the model(s). If it is, they answer YES, otherwise they answer NOT VALID.

What is unique about these Geometry Truth Judgment problems is that while in typical syllogism experiments subjects imagine their own model(s), here subjects are given a candidate model[1]. The diagram that goes along with a problem is a model in that it contains specific features which are not stated in the premises. Some of these features may actually follow from the premises, for example, **∠STU ≅ ∠SRU** in problem 9a of Table 3.1, while others may be fortuitous, for example, **∠STU ≅ ∠QTU** in the same diagram.

By giving subjects a model, the Truth Judgment test provides a different kind of evidence for the mental models theory. In the typical experiment, the match between the theory and the subjects' actual reasoning process is performed indirectly by comparing the error patterns predicted by the theory with the subjects' error patterns. In contrast, the Truth Judgment test attempts to directly manipulate the reasoning process by providing a candidate model. If students are reasoning purely by abstract rules, they should not be influenced by the model we provide. The clear evidence that subjects are influenced by inconsistent diagrams indicates they tend to use the provided model and that, therefore, in those cases they are reasoning by model rather than by rule.

If performed systematically and carefully, model-based reasoning can be quite effective. However, it can lead to error, for example, in logical syllogisms when not enough models are considered. This type of model construction error also accounts for the subjects tendency toward errors on the CAN'T TELL-inconsistent items in which the diagram was over-specialized. Rather than attempting to construct other models (counter-examples) that might contradict the over-specializations, subjects tended to be influenced by the given model.

The errors on the YES-inconsistent items are different. In this case, the provided diagram was not only inconsistent with the goal, but also inconsistent with the givens – in other words, the provided diagrams were not good models. An effective form of model-based reasoning would notice this discrepancy and construct a new model consistent with the givens. (Note that if the goal is true, that is, if it follows from the givens, then it will necessarily appear true in a good model.)

As was hinted at in Section 1.6.1, an important supplement to model-based reasoning is a component for chaining inferences from object to object. Without this, problems like the *hard* items could not be effectively solved.

[1] While subjects were certainly free to construct other models, the evidence suggests that, for the most part, they did not.

3.3.3.4 Naive Geometry. Looking at these results a different way, they suggest the beginnings of a theory of naive geometry similar to the work on naive physics. Perhaps the difficulty students have with learning proof construction stems in part from conflicts with (or inadequacies of) their prior geometry conceptions or, more to the point, their prior reasoning strategies.

This section has identified a number of candidate geometry misconceptions (for lack of a better word). All of these are best thought of as representing tendencies rather than strict modes of thought. The first, as indicated by the lack of transfer from the proof instruction, is the inability to recognize, without prompting, the relevance of proof construction to making truth judgments. The misleading model effect suggests two problems with students' use of mental models corresponding with the two types of inconsistent items. In the case of the CAN'T TELL-inconsistent items (e.g., problem 1a in Table 3.1), they often fail to generate an alternative model, a counter-example, that would indicate that although the goal looks true in the provided model, it does so only because the diagram is over-specialized. In the case of the YES-inconsistent items, they often fail to notice that the given model is not actually a model of the problem at all since the givens are not true in it.

3.3.4 Analysis of On-line Tutoring Data

Both computer tutors maintained protocol records of student actions and tutor responses. We can look as these records to get a finer grain view of the learning that occurred and, in particular, we can investigate issues of (1) ease of interface use and (2) effectiveness of tutoring strategies and messages.

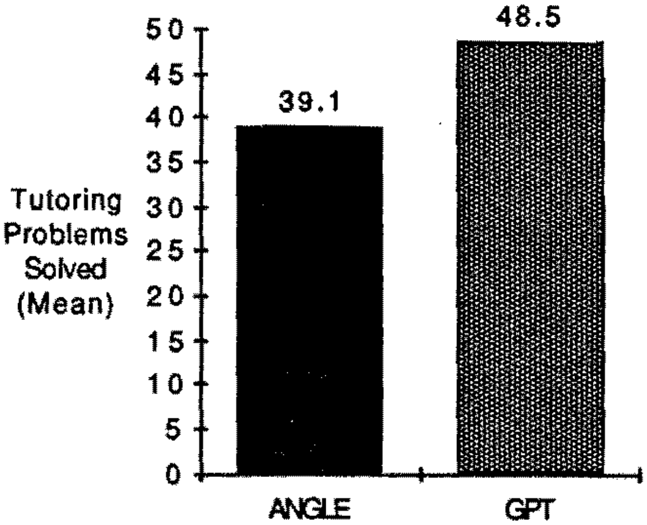


Figure 3.9. Average number of problems solved by students during tutoring. The ANGLE subjects solved significantly fewer problems (p < .05).

From watching students during the study it appeared that ANGLE was not always doing a good job of keeping students on-track during tutoring. Occasionally, students were observed clearly floundering. One potential manifestation of this is that the

ANGLE students solved significantly fewer problems during training than the GPT students did (see Figure 3.9), $p < .05$.

It appears ANGLE subjects were doing more floundering than GPT as a result of the flexibility of the ANGLE interface and tutoring strategy. Flexibility is typically considered a virtue. It is considered particularly important by those advocating discovery learning and also by many of the early intelligent tutoring system designers (e.g., see Brown and Burton, 1982). However, in this study it appeared that a number of problems students encountered stemmed from ANGLE's flexibility.

3.3.4.1 Problems with the Bottom-out Hints. In both ANGLE and GPT, strategic hints are focussed on a good next step the student might take. Hints start out general and get successively more specific if a student continues to have trouble. In the end, the hints bottom-out by telling students exactly what they should do next. This is the last line of defense against continued floundering and is intended to be fool proof in helping the student get closer to the problem solution. In GPT, the hints bottom-out by not only telling the student what to do next, but actually doing it for them. To prevent the students from relying too heavily on the tutor, ANGLE did not perform the next-step for the student. It simply told them what to do.

Unfortunately, students were not always able to translate these "bottom-out" hints into the appropriate actions and because of the flexibility designed into ANGLE, students were not forced to perform them. Thus, they could potentially go off and continue to perform unproductive actions and waste valuable learning time.

In turns out that this situation arose quite often. Bottom-out hints were given on 128 or 23% of the 561 problems solved.[1] On average, ANGLE students received about 2.5 bottom-out hints on these 128 problems. Often students would not immediately perform the suggested next step. This is measured by two statistics. Since ANGLE repeats the bottom-out hint if the student continues to have trouble, one relevant statistic is the number of times a particular bottom-out hint was repeated on average. Bottom-out hints were repeated about once (0.97) on average. Any repeats indicate a discrepancy between the tutoring intentions and what the student is thinking.

The second statistic is more dramatic and leads to a more telling result. It is the amount of time that is spent from the time the bottom-out hint is given to the time the student performs the suggested next step. Since there is more than one solution to these problems, it is possible that the student never performs the suggested next step in the course of completing the problems. This happened rarely though, only about 8% of the bottom-out hints given were never followed up with the suggested next step. In the cases where students do end up eventually performing the suggested next step, about 75 seconds elapse on average. Multiplying this by the number of bottom-out hints given, we find that the students in this study spent about 26.6 minutes on average between being told what to do next and actually doing it. This is about 6% of the total instruction time.

It is certainly possible that some useful work is going on during this time. The student may be pursuing alternative paths, working out details, posting future subgoal "islands", or working backward. Nevertheless, this statistic clearly indicates a discrepancy between the way I expected the student-tutor interaction to go and the

[1] Some of the early protocol data was lost. Students actually solved 586 problems.

way it actually did. In the Chapter 4, I suggest a remedy to this problem involving a simple "interface tutor" that will kick-in after bottom-out hints are given and guide the student in performing the suggested action.

3.3.4.2 Trash Can Misuse. Another fairly common behavior that was indicative of floundering was the way in which students occasionally misused the trash can that is available (in the lower left corner of the screen) to dispose of unwanted statements. There were quite a few cases where a student would post a useful schema, end up throwing it out, and then have to reconstruct it again later. One of the intended benefits of ANGLE's interface was its flexibility to allow students to post "island" subgoals without having to immediately link them into the proof. I observed a couple of pilot subjects using the screen in this way on an earlier version of the system without the trash can (briefly described in Koedinger & Anderson, 1990b). Sometimes students in this study posted such island subgoals, but there are many cases where this might have happened but didn't because the subject threw away the statement.

Perhaps making the trash can less clearly visible and not so simple to use, would discourage this kind of misuse. In addition, it would probably be a good idea for the tutor to stop students from throwing away certain statements – especially when the statement is one of the possible next steps.

3.3.4.3 Execution Feedback Inadequate. Another curable problem, that contributed to floundering involved the overly simple execution feedback that said, for example, "the premises of SAS should be 3 segment or angle congruence statements". Students were quite confused when this came up in cases where they actually had 3 segment or angle congruence statements as premises. The statements they had chosen were wrong, for example, because they corresponded with the illegal side-side-angle combination.

3.4 CONCLUSIONS OF THIS PRELIMINARY EVALUATION

Unfortunately, we are not yet at the point where we can decisively affirm or disconfirm the hypothesis put forth at the beginning of this chapter, namely, that the development of more accurate and powerful cognitive models of problem solving can lead to major improvements in the instruction of problem solving. We need to address two key problems in order to perform a better test of this hypothesis: (1) expand the size of the curriculum, and (2) improve the implementation of ANGLE's interface and tutoring components.

Because of the relatively short time for instruction (8 hours), the curriculum in this study was kept small (only 8 geometry rules and/or 4 schemas). As it turned out, the majority of students had plenty of time to finish the 30 problems provided – only 4 did not and the others did problems over again. Thus, the curriculum could easily be expanded without increasing the instruction time. This is important because a key difference between the problem solving method ANGLE teaches and the one GPT teaches is that ANGLE's method is more effective in the large search space of geometry rules (the execution space). By limiting the curriculum, the size of the execution space is reduced and if it is small enough, the search benefits of ANGLE's method effectively disappear. With a larger curriculum, then, we are more likely to see a difference between ANGLE and GPT.

The second problem we need to address is the limitations in the implementation of ANGLE's interface and tutoring components as revealed by this study. Potential remedies for the more significant of these limitations are discussed in the first section of Chapter 4.

Perhaps it is well-known to those few who have designed and successfully tested an intelligent tutoring system, that getting the student-tutor interaction right requires careful and lengthy user study that goes beyond informally watching a few students work with the system for a couple of hours. While I knew prior to the study that ANGLE's tutoring schemes were not perfect, I did not anticipate certain problems like the trouble students had understanding and implementing the bottom-out hints. Perhaps the conclusion one should draw is that such problems are necessarily a part of the development process and that the tuning of the interface and tutoring components is equally as important as following through on the implications of a cognitive model.

Still, it remains to be seen what instructional effect an improved cognitive model can have. When the interface and tutoring components of ANGLE have been cleaned up and further tuned, a more realistic test can be performed.

CHAPTER 4.
GENERAL DISCUSSION AND FUTURE PLANS

This chapter has a double role of (1) indicating directions for future work as inspired by the preliminary evaluation of ANGLE reported in Chapter 3, and (2) discussing the general implications of this research for model-based instructional design. One obvious direction for future work is improving the current limitations in the implementation of ANGLE's interface and tutoring components. This direction is discussed in Section 4.1. A less obvious direction involves improving the knowledge measurement tools that are crucial both for measuring learning outcomes of different instructional methods and for understanding the nature of that learning. This direction is discussed in Section 4.2. Finally, Section 4.3 concludes by recapping this research program and discussing how it might be applied in other domains.

4.1 PLANS FOR CONTINUED DEVELOPMENT OF ANGLE

Perhaps, the most important agenda item for improving ANGLE is to get rid of a number of minor bugs and discrepancies between the expected and observed student-tutor interaction. Improvements are needed in the interface and the wording of the tutor messages. Rather than mention all of these, I'll focus on the ones that could be of more general interest.

4.1.1 Improving the Interface

4.1.1.1 Dealing With the Bottom-out Hint Problem. One of the major discrepancies between the expected and observed student-tutor interaction was the way in which students responded (or failed to respond) to the bottom-out hints when they were provided by ANGLE. One possible remedy would be to do as is done in GPT, and perform the next step for students in this situation. An intermediate possibility is to still have the student perform the step, but require them to do it before going on. In addition to keeping the student at least physically involved, this approach has the added side-effect of essentially tutoring them on the interface.

In other words, the bottom-out message essentially presents the student with an interface goal, for example, "Select the CONGRUENT TRIANGLES concept and indicate $\triangle ABC \equiv \triangle XYZ$ in the diagram". The proposed change would require the student to perform the necessary interface actions to implement this step. If the student performs an inappropriate interface action, the tutor would give them feedback specifically addressed at performing the correct interface action.

This change amounts to an on-line interface tutor. Since the interface goals presented in the bottom-out hints can be performed by a simple list of interface actions, implementing this interface tutor should not be too difficult.

4.1.1.2 Dealing with the Whole-statement Encoding Problem. One of the difficulties encountered in thinking about how the interface should reify the cognitive model, involved the distinction made in the model between the whole-statement of a schema and the schema itself. For the most part, whole-statements can be considered equivalent to schemas and thus, in the ANGLE interface a schema/concept is represented on the screen with both its diagram configuration and its whole-statement. However, there are some places where the whole-statement might be treated independently of the schema, for example, when a whole-statement appears as part of the problem givens or goal. In this case, there is a translation process necessary to convert from the whole-statement to a corresponding schema. This process is part of the statement encoding process in the DC model (see section 1.3.2.2). The question was whether and how this whole-statement encoding process would be represented in ANGLE.

A compromise was made. On one hand, whole-statements that appeared in the givens or goal could be treated, for the purposes of making justification links, just as if they were schemas. This is a short cut in which the whole-statement encoding process is left implicit. Alternatively, the student could create the corresponding schema and attach it to the whole-statement, and thus explicitly perform the whole-statement encoding process. The interface responded by putting the schema (whole-statement plus configuration instance) in the given or goal position that the whole-statement had occupied. An example of the result is shown in Figure 2.13.

While the student was free to do it either way, in the case that the student needed help at this point, the tutor suggested that they explicitly perform the whole-statement encoding step. Two problems arose. While the interface actions to perform this step were the same as those necessary to select and justify any schema, the effect was different (i.e., the schema replaces the whole-statement to which it was linked). Understanding this interface operation was another thing for students to learn that may have distracted them from learning geometry.

A second problem was the unanticipated awkwardness that sometimes resulted from the way the schema hint templates got applied to this situation. While the majority of these hints read just fine, one of them, the justification bottom-out hint (see section 2.4.3.2), turned out to be particularly awkward. Since schemas/concepts were referred to in the hint messages using the same label as the label for the whole-statement, the justification bottom-out hint message, which has the generic form "Justify <the desired schema> using statements: <the necessary statements>.", ended up as "Justify <whole-statement> using statements: <whole-statement>.". For example, if the whole-statement $\triangle ABD \cong \triangle CBD$ appeared as a given and the corresponding TRIANGLE-CONGRUENCE-SHARED-SIDE schema had been constructed, then if the student happened to need a bottom-out hint, it would read "Justify $\triangle ABD \cong \triangle CBD$ using statements: $\triangle ABD \cong \triangle CBD$".

This proved a bit confusing to students. Many of the cases where a student never performed the step suggested by a bottom-out hint were cases where the student got this hint, didn't know how to implement it, and was eventually able to perform a different step (i.e., one that involved skipping the whole-statement encoding step) with no help from the tutor.

The proposed remedy for this problem is quite simple: Eliminate the need to explicitly perform whole-statement encoding. In addition, whenever problem given or goal is a whole-statement, the tutor should display this statement as a schema complete with the diagram configuration. Prior to the study I had thought that encouraging the whole-statement encoding process would be helpful for the student who did not know what to do next. In such a situation, it seemed that the student's problem might be that he or she wasn't recognizing in the diagram the particular objects referred to by the whole-statement. The process of selecting the corresponding schema would help the student make this recognition and focus on this schema. From watching students, this recognition of given or goal statements in the diagram does not appear to be a great difficulty, especially in comparison to the difficulties that arose from allowing and sometimes encouraging the explicit whole-statement encoding step.

4.1.2 Improving Tutoring Messages and Strategies

4.1.2.1 Adding Buggy Ways-to-prove. One of the problems with ANGLE's tutoring messages was that the execution feedback was quite simple and sometimes misleading. In particular, this feedback did not respond well to common bugs, like trying to prove triangles congruent using two sides and a non-included angle. These situations are captured in GPT by matching them against particular "buggy" production rules. A similar thing can be done in ANGLE by elaborating the schema representation to include buggy ways-to-prove in addition to the existing (non-buggy) ways-to-prove. Tutoring messages can then be attached to these, just like they are attached to buggy rules in GPT.

4.1.2.2 What's the Proper Role of Execution Training? One important question that was considered before the study and still remains unanswered is: what is the proper role of execution training? One could take the view that proof instruction in high school geometry should emphasize proof planning and not be too concerned about proof execution, that is, whether students get the formal details exactly right. In fact, a recent proposal for high school standard suggests a deemphasis on formal proof and more emphasis on informal proof (Romberg, 1987). However, at least for comparison sake, it seemed important that the two groups (ANGLE and GPT students) be tested on the same standard two-column format. Thus, it also seemed important to give training within ANGLE on execution as well as proof planning.

Following the cognitive model, ANGLE's feedback scheme always suggests planning moves first and only suggests execution moves once a complete plan has been found. However, consistent with the effort to make the system flexible, ANGLE allows students to integrate planning and execution. In fact, students rarely completed planning before beginning execution. One measure of this is the percent of inferences that occurred after execution began, but before planning was finished. Thus, for example, if all the planning is done first this percentage should be 0. On average, 47% of students' inferences were in this mixed stage. Students' tendency to mix planning and execution was not quite significantly correlated with post-test performance ($p = .06$). However, the pattern is that the students who began execution early also scored better on the post-test. This is probably a reflection of students' prior familiarity with the execution space and the good students' better facility with it.

Another issue relating to the role of the execution training in ANGLE, is the complexities to the interface that it added. Again such complexities presented students with a learning task that distracted them from geometry.

A couple of possibilities may be pursued with respect to this issue. One is to eliminate the execution training from ANGLE and simply focus on tutoring proof plans. It would be interesting to see how such instruction would transfer to the task of coming up with a completely detailed two-column proof. Another possibility is to have the tutor enforce the planning first approach of the cognitive model. Only after completing a plan would students be allowed to do the proof execution.

4.2 IMPROVING KNOWLEDGE MEASURES

4.2.1 Improving the Truth Judgement Test

It was hoped that the Truth Judgment test would measure students' planning abilities in a way that wouldn't be masked by lack of execution abilities. Having such a test is still desirable, however, as it turned out students did not appear to spontaneously see the relevance of proof planning to answering Truth Judgment items. Part of the reason may be due to the fact that as YES-NO type questions these items had the appearance of being easy, and thus, perhaps it did not occur to students that anything as difficult as doing proof planning would be relevant. In addition, students were given only 15 minutes to do the 17 items on this test (in fact, most finished in about 10 minutes). Four of these items could be solved with proofs about as difficult as the four proof problems on the Proof Construction test which students were given 35 minutes to solve. In other words, students really didn't have time to do proofs to help solve these problems.

A number of things can be done to encourage proof planning. First, an illustration should be given to students, prior to taking the test, of how planning a proof can help answer these questions. Second, fewer problems should be given with more time allowed for each. It should be emphasized to students that they have lots of time to think hard about each one of the items. Third, students can be asked to give a reason for their answer. In the case that they answer YES, they should provide a proof sketch. In the case that they answer CAN'T TELL, they should provide a counter example. Examples of both types of reasons should be given prior to the test.

4.2.2 The Need for a Measure of Schema Knowledge

Both GPT and ANGLE are primarily focussed on teaching the process of constructing proofs and not on teaching the declarative knowledge of the basic operators, rules in the case of GPT and schemas/concepts in the case of ANGLE. To the extent that students don't have a reasonable grasp of this basic knowledge, they are likely to have trouble. This situation is potentially more problematic in ANGLE because the units of declarative knowledge in ANGLE, the concepts: (1) are not explicitly taught in the standard curriculum and (2) are much bigger than the formal rules which are the declarative knowledge units in GPT. It seems possible that the effectiveness of ANGLE might interact with the level of students' prior knowledge of the concepts. Thus, it seems important to have a measure of this knowledge.

Such a test might be made up of items, much like the *easy* items on the Truth Judgment test, in which the student is given a diagram configuration and some facts about it, and asked if a particular conclusion follows. One type of item would test knowledge of the part-statements by providing the whole-statement as the given and a possible part-statement as the goal. Another type of item would test knowledge of the ways-to-prove by providing sets of part-statements as the givens and the whole-statement as the goal.

4.3 RESEARCH SUMMARY

This final section provides a recap of the research agenda carried out in this thesis. However, rather than simply summarizing, I've attempted to generalize the key steps and present them as a prescription for tutor design. Certainly there are other theoretically motivated routes to successful tutor design, not to mention getting there by good intuitions or serendipity. This prescription is provided merely as one possible route that may (1) be directly applied in some domains or (2) be used as a departure point for developing related approaches in other domains.

The approach can characterized in five steps:

1. Identify the *execution space.*
2. Look for *implicit planning* in verbal reports.
3. Model this implicit planning.
4. Use the model to drive tutor design.
5. *Tune* the tutor implementation.

Below I discuss the significance of each step, suggest how it might be done in general, and review how it was done in this project.

4.3.1 Identify the Execution Space

This step sets the stage for step 2 where one looks for underlying problem solving processes that are effectively hidden in current instruction. First, we need to know what aspects of the problem solving process are revealed, at least implicitly, by current instruction. This is the task of identifying the execution space. The execution space for a domain is the problem space most directly induced from the way problem solution steps are typically or conventionally written down[1]. In other words, the operators of this space correspond one-to-one with the written problem steps.

As discussed in Section 1.1, the execution space operators for geometry are the various definitions, postulates, and theorems that appear as the "reasons" in the steps of the conventional two-column proof format. The execution space operators for algebra equation solving are the various rules (e.g., You can add the same number to both sides) for manipulating equations. In physics problem solving (e.g., the kind analyzed by Larkin, et. al., 1980), the execution space operators might be the relevant physics formulas.

Another potential guide to the operators of the execution space is to look at the units of knowledge that are provided to students in their textbooks or lectures. Quite often, these units of knowledge correspond with the written problem steps. For example, the traditional geometry curriculum is organized around presenting and illustrating the very same rules that appear as reasons in two-column proof solutions. A similar situation is apparent in algebra and physics.

A straight-forward way to model problem solving in these domains is as a heuristic search in the execution space – the only trick is to find appropriate operator selection heuristics. From the perspective of a student, to the extent that the execution space provides a good characterization of skilled problem solving, his or her learning job is made easier. By definition, execution operators can be induced fairly directly from the steps of worked out examples and may be supported by verbal descriptions in textbooks and lectures. For example, consider algebra equation solving. An example worked out solution is shown in Table 4.1.

[1] Newell and Simon (1972, p. 144) refered to a "basic problem space" and gave examples of one in a number of domains. It is evident from their examples that what they meant by a basic problem space is similar to what I mean by an execution space, however, they did not explicitly define it.

Table 4.1 An worked solution in the domain of Algebra equation solving.

3x - 13	=	2(x - 3)	
3x - 13	=	2x - 6	Distribute
3x - 13 - 2x	=	- 6	x's to left side
3x - 2x	=	- 6 + 13	Num's to right side
x	=	7	

In this domain, the execution operators can be fairly directly induced from the steps in worked out examples like that in Table 4.1. This claim is supported by the fact that an early machine learning program did exactly that (Neves, 1978). In addition, the general difference-reduction heuristic turns out to be an effective means of operator selection. Because this domain independent weak-method works in this domain, learning operator selection is relatively easy.

While heuristic search in the execution space is a straight-forward candidate for modeling problem solving in a domain, it may not be the problem space that skilled problem solvers typically use in this domain. The next step is to see if it is or not.

4.3.2 Look for Implicit Planning in Verbal Reports

The purpose of this step is to identify the nature of skilled problem solving in the domain and in particular, to see if it deviates from heuristic search in the execution space. To do so, one can collect concurrent verbal reports (Ericsson & Simon, 1984) of skilled subjects solving problems in the domain. As Ericsson & Simon point out, subjects should not explain what they are doing, but merely report what they are thinking. To the extent that heuristic search in the execution space provides a good model, subjects' successive verbalizations should correspond with successive states in the execution space.

However, subjects' verbalizations could deviate from the execution space in a number of ways:

1. Multiple execution steps might be *aggregated* into a single verbalization.
2. Successive verbalizations may *skip* steps in the execution space.
3. Verbalizations may not specify an execution state in full detail, but rather only indicate some *abstract* feature(s) of it.

Regularities in such deviations suggest "thinking steps" which are not represented in the execution space. From the perspective of the student, these thinking steps are an implicit part of the planning process which, in contrast to the execution operators, cannot be directly induced from worked out examples. In other words, when there is implicit planning in the thinking of skilled problem solvers, there may be aspects of a successful problem solving method which are hidden in the traditional curriculum.

In Section 1.2, I showed that skilled geometry problem solvers skip steps in execution space (#2 above) while developing an initial proof plan. The knowledge that allows them to do so is hidden in the geometry curriculum. While it is probably possible to induce the execution operators from the steps of worked out geometry proofs in similar fashion to Neves' (1978) program for Algebra, it is not possible to apply a general weak-method like difference-reduction or means-ends analysis to effectively perform operator selection in the execution space of geometry. As the step skipping in the verbal reports suggests, successful search in geometry involves operators other than those in the execution space.

It is possible that no evidence for implicit planning is found in a domain, in other words, skilled subjects work in the execution space. This seems likely in the domain of algebra equation solving. In such a case, building a tutor for this domain may be inappropriate. Conventional instruction may be adequate. In other words, finding no implicit planning would suggest that the conventional written display of problem solving steps corresponds fairly well with the thought steps necessary for successful problem solving in that domain. In this case, well-motivated students may not have much trouble in inducing the necessary operators. A cognitively-based intelligent tutoring system (ITS) is unlikely to be much different from conventional instruction and thus, unlikely to help much. In fact, an ITS for algebra equation solving has been compared with a conventional classroom and although students learn with this tutor, they don't learn any better than students in a normal classroom (J. R. Anderson, personal communication).

4.3.3 Model this Implicit Planning

If evidence for implicit planning is found, the question becomes what is the knowledge that is responsible for the non-execution space inferences? I can offer no sweeping generalizations for how to come up with the elements for a model of implicit planning. I can only say that the diagram configuration schemas described in Section 1.3.1, evolved from an initial attempt to apply ideas about abstract planning and abstract problem spaces (Sacerdoti, 1974; Newell & Simon, 1972). In particular, the first attempt at a model was based on applying the idea of equivalence classes as a way to collapse nearby problem states into one.

In the case of verbalization types (1) and (2) above, it is possible that the implicit planning knowledge is the result of composing execution operators. In section 1.6.3, I argued against a macro-operator interpretation of the genesis of DC-schemas. In the process, a number of potentially general criteria were used to distinguish macro-operators derived from execution operators from operators that merely bear a macro-operator relationship with the execution operators. In the case of verbalization type (3), the abstract problem space ideas are likely to be relevant. For example, Newell and Simon's (p. 152, 1972) augmented problem space for crytarithmetic provides a model of such verbalizations (e.g., abstract features of states like the number must be even).

4.3.4 Use the Model to Drive Tutor Design

Once an accurate model of implicit planning has been developed the challenge is to find a way to communicate this model to students. This issue is discussed in Section 2.1. Since people seem to learn best by doing, directly communicating it to them doesn't usually help much. By design, this model contains problem solving process that are not reflected in the notation of the current curriculum. Thus, it is necessary to invent new notations which reify the previously hidden structures and processes. These notations can be the basis for interface design. For example, diagram configuration schemas became the basis for the icons in ANGLE's concept menu and for the representation of concept instances in the proof graph. In addition, the computer medium affords the possibility of inventing *novel actions* which can more directly reify the processes of the model. For example, the interface actions for selecting a schema instance from the problem diagram reify the diagram parsing process (see Section 2.3.2 and Figure 2.2).

The goal of the design of tutoring strategies and messages should be to support and articulate the problem solving method. In ANGLE, the tutoring strategy of focussing on the schema level as the proper grain size for next-step advice is intended to support the learning of the diagram configuration space (see Section 2.4.4 and Figures 2.13-15). The use of terms like "concept", "part-statement", and "ways-to-prove" is intended to articulate crucial aspects of this novel notation.

4.3.5 Tune the Tutor Implementation

While cognitive models can guide the design of the interface and tutoring components, the process of implementing these guidelines is still somewhat of a black art. Thus, it is important to test and tune the implementation. What seems right intuitively may not be effective. In addition, because of the great complexity of such systems, all possible interactions cannot be anticipated. Long periods of use by multiple students is the only way to "shakedown" possible unanticipated interactions.

Two examples from the preliminary study are most notable. First, there was the problem with bottom-out hints and more generally, with the conflict between interface flexibility and user confusion. Intuitively it had seemed that learning would be facilitated by both (1) always requiring students to enter proof steps even in the case that the tutor has told them what to do, and (2) always allowing students the flexibility to ignore hints, for example, if they had something else they wanted to do. These two notions led to the fact that students were able to ignore ANGLE's bottom-out hints. Perhaps in part because the interface was not clear enough and in part because the wording of the bottom-out hints was conceptually focussed rather than interface focussed, students ignored these hints much more often and for a longer period of time than was expected.

Second, there was the problem with whole-statement encoding. Here again, there were apparently well-motivated goals to (1) maximize the student's involvement, that is, by not doing the whole-statement encoding step for them, and (2) allow flexibility, that is, allowing students to either explicitly perform or skip this step. However, the result was, on one hand, added complexity to the interface with little or no added instructional impact and, on the other hand, an interaction within the generic hinting scheme that resulted in a confusing tutor message.

CONCLUSION

This thesis illustrates a program of basic and applied research that started by understanding the nature of a complex problem solving domain and next, applied this new understanding to the development of improved instruction for that domain. This instruction comes in the form of an intelligent tutoring system called ANGLE.

I conclude by reviewing the general contributions of this thesis as stated in the introduction:

1. A new methodology for verbal protocol analysis involving the identification of *step-skipping* with respect to the *execution space* of a domain.

In Section 1.1, I defined the execution space as the problem space that corresponds one-to-one with the steps that problem solvers conventionally write down in solving problems in that domain. I showed how one could use the execution space of a domain as the basis for analyzing verbal reports of subjects solving problems in that domain. By looking for step-skipping with respect to the execution space (see Section 1.2), one can identify *implicit planning* (i.e., thought that is not represented in the conventional notations of the domain) and take a step toward both a deeper understanding of the domain and a promising approach to improved instruction.

2. A new theory of geometry expertise (DC) that accurately describes human behavior, has an efficient computer implementation, and pulls together a number of empirical results on the nature of human expertise.

In Section 1.3, I described DC, a model of skilled geometers informal and intuitive proof planning skills. This model is based on knowledge components, diagram configuration schemas, which merge *perceptual knowledge* about important geometric categories, *conceptual knowledge* about the properties and sufficiency conditions of these categories, and *rule knowledge* of how these categories relate to the formal language of geometry. Section 1.4 established the computational power and empirical accuracy of the model. Section 1.6.2 showed how DC's perceptually-based schemas provide both a detailed explanation of experts' ability, in certain domains, to solve relatively simple problems by pure forward inferencing and an integration of the empirical results indicating experts' superior problem-state memory and their superior problem solving effectiveness.

3. A detailed characterization of the end-state of a complex learning process that challenges current learning theories and that can be used as a test-case for new learning theories.

In Section 1.6.3, I discussed how straight-forward applications of learning theories such as ACT* or Soar are unlikely to produce the regularity in knowledge organization reflected by DC's schemas. In particular, this regularity seems to exclude deductive or symbol-level learning processes which acquire expertise through the composition of execution space operators. Rather, it seems more likely that DC schemas (and perhaps perceptually-based planning schemas in general) are learned through inductive or knowledge-level learning processes. These inductive learning mechanisms would include a perceptual chunking process capable of creating ever-larger diagram configurations (percepts in general terms), and a categorization process capable of attaching part-statements (properties) and ways-to-prove (sufficiency conditions) to these percepts.

The fact that novice problem solvers exhibited DC-like step-stepping on the proof pre-test (see Section 3.3.2) provides further evidence that learning in geometry doesn't start with the execution space and then improve on this representation through composing operators. Rather, very early in the learning process, these novices appear to have acquired some non-execution operators that, though sometimes erroneous, bear a closer similarity with DC schemas than with the formal rules of geometry.

4. A theory-based approach to the design of the interface and tutoring components of an intelligent tutoring system (ANGLE).

In Chapter 2, I showed how a cognitive model of implicit planning (DC) could be translated into design specifications for the interface and tutoring components of an intelligent tutoring system. Perhaps the most important aspect of this approach is how a model of implicit planning can be used to design novel interface notations and actions. By definition, a model of implicit planning must make use of representations and processes that are not represented in the current notation of the domain. Thus, with a model of implicit planning in hand, the tutor designer can use it to guide the invention of new notations that more accurately reflect the processes of skilled problem solving. As part of a tutor's interface, these notations deliver instruction implicitly by providing students with a more cognitively meaningful way to think about the domain.

5. An initial test of the hypothesis that the development of more accurate and powerful cognitive models of problem solving can lead to major improvements in the instruction of problem solving, particularly within the context of an intelligent tutoring system.

While there have been numerous successful examples of applying cognitive science to instructional design (e.g., Anderson, et. al., 1990), these have been cases of comparing traditional (non-cognitive) instruction with cognitively-based instruction. These studies show the impact of designing instruction based on a cognitive model rather than on intuitions. The hypothesis above poses a more detailed question: Do better cognitive models lead to better instruction?

By testing this hypothesis in the medium of intelligent tutoring systems, we can have both control over the exact nature of the instruction that is delivered (which is hard to do with human teachers) and maintain some of the on-line flexibility of a human teacher (which cannot be done with text-based instruction). While the preliminary study described in Chapter 3 did not conclusively affirm or negate this hypothesis, it moves us a step closer. Despite the fact that much less effort has been put into ANGLE to this point, it is about equal in instructional effectiveness to the previously successful Geometry Proof Tutor. The hope is that by eliminating the problems with ANGLE's implementation and expanding its curriculum, we can exceed the effectiveness of GPT and confirm the hypothesis.

REFERENCES

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

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds), *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence, 42*, 7-49.

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: Lawrence Erlbaum Associates.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13*, 3-16.

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

Brown, J. S. & Burton, R. R. (1982). An investigation of computer coaching for informal learning activities. In *Intelligent Tutoring Systems.* Ed. by Sleeman, D. & Brown, J. S. London: Academic Press.

Bonar, J. G., & Cunningham, R. (1988). Intelligent tutoring with intermediate representations. Paper presented at ITS-88. Montreal.

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

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.* NY: Wiley.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data.* Cambridge, MA: The MIT Press.

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

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.* New York: McGraw-Hill Book Company.

Goldstein, I. (1973). Elementary geometry theorem proving. MIT AI Memo 280.

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, 1.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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: Lawrence Erlbaum Associates.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of Inference, Learning, and Discovery.* Cambridge, MA: The 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: Lawrence Erlbaum Associates.

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

Koedinger, K.R., & Anderson, J.R. (1989). Perceptual chunks in geometry problem solving: A challenge to theories of skill acquisition. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Koedinger, K. R., & Anderson, J. R. (1990a). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science, 14*, 511-550.

Koedinger, K. R., & Anderson, J. R. (1990b). Theoretical and empirical motivations for the design of ANGLE: A New Geometry Learning Environment. Presented at the AAAI Spring Symposium on Knowledge-Based Environments for Learning and Teaching, Stanford University, Palo Alto, CA.

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

Larkin, J. (1988). Display-based problem solving. To appear 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 trouble shooting job. LRDC Report. Pittsburgh, PA: University of Pittsburgh.

Neves, D. M. (1978). A computer program that learns algebraic procedures. *Proceedings of the 2nd Conference on Computational Studies of Intelligence*, Toronto.

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 Chase, W. G. (Ed.), *Visual Information Processing*. New York: Academic Press.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.

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 Publishing Co.

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: Lawrence Erlbaum Associates.

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. *Proceedings CHI'88*.

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

Romberg , T.A. & NCTM Commission on Standards for School Mathematics (1987). Curriculum and evaluation standards for school mathematics. Working draft of the commission on standards for school mathematics of the National Council of Teachers of Mathematics. Reston, VA: NCTM.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (1987). Knowledge level learning in Soar. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 499-504.

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

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.

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 generation 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: Penguin.

Witkin, H. A., Oltman, P. K., Raskin, E., & Karp, S. A. (1971). *A Manual for the Embedded Figures Tests.* Palo Alto, CA: Consulting Psychologists Press.

APPENDIX A

This appendix includes the materials used in this study:

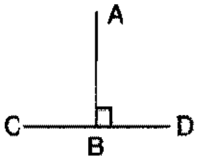
Handouts: Rule Summary Sheet, Tracking Sheet, Concept Summary Sheet (ANGLE group only), ANGLE Text (ANGLE group only), GPT Text (GPT group only).

Tests: Proof Construction A&B, Hidden Figures A&B, Truth Judgment A&B, and Proof Checking A&B.

Rule Summary Sheet

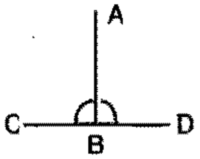
DEF-PERP:

$\overline{AB} \perp \overline{CD}$
$\iff$
rt$\angle ABC$



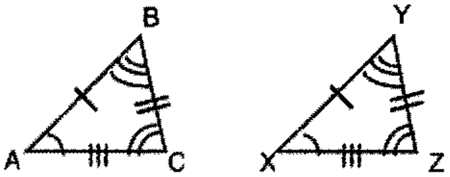
CONG-ADJ-ANGS:

$\angle ABC \cong \angle ABD$
$\iff$
$\overline{AB} \perp \overline{CD}$



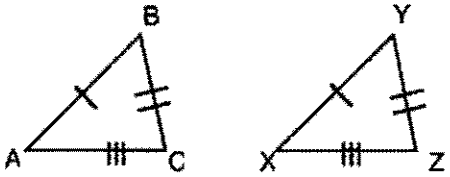
CORRES-PARTS:

$\triangle ABC \cong \triangle XYZ$
$\Longrightarrow$
$\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, $\overline{CA} \cong \overline{ZX}$,
$\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$,
$\angle CAB \cong \angle ZXY$



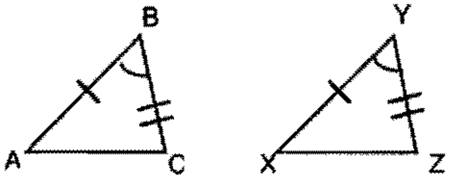
SSS:

$\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



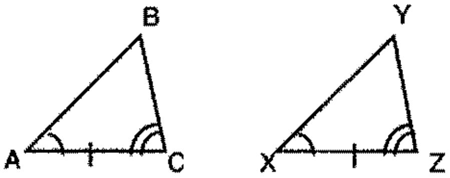
SAS:

$\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, $\angle ABC \cong \angle XYZ$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



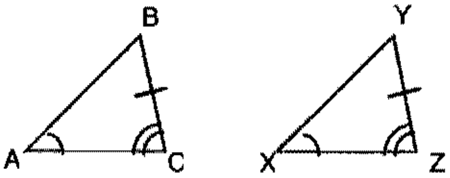
ASA:

$\angle BAC \cong \angle YXZ$, $\overline{AC} \cong \overline{XZ}$, $\angle ACB \cong \angle XZY$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



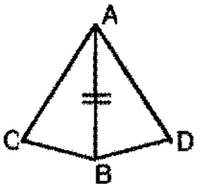
AAS:

$\angle BAC \cong \angle YXZ$, $\angle ACB \cong \angle XZY$, $\overline{BC} \cong \overline{YZ}$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



REFLEXIVE:

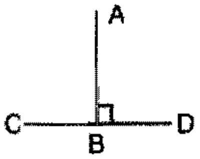
Segment $\overline{AB}$ is in the diagram
$\Longrightarrow$
$\overline{AB} \cong \overline{AB}$



More Detailed Rule Review:

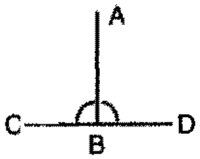
DEF-PERP:

Two lines are perpendicular, $\overline{AB} \perp \overline{CD}$,
iff[1] they form right angles, rt$\angle ABC$ and rt$\angle ABD$.



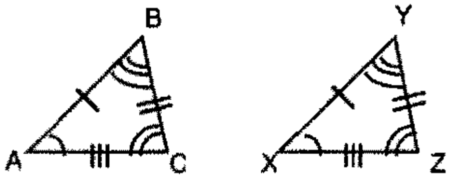
CONG-ADJ-ANGS:

Angles formed by connecting lines are congruent, $\angle ABC \cong \angle ABD$,
iff the lines are perpendicular, $\overline{AB} \perp \overline{CD}$.



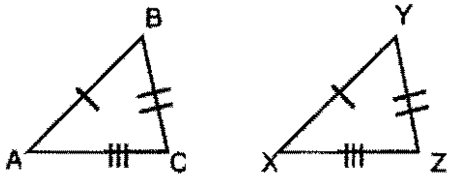
CORRES-PARTS

If two triangles are congruent, $\triangle ABC \cong \triangle XYZ$,
then all the corresponding sides are congruent: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{CA} \cong \overline{ZX}$, and all the corresponding angles are congruent: $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\angle CAB \cong \angle ZXY$.



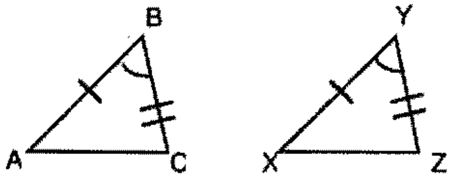
SSS:

If three sides of one triangle, $\overline{AB}$, $\overline{BC}$, and $\overline{AC}$, are congruent to the corresponding sides of another triangle, $\overline{XY}$, $\overline{YZ}$, and $\overline{XZ}$, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



SAS:

If two sides and the *included* angle of one triangle, $\overline{AB}$, $\overline{BC}$, and $\angle ABC$, are congruent to the corresponding parts, $\overline{XY}$, $\overline{YZ}$, and $\angle XYZ$, of another triangle, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\angle ABC \cong \angle XYZ$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

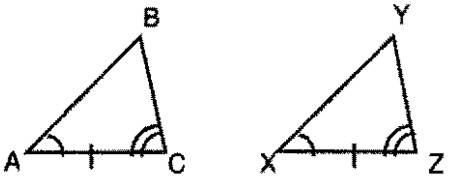


[1] *iff* means *if and only if* and indicates the rule can be applied in both directions. For example, the DEF-PERP rule is really two rules: 1) *If* two lines are perpendicular *then* they form right angles and 2) *If* two lines form right angles *then* they are perpendicular.

ASA:

If two angles and the included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{BC}$, are congruent to the corresponding parts, $\angle XYZ$, $\angle YZX$, and $\overline{YZ}$, of another triangle, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{BC} \cong \overline{YZ}$,

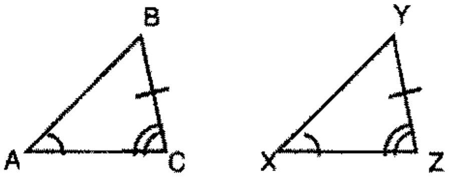
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



AAS:

If two angles and a non-included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{AB}$, are congruent to the corresponding sides of another triangle, $\angle XYZ$, $\angle YZX$, and $\overline{XY}$, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{AB} \cong \overline{XY}$,

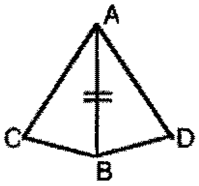
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



REFLEXIVE:

If segment $\overline{AB}$ appears in the diagram,

then $\overline{AB} \cong \overline{AB}$.



Tracking Sheet

Name:______________ Date started:____________
Id#:___

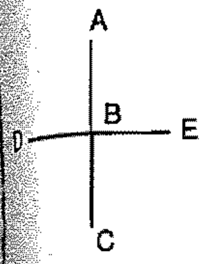
Here are the things you'll be doing as you work through the tutor. Please use this sheet to keep track of what you've done, so that when you come in on the next day you'll know where you left off. On the following page, please record the time spend with the tutor on each day and which problems you did on that day.

Please check off each problem as you do it.

1. Review DEF-PERP.
2. Do problems:
    - [ ] PROB150
    - [ ] PROB152
3. Review CONG-ADJ-ANGS.
4. Do problems:
    - [ ] N1
    - [ ] PROB151
    - [ ] P1
    - [ ] N2
5. Review CORRES-PARTS.
6. Do problem:
    - [ ] N3
7. Review SSS, SAS, ASA, and AAS.
8. Do problems:
    - [ ] PROB310
    - [ ] N4
    - [ ] N5
    - [ ] P5
    - [ ] P6
    - [ ] P7
    - [ ] P8
9. Review REFLEXIVE.
10. Do problems:
    - [ ] PROB311
    - [ ] PROB352
    - [ ] PROB353
    - [ ] N7
    - [ ] P2
    - [ ] P3
    - [ ] N16
    - [ ] N17
    - [ ] N15
    - [ ] N9
    - [ ] N8
    - [ ] N10
    - [ ] N11
    - [ ] N12
    - [ ] N13
    - [ ] N14

DAY	START TIME	FINISH TIME	TIME SPENT	PROBLEMS COMPLETED
1				
2				
3				
4				
5				
6				

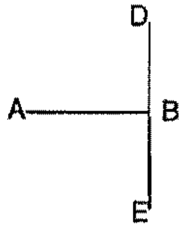
PROB150



GIVENS: $\overline{CA} \perp \overline{DE}$

GOAL: $rt\angle ABE$

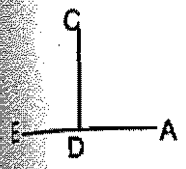
PROB151



GIVENS: $\overline{AB} \perp \overline{ED}$

GOAL: $\angle ABD \cong \angle EBA$

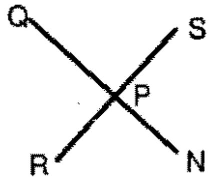
PROB152



GIVENS: $rt\angle CDA$

GOAL: $\overline{DC} \perp \overline{EA}$

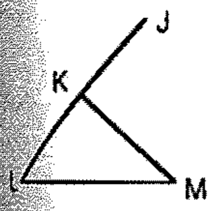
N1



GIVENS: $\angle RPQ \cong \angle SPQ$

GOAL: $\overline{RS} \cong \overline{QN}$

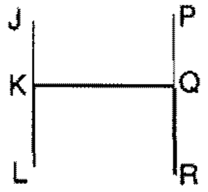
P1



GIVENS: $\angle JKM \cong \angle LKM$

GOAL: $rt\angle LKM$

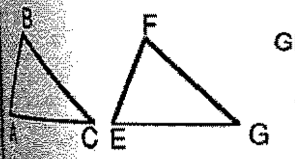
N2



GIVENS: $\overline{KQ} \perp \overline{PR}$
$rt\angle JKQ$

GOAL: $\angle JKQ \cong \angle LKQ$

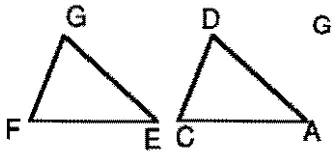
N3



GIVENS: $\triangle ABC \cong \triangle EFG$

GOAL: $\overline{AB} \cong \overline{EF}$

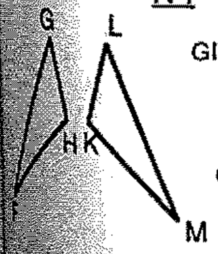
PROB310



GIVENS: $\overline{CD} \cong \overline{FG}$
$\overline{AD} \cong \overline{EG}$
$\overline{CA} \cong \overline{FE}$

GOAL: $\triangle GFE \cong \triangle DCA$

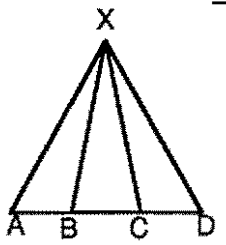
N4



GIVENS: $\angle FGH \cong \angle KLM$
$\overline{GF} \cong \overline{LM}$
$\overline{GH} \cong \overline{LK}$

GOAL: $\angle GFH \cong \angle LMK$

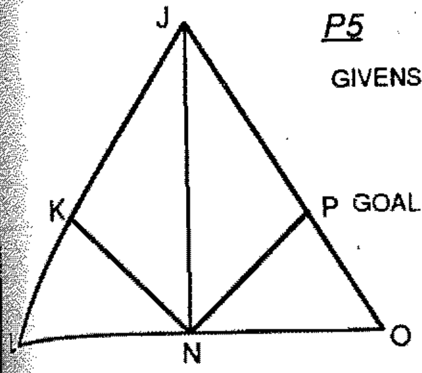
N5



GIVENS: $\angle AXB \cong \angle CXD$
$\overline{BX} \cong \overline{CX}$
$\angle ABX \cong \angle DCX$

GOAL: $\overline{XA} \cong \overline{XD}$

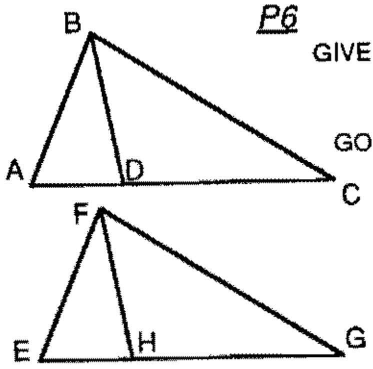
P5



GIVENS: $\triangle JKN \cong \triangle JPN$
$\angle KLN \cong \angle LOP$
$\angle LNK \cong \angle PNO$

GOAL: $\triangle LKN \cong \triangle OPN$

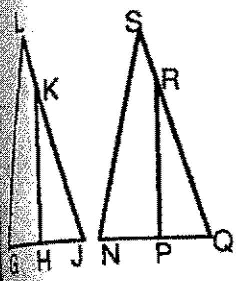
P6



GIVENS: $\triangle ABD \cong \triangle EFH$
$\triangle BCD \cong \triangle FGH$

GOAL: $\triangle ABC \cong \triangle EFG$

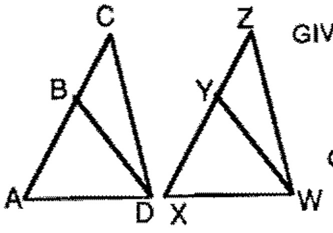
P7



GIVENS: $\angle LGH \cong \angle SNP$
$\overline{GJ} \cong \overline{NQ}$
$\triangle KHJ \cong \triangle RPQ$

GOAL: $\angle JLG \cong \angle QSN$

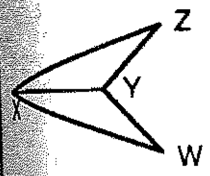
P8



GIVENS: $\triangle BCD \cong \triangle YZW$
$\overline{AC} \cong \overline{XZ}$
$\angle DBA \cong \angle WYX$

GOAL: $\angle ADB \cong \angle XWY$

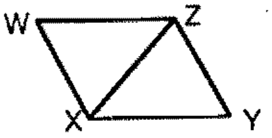
PROB311



GIVENS: $\overline{WY} \cong \overline{YZ}$
$\overline{WX} \cong \overline{XZ}$

GOAL: $\triangle WXY \cong \triangle ZXY$

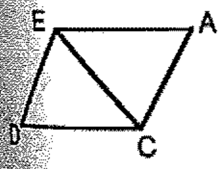
PROB352



GIVENS: $\overline{XW} \cong \overline{ZY}$
$\overline{WZ} \cong \overline{XY}$

GOAL: $\angle XYZ \cong \angle ZWX$

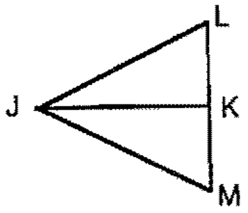
PROB353



GIVENS: $\angle CED \cong \angle ACE$
$\angle AEC \cong \angle DCE$

GOAL: $\overline{DC} \cong \overline{EA}$

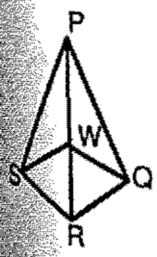
N7



GIVENS: $\overline{JL} \cong \overline{JK}$
$\angle LJK \cong \angle MJK$

GOAL: $\overline{LM} \perp \overline{JK}$

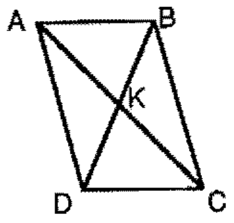
P2



GIVENS: $\triangle PSR \cong \triangle PQR$

GOAL: $\triangle WSR \cong \triangle WQR$

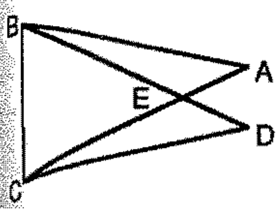
P3



GIVENS: $\triangle DAC \cong \triangle BCA$
$\angle AKD \cong \angle BKC$

GOAL: $\triangle AKD \cong \triangle CKB$

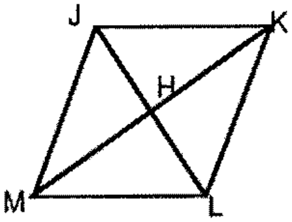
N16



GIVENS: $\angle AEB \cong \angle CED$
$\angle ABC \cong \angle DCB$
$\overline{BA} \cong \overline{CD}$

GOAL: $\triangle ABC \cong \triangle DCE$

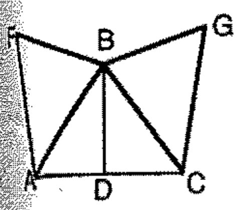
N17



GIVENS: $\overline{JH} \cong \overline{HL}$
$\overline{MH} \cong \overline{HK}$
$\angle JHK \cong \angle MHL$

GOAL: $\triangle JKL \cong \triangle LMJ$

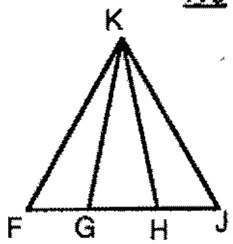
N15



GIVENS: $\overline{AC} \perp \overline{BD}$
$\overline{AD} \cong \overline{DC}$
$\angle FAB \cong \angle GCB$
$\overline{AF} \cong \overline{CG}$

GOAL: $\angle AFB \cong \angle CGB$

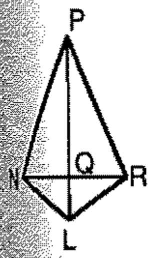
N9



GIVENS: $\overline{KG} \cong \overline{KH}$
$\overline{FH} \cong \overline{GJ}$
$\angle KGJ \cong \angle KHF$
$\angle FGK \cong \angle JHK$

GOAL: $\angle FKG \cong \angle HKJ$

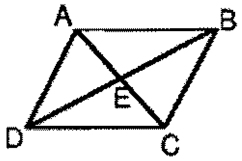
N8



GIVENS: $\text{rt}\angle LQR$
$\angle PNR \cong \angle PRN$

GOAL: $\angle NLP \cong \angle RLP$

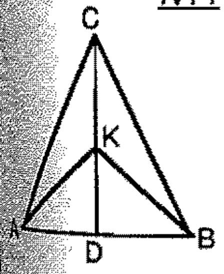
N10



GIVENS: $\overline{DE} \cong \overline{EB}$
$\angle AED \cong \angle BEC$
$\angle ADB \cong \angle CBD$

GOAL: $\overline{AB} \cong \overline{CD}$

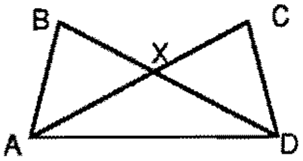
N11



GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

GOAL: $\text{rt}\angle ADC$

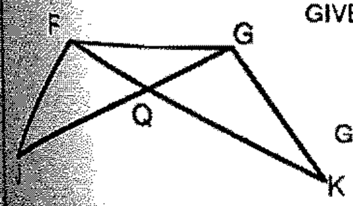
N12



GIVENS: $\angle CAD \cong \angle BDA$
$\angle BAD \cong \angle CDA$

GOAL: $\overline{AB} \cong \overline{CD}$

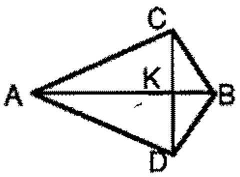
N13



GIVENS: $\overline{JQ} \cong \overline{KQ}$
$\overline{JF} \cong \overline{KG}$
$\angle JFG \cong \angle KGF$

GOAL: $\overline{FQ} \cong \overline{GQ}$

N14



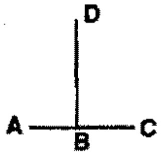
GIVENS: $\overline{AC} \cong \overline{AD}$
$\overline{CK} \cong \overline{DK}$

GOAL: $\angle BCK \cong \angle BDK$

CONCEPT SUMMARY SHEET

PERPENDICULAR-ADJACENT-ANGLES

Configuration:



Whole-statement: $\overline{AC} \perp \overline{BD}$

Part-statements:

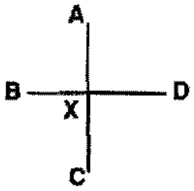
1. rt ∠ABD
2. rt ∠CBD
3. ∠ABD ≅ ∠CBD

Ways-to-prove:

DEF-PERP: {1} {2}
CONG-ADJ-ANGS: {3}

PERPENDICULAR-CROSS

Configuration:



Whole-statement: $\overline{AC} \perp \overline{BD}$

Part-statements:

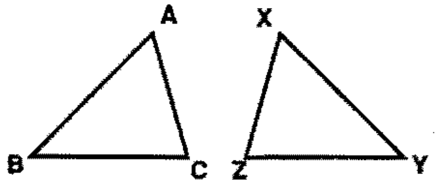
1. rt ∠AXB
2. rt ∠BXC
3. rt ∠CXD
4. rt ∠AXD
5. ∠AXB ≅ ∠BXC
6. ∠BXC ≅ ∠CXD
7. ∠CXD ≅ ∠AXD
8. ∠AXD ≅ ∠AXB

Ways-to-prove:

DEF-PERP: {1} {2} {3} {4}
CONG-ADJ-ANGS: {5} {6} {7}
{8}

CONGRUENT-TRIANGLES

Configuration:



Whole-statement: $\triangle ABC \cong \triangle XYZ$

Part-statements:

1. $\overline{AB} \cong \overline{XY}$
2. $\overline{BC} \cong \overline{YZ}$
3. $\overline{CA} \cong \overline{ZX}$
4. $\angle B \cong \angle Y$
5. $\angle C \cong \angle Z$
6. $\angle A \cong \angle X$

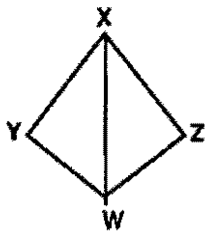
Ways-to-prove:

SSS: {1 2 3}
SAS: {1 4 2} {2 5 3} {3 6 1}
ASA: {1 4 6} {2 4 5} {3 5 6}
AAS: {1 4 5} {1 5 6} {2 4 6}
{2 5 6} {3 4 5} {3 4 6}

Other related rules: CORRES-PARTS

CONGRUENT-TRIANGLES-SHARED-SIDE

Configuration:



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

Part-statements:

1. $\overline{XY} \cong \overline{XZ}$
2. $\overline{YW} \cong \overline{ZW}$
3. $\angle Y \cong \angle Z$
4. $\angle YXW \cong \angle ZXW$
5. $\angle XWY \cong \angle XWZ$

Ways-to-prove:

SSS: {1 2}
SAS: {1 4} {2 5}
ASA: {4 5}
AAS: {3 4} {3 5}

Other related rules: CORRES-PARTS
REFLEXIVE

diagram does not appear to be a great difficulty, especially in comparison to the difficulties that arose from allowing and sometimes encouraging the explicit whole-statement encoding step.

4.1.2 Improving Tutoring Messages and Strategies

4.1.2.1 Adding Buggy Ways-to-prove. One of the problems with ANGLE's tutoring messages was that the execution feedback was quite simple and sometimes misleading. In particular, this feedback did not respond well to common bugs, like trying to prove triangles congruent using two sides and a non-included angle. These situations are captured in GPT by matching them against particular "buggy" production rules. A similar thing can be done in ANGLE by elaborating the schema representation to include buggy ways-to-prove in addition to the existing (non-buggy) ways-to-prove. Tutoring messages can then be attached to these, just like they are attached to buggy rules in GPT.

4.1.2.2 What's the Proper Role of Execution Training? One important question that was considered before the study and still remains unanswered is: what is the proper role of execution training? One could take the view that proof instruction in high school geometry should emphasize proof planning and not be too concerned about proof execution, that is, whether students get the formal details exactly right. In fact, a recent proposal for high school standard suggests a deemphasis on formal proof and more emphasis on informal proof (Romberg, 1987). However, at least for comparison sake, it seemed important that the two groups (ANGLE and GPT students) be tested on the same standard two-column format. Thus, it also seemed important to give training within ANGLE on execution as well as proof planning.

Following the cognitive model, ANGLE's feedback scheme always suggests planning moves first and only suggests execution moves once a complete plan has been found. However, consistent with the effort to make the system flexible, ANGLE allows students to integrate planning and execution. In fact, students rarely completed planning before beginning execution. One measure of this is the percent of inferences that occurred after execution began, but before planning was finished. Thus, for example, if all the planning is done first this percentage should be 0. On average, 47% of students' inferences were in this mixed stage. Students' tendency to mix planning and execution was not quite significantly correlated with post-test performance ($p = .06$). However, the pattern is that the students who began execution early also scored better on the post-test. This is probably a reflection of students' prior familiarity with the execution space and the good students' better facility with it.

Another issue relating to the role of the execution training in ANGLE, is the complexities to the interface that it added. Again such complexities presented students with a learning task that distracted them from geometry.

A couple of possibilities may be pursued with respect to this issue. One is to eliminate the execution training from ANGLE and simply focus on tutoring proof plans. It would be interesting to see how such instruction would transfer to the task of coming up with a completely detailed two-column proof. Another possibility is to have the tutor enforce the planning first approach of the cognitive model. Only after completing a plan would students be allowed to do the proof execution.

4.2 IMPROVING KNOWLEDGE MEASURES

4.2.1 Improving the Truth Judgement Test

It was hoped that the Truth Judgment test would measure students' planning abilities in a way that wouldn't be masked by lack of execution abilities. Having such a test is still desirable, however, as it turned out students did not appear to spontaneously see the relevance of proof planning to answering Truth Judgment items. Part of the reason may be due to the fact that as YES-NO type questions these items had the appearance of being easy, and thus, perhaps it did not occur to students that anything as difficult as doing proof planning would be relevant. In addition, students were given only 15 minutes to do the 17 items on this test (in fact, most finished in about 10 minutes). Four of these items could be solved with proofs about as difficult as the four proof problems on the Proof Construction test which students were given 35 minutes to solve. In other words, students really didn't have time to do proofs to help solve these problems.

A number of things can be done to encourage proof planning. First, an illustration should be given to students, prior to taking the test, of how planning a proof can help answer these questions. Second, fewer problems should be given with more time allowed for each. It should be emphasized to students that they have lots of time to think hard about each one of the items. Third, students can be asked to give a reason for their answer. In the case that they answer YES, they should provide a proof sketch. In the case that they answer CAN'T TELL, they should provide a counter example. Examples of both types of reasons should be given prior to the test.

4.2.2 The Need for a Measure of Schema Knowledge

Both GPT and ANGLE are primarily focussed on teaching the process of constructing proofs and not on teaching the declarative knowledge of the basic operators, rules in the case of GPT and schemas/concepts in the case of ANGLE. To the extent that students don't have a reasonable grasp of this basic knowledge, they are likely to have trouble. This situation is potentially more problematic in ANGLE because the units of declarative knowledge in ANGLE, the concepts: (1) are not explicitly taught in the standard curriculum and (2) are much bigger than the formal rules which are the declarative knowledge units in GPT. It seems possible that the effectiveness of ANGLE might interact with the level of students' prior knowledge of the concepts. Thus, it seems important to have a measure of this knowledge.

Such a test might be made up of items, much like the *easy* items on the Truth Judgment test, in which the student is given a diagram configuration and some facts about it, and asked if a particular conclusion follows. One type of item would test knowledge of the part-statements by providing the whole-statement as the given and a possible part-statement as the goal. Another type of item would test knowledge of the ways-to-prove by providing sets of part-statements as the givens and the whole-statement as the goal.

4.3 RESEARCH SUMMARY

This final section provides a recap of the research agenda carried out in this thesis. However, rather than simply summarizing, I've attempted to generalize the key steps and present them as a prescription for tutor design. Certainly there are other theoretically motivated routes to successful tutor design, not to mention getting there by good intuitions or serendipity. This prescription is provided merely as one possible route that may (1) be directly applied in some domains or (2) be used as a departure point for developing related approaches in other domains.

The approach can characterized in five steps:

1. Identify the *execution space.*
2. Look for *implicit planning* in verbal reports.
3. Model this implicit planning.
4. Use the model to drive tutor design.
5. *Tune* the tutor implementation.

Below I discuss the significance of each step, suggest how it might be done in general, and review how it was done in this project.

4.3.1 Identify the Execution Space

This step sets the stage for step 2 where one looks for underlying problem solving processes that are effectively hidden in current instruction. First, we need to know what aspects of the problem solving process are revealed, at least implicitly, by current instruction. This is the task of identifying the execution space. The execution space for a domain is the problem space most directly induced from the way problem solution steps are typically or conventionally written down[1]. In other words, the operators of this space correspond one-to-one with the written problem steps.

As discussed in Section 1.1, the execution space operators for geometry are the various definitions, postulates, and theorems that appear as the "reasons" in the steps of the conventional two-column proof format. The execution space operators for algebra equation solving are the various rules (e.g., You can add the same number to both sides) for manipulating equations. In physics problem solving (e.g., the kind analyzed by Larkin, et. al., 1980), the execution space operators might be the relevant physics formulas.

Another potential guide to the operators of the execution space is to look at the units of knowledge that are provided to students in their textbooks or lectures. Quite often, these units of knowledge correspond with the written problem steps. For example, the traditional geometry curriculum is organized around presenting and illustrating the very same rules that appear as reasons in two-column proof solutions. A similar situation is apparent in algebra and physics.

A straight-forward way to model problem solving in these domains is as a heuristic search in the execution space – the only trick is to find appropriate operator selection heuristics. From the perspective of a student, to the extent that the execution space provides a good characterization of skilled problem solving, his or her learning job is made easier. By definition, execution operators can be induced fairly directly from the steps of worked out examples and may be supported by verbal descriptions in textbooks and lectures. For example, consider algebra equation solving. An example worked out solution is shown in Table 4.1.

[1] Newell and Simon (1972, p. 144) refered to a "basic problem space" and gave examples of one in a number of domains. It is evident from their examples that what they meant by a basic problem space is similar to what I mean by an execution space, however, they did not explicitly define it.

Table 4.1 An worked solution in the domain of Algebra equation solving.

3x - 13	=	2(x - 3)	
3x - 13	=	2x - 6	Distribute
3x - 13 - 2x	=	- 6	x's to left side
3x - 2x	=	- 6 + 13	Num's to right side
x	=	7	

In this domain, the execution operators can be fairly directly induced from the steps in worked out examples like that in Table 4.1. This claim is supported by the fact that an early machine learning program did exactly that (Neves, 1978). In addition, the general difference-reduction heuristic turns out to be an effective means of operator selection. Because this domain independent weak-method works in this domain, learning operator selection is relatively easy.

While heuristic search in the execution space is a straight-forward candidate for modeling problem solving in a domain, it may not be the problem space that skilled problem solvers typically use in this domain. The next step is to see if it is or not.

4.3.2 Look for Implicit Planning in Verbal Reports

The purpose of this step is to identify the nature of skilled problem solving in the domain and in particular, to see if it deviates from heuristic search in the execution space. To do so, one can collect concurrent verbal reports (Ericsson & Simon, 1984) of skilled subjects solving problems in the domain. As Ericsson & Simon point out, subjects should not explain what they are doing, but merely report what they are thinking. To the extent that heuristic search in the execution space provides a good model, subjects' successive verbalizations should correspond with successive states in the execution space.

However, subjects' verbalizations could deviate from the execution space in a number of ways:

1. Multiple execution steps might be *aggregated* into a single verbalization.
2. Successive verbalizations may *skip* steps in the execution space.
3. Verbalizations may not specify an execution state in full detail, but rather only indicate some *abstract* feature(s) of it.

Regularities in such deviations suggest "thinking steps" which are not represented in the execution space. From the perspective of the student, these thinking steps are an implicit part of the planning process which, in contrast to the execution operators, cannot be directly induced from worked out examples. In other words, when there is implicit planning in the thinking of skilled problem solvers, there may be aspects of a successful problem solving method which are hidden in the traditional curriculum.

In Section 1.2, I showed that skilled geometry problem solvers skip steps in execution space (#2 above) while developing an initial proof plan. The knowledge that allows them to do so is hidden in the geometry curriculum. While it is probably possible to induce the execution operators from the steps of worked out geometry proofs in similar fashion to Neves' (1978) program for Algebra, it is not possible to apply a general weak-method like difference-reduction or means-ends analysis to effectively perform operator selection in the execution space of geometry. As the step skipping in the verbal reports suggests, successful search in geometry involves operators other than those in the execution space.

It is possible that no evidence for implicit planning is found in a domain, in other words, skilled subjects work in the execution space. This seems likely in the domain of algebra equation solving. In such a case, building a tutor for this domain may be inappropriate. Conventional instruction may be adequate. In other words, finding no implicit planning would suggest that the conventional written display of problem solving steps corresponds fairly well with the thought steps necessary for successful problem solving in that domain. In this case, well-motivated students may not have much trouble in inducing the necessary operators. A cognitively-based intelligent tutoring system (ITS) is unlikely to be much different from conventional instruction and thus, unlikely to help much. In fact, an ITS for algebra equation solving has been compared with a conventional classroom and although students learn with this tutor, they don't learn any better than students in a normal classroom (J. R. Anderson, personal communication).

4.3.3 Model this Implicit Planning

If evidence for implicit planning is found, the question becomes what is the knowledge that is responsible for the non-execution space inferences? I can offer no sweeping generalizations for how to come up with the elements for a model of implicit planning. I can only say that the diagram configuration schemas described in Section 1.3.1, evolved from an initial attempt to apply ideas about abstract planning and abstract problem spaces (Sacerdoti, 1974; Newell & Simon, 1972). In particular, the first attempt at a model was based on applying the idea of equivalence classes as a way to collapse nearby problem states into one.

In the case of verbalization types (1) and (2) above, it is possible that the implicit planning knowledge is the result of composing execution operators. In section 1.6.3, I argued against a macro-operator interpretation of the genesis of DC-schemas. In the process, a number of potentially general criteria were used to distinguish macro-operators derived from execution operators from operators that merely bear a macro-operator relationship with the execution operators. In the case of verbalization type (3), the abstract problem space ideas are likely to be relevant. For example, Newell and Simon's (p. 152, 1972) augmented problem space for crytarithmetic provides a model of such verbalizations (e.g., abstract features of states like the number must be even).

4.3.4 Use the Model to Drive Tutor Design

Once an accurate model of implicit planning has been developed the challenge is to find a way to communicate this model to students. This issue is discussed in Section 2.1. Since people seem to learn best by doing, directly communicating it to them doesn't usually help much. By design, this model contains problem solving process that are not reflected in the notation of the current curriculum. Thus, it is necessary to invent new notations which reify the previously hidden structures and processes. These notations can be the basis for interface design. For example, diagram configuration schemas became the basis for the icons in ANGLE's concept menu and for the representation of concept instances in the proof graph. In addition, the computer medium affords the possibility of inventing *novel actions* which can more directly reify the processes of the model. For example, the interface actions for selecting a schema instance from the problem diagram reify the diagram parsing process (see Section 2.3.2 and Figure 2.2).

The goal of the design of tutoring strategies and messages should be to support and articulate the problem solving method. In ANGLE, the tutoring strategy of focussing on the schema level as the proper grain size for next-step advice is intended to support the learning of the diagram configuration space (see Section 2.4.4 and Figures 2.13-15). The use of terms like "concept", "part-statement", and "ways-to-prove" is intended to articulate crucial aspects of this novel notation.

4.3.5 Tune the Tutor Implementation

While cognitive models can guide the design of the interface and tutoring components, the process of implementing these guidelines is still somewhat of a black art. Thus, it is important to test and tune the implementation. What seems right intuitively may not be effective. In addition, because of the great complexity of such systems, all possible interactions cannot be anticipated. Long periods of use by multiple students is the only way to "shakedown" possible unanticipated interactions.

Two examples from the preliminary study are most notable. First, there was the problem with bottom-out hints and more generally, with the conflict between interface flexibility and user confusion. Intuitively it had seemed that learning would be facilitated by both (1) always requiring students to enter proof steps even in the case that the tutor has told them what to do, and (2) always allowing students the flexibility to ignore hints, for example, if they had something else they wanted to do. These two notions led to the fact that students were able to ignore ANGLE's bottom-out hints. Perhaps in part because the interface was not clear enough and in part because the wording of the bottom-out hints was conceptually focussed rather than interface focussed, students ignored these hints much more often and for a longer period of time than was expected.

Second, there was the problem with whole-statement encoding. Here again, there were apparently well-motivated goals to (1) maximize the student's involvement, that is, by not doing the whole-statement encoding step for them, and (2) allow flexibility, that is, allowing students to either explicitly perform or skip this step. However, the result was, on one hand, added complexity to the interface with little or no added instructional impact and, on the other hand, an interaction within the generic hinting scheme that resulted in a confusing tutor message.

CONCLUSION

This thesis illustrates a program of basic and applied research that started by understanding the nature of a complex problem solving domain and next, applied this new understanding to the development of improved instruction for that domain. This instruction comes in the form of an intelligent tutoring system called ANGLE.

I conclude by reviewing the general contributions of this thesis as stated in the introduction:

1. A new methodology for verbal protocol analysis involving the identification of *step-skipping* with respect to the *execution space* of a domain.

In Section 1.1, I defined the execution space as the problem space that corresponds one-to-one with the steps that problem solvers conventionally write down in solving problems in that domain. I showed how one could use the execution space of a domain as the basis for analyzing verbal reports of subjects solving problems in that domain. By looking for step-skipping with respect to the execution space (see Section 1.2), one can identify *implicit planning* (i.e., thought that is not represented in the conventional notations of the domain) and take a step toward both a deeper understanding of the domain and a promising approach to improved instruction.

2. A new theory of geometry expertise (DC) that accurately describes human behavior, has an efficient computer implementation, and pulls together a number of empirical results on the nature of human expertise.

In Section 1.3, I described DC, a model of skilled geometers informal and intuitive proof planning skills. This model is based on knowledge components, diagram configuration schemas, which merge *perceptual knowledge* about important geometric categories, *conceptual knowledge* about the properties and sufficiency conditions of these categories, and *rule knowledge* of how these categories relate to the formal language of geometry. Section 1.4 established the computational power and empirical accuracy of the model. Section 1.6.2 showed how DC's perceptually-based schemas provide both a detailed explanation of experts' ability, in certain domains, to solve relatively simple problems by pure forward inferencing and an integration of the empirical results indicating experts' superior problem-state memory and their superior problem solving effectiveness.

3. A detailed characterization of the end-state of a complex learning process that challenges current learning theories and that can be used as a test-case for new learning theories.

In Section 1.6.3, I discussed how straight-forward applications of learning theories such as ACT* or Soar are unlikely to produce the regularity in knowledge organization reflected by DC's schemas. In particular, this regularity seems to exclude deductive or symbol-level learning processes which acquire expertise through the composition of execution space operators. Rather, it seems more likely that DC schemas (and perhaps perceptually-based planning schemas in general) are learned through inductive or knowledge-level learning processes. These inductive learning mechanisms would include a perceptual chunking process capable of creating ever-larger diagram configurations (percepts in general terms), and a categorization process capable of attaching part-statements (properties) and ways-to-prove (sufficiency conditions) to these percepts.

The fact that novice problem solvers exhibited DC-like step-stepping on the proof pre-test (see Section 3.3.2) provides further evidence that learning in geometry doesn't start with the execution space and then improve on this representation through composing operators. Rather, very early in the learning process, these novices appear to have acquired some non-execution operators that, though sometimes erroneous, bear a closer similarity with DC schemas than with the formal rules of geometry.

4. A theory-based approach to the design of the interface and tutoring components of an intelligent tutoring system (ANGLE).

In Chapter 2, I showed how a cognitive model of implicit planning (DC) could be translated into design specifications for the interface and tutoring components of an intelligent tutoring system. Perhaps the most important aspect of this approach is how a model of implicit planning can be used to design novel interface notations and actions. By definition, a model of implicit planning must make use of representations and processes that are not represented in the current notation of the domain. Thus, with a model of implicit planning in hand, the tutor designer can use it to guide the invention of new notations that more accurately reflect the processes of skilled problem solving. As part of a tutor's interface, these notations deliver instruction implicitly by providing students with a more cognitively meaningful way to think about the domain.

5. An initial test of the hypothesis that the development of more accurate and powerful cognitive models of problem solving can lead to major improvements in the instruction of problem solving, particularly within the context of an intelligent tutoring system.

While there have been numerous successful examples of applying cognitive science to instructional design (e.g., Anderson, et. al., 1990), these have been cases of comparing traditional (non-cognitive) instruction with cognitively-based instruction. These studies show the impact of designing instruction based on a cognitive model rather than on intuitions. The hypothesis above poses a more detailed question: Do better cognitive models lead to better instruction?

By testing this hypothesis in the medium of intelligent tutoring systems, we can have both control over the exact nature of the instruction that is delivered (which is hard to do with human teachers) and maintain some of the on-line flexibility of a human teacher (which cannot be done with text-based instruction). While the preliminary study described in Chapter 3 did not conclusively affirm or negate this hypothesis, it moves us a step closer. Despite the fact that much less effort has been put into ANGLE to this point, it is about equal in instructional effectiveness to the previously successful Geometry Proof Tutor. The hope is that by eliminating the problems with ANGLE's implementation and expanding its curriculum, we can exceed the effectiveness of GPT and confirm the hypothesis.

REFERENCES

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

Anderson, J. R. (1988). The expert module. In M. C. Polson & J. J. Richardson (Eds), *Foundations of Intelligent Tutoring Systems.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Anderson, J. R., Boyle, C. F., Corbett, A., & Lewis, M. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence*, *42*, 7-49.

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: Lawrence Erlbaum Associates.

Bloom, B. S. (1984). The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, *13*, 3-16.

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

Brown, J. S. & Burton, R. R. (1982). An investigation of computer coaching for informal learning activities. In *Intelligent Tutoring Systems.* Ed. by Sleeman, D. & Brown, J. S. London: Academic Press.

Bonar, J. G., & Cunningham, R. (1988). Intelligent tutoring with intermediate representations. Paper presented at ITS-88. Montreal.

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

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.* NY: Wiley.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol Analysis: Verbal Reports as Data.* Cambridge, MA: The MIT Press.

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

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.* New York: McGraw-Hill Book Company.

Goldstein, I. (1973). Elementary geometry theorem proving. MIT AI Memo 280.

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, 1.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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: Lawrence Erlbaum Associates.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of Inference, Learning, and Discovery.* Cambridge, MA: The 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: Lawrence Erlbaum Associates.

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

Koedinger, K.R., & Anderson, J.R. (1989). Perceptual chunks in geometry problem solving: A challenge to theories of skill acquisition. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Koedinger, K. R., & Anderson, J. R. (1990a). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science, 14*, 511-550.

Koedinger, K. R., & Anderson, J. R. (1990b). Theoretical and empirical motivations for the design of ANGLE: A New Geometry Learning Environment. Presented at the AAAI Spring Symposium on Knowledge-Based Environments for Learning and Teaching, Stanford University, Palo Alto, CA.

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

Larkin, J. (1988). Display-based problem solving. To appear 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 trouble shooting job. LRDC Report. Pittsburgh, PA: University of Pittsburgh.

Neves, D. M. (1978). A computer program that learns algebraic procedures. *Proceedings of the 2nd Conference on Computational Studies of Intelligence*, Toronto.

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 Chase, W. G. (Ed.), *Visual Information Processing*. New York: Academic Press.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA.

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 Publishing Co.

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: Lawrence Erlbaum Associates.

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. *Proceedings CHI'88.*

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

Romberg , T.A. & NCTM Commission on Standards for School Mathematics (1987). Curriculum and evaluation standards for school mathematics. Working draft of the commission on standards for school mathematics of the National Council of Teachers of Mathematics. Reston, VA: NCTM.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (1987). Knowledge level learning in Soar. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 499-504.

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

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.

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 generation 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: Penguin.

Witkin, H. A., Oltman, P. K., Raskin, E., & Karp, S. A. (1971). *A Manual for the Embedded Figures Tests.* Palo Alto, CA: Consulting Psychologists Press.

APPENDIX A

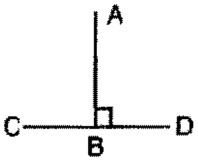
This appendix includes the materials used in this study:

Handouts: Rule Summary Sheet, Tracking Sheet, Concept Summary Sheet (ANGLE group only), ANGLE Text (ANGLE group only), GPT Text (GPT group only).

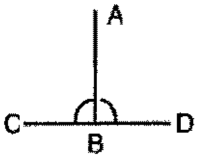
Tests: Proof Construction A&B, Hidden Figures A&B, Truth Judgment A&B, and Proof Checking A&B.

Rule Summary Sheet

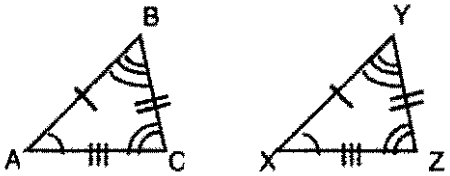
DEF-PERP: $\overline{AB} \perp \overline{CD}$
$\Longleftrightarrow$
rt$\angle ABC$



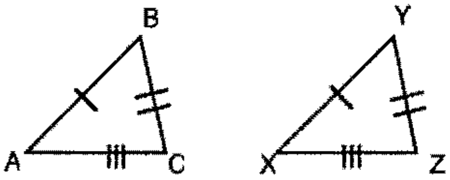
CONG-ADJ-ANGS: $\angle ABC \cong \angle ABD$
$\Longleftrightarrow$
$\overline{AB} \perp \overline{CD}$



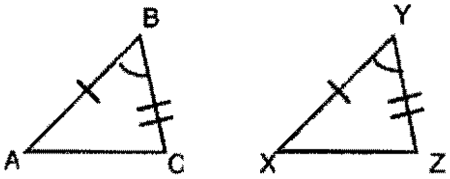
CORRES-PARTS: $\triangle ABC \cong \triangle XYZ$
$\Longrightarrow$
$\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, $\overline{CA} \cong \overline{ZX}$,
$\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$,
$\angle CAB \cong \angle ZXY$



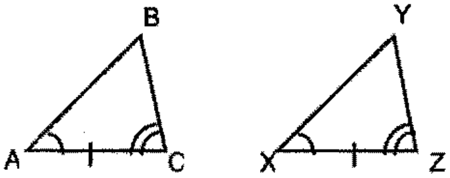
SSS: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



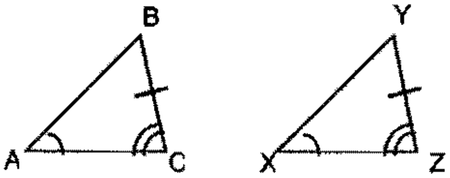
SAS: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, $\angle ABC \cong \angle XYZ$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



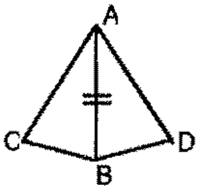
ASA: $\angle BAC \cong \angle YXZ$, $\overline{AC} \cong \overline{XZ}$, $\angle ACB \cong \angle XZY$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



AAS: $\angle BAC \cong \angle YXZ$, $\angle ACB \cong \angle XZY$, $\overline{BC} \cong \overline{YZ}$
$\Longrightarrow$
$\triangle ABC \cong \triangle XYZ$



REFLEXIVE: Segment $\overline{AB}$ is in the diagram
$\Longrightarrow$
$\overline{AB} \cong \overline{AB}$

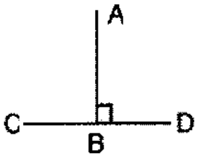


More Detailed Rule Review:

DEF-PERP:

Two lines are perpendicular, $\overline{AB} \perp \overline{CD}$,

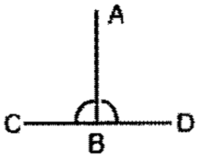
iff[1] they form right angles, rt$\angle ABC$ and rt$\angle ABD$.



CONG-ADJ-ANGS:

Angles formed by connecting lines are congruent, $\angle ABC \cong \angle ABD$,

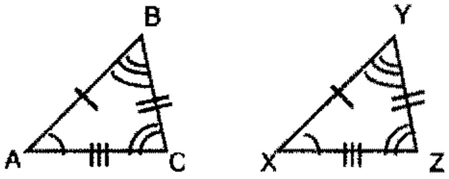
iff the lines are perpendicular, $\overline{AB} \perp \overline{CD}$.



CORRES-PARTS

If two triangles are congruent, $\triangle ABC \cong \triangle XYZ$,

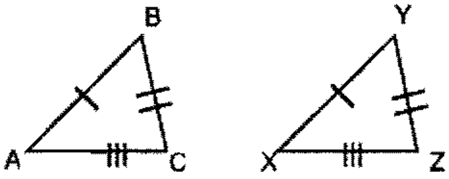
then all the corresponding sides are congruent: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{CA} \cong \overline{ZX}$, and all the corresponding angles are congruent: $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\angle CAB \cong \angle ZXY$.



SSS:

If three sides of one triangle, $\overline{AB}$, $\overline{BC}$, and $\overline{AC}$, are congruent to the corresponding sides of another triangle, $\overline{XY}$, $\overline{YZ}$, and $\overline{XZ}$, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$,

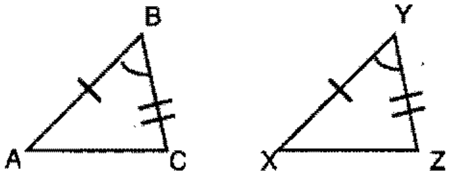
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



SAS:

If two sides and the *included* angle of one triangle, $\overline{AB}$, $\overline{BC}$, and $\angle ABC$, are congruent to the corresponding parts, $\overline{XY}$, $\overline{YZ}$, and $\angle XYZ$, of another triangle, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\angle ABC \cong \angle XYZ$,

then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

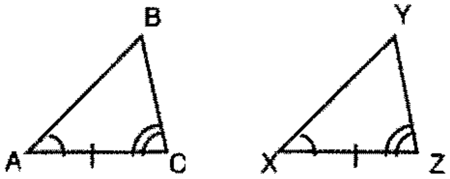


[1] *iff* means *if and only if* and indicates the rule can be applied in both directions. For example, the DEF-PERP rule is really two rules: 1) *If* two lines are perpendicular *then* they form right angles and 2) *If* two lines form right angles *then* they are perpendicular.

ASA:

If two angles and the included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{BC}$, are congruent to the corresponding parts, $\angle XYZ$, $\angle YZX$, and $\overline{YZ}$, of another triangle, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{BC} \cong \overline{YZ}$,

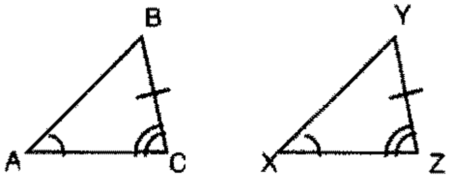
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



AAS:

If two angles and a non-included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{AB}$, are congruent to the corresponding sides of another triangle, $\angle XYZ$, $\angle YZX$, and $\overline{XY}$, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{AB} \cong \overline{XY}$,

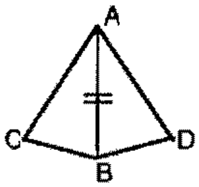
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.



REFLEXIVE:

If segment $\overline{AB}$ appears in the diagram,

then $\overline{AB} \cong \overline{AB}$.



Tracking Sheet

Name:_______________ Date started:_____________
Id#:___

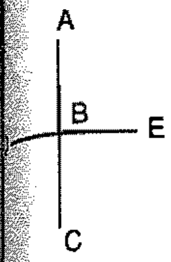
Here are the things you'll be doing as you work through the tutor. Please use this sheet to keep track of what you've done, so that when you come in on the next day you'll know where you left off. On the following page, please record the time spend with the tutor on each day and which problems you did on that day.

Please check off each problem as you do it.

1. Review DEF-PERP.
2. Do problems:
    [ ] PROB150
    [ ] PROB152
3. Review CONG-ADJ-ANGS.
4. Do problems:
    [ ] N1
    [ ] PROB151
    [ ] P1
    [ ] N2
5. Review CORRES-PARTS.
6. Do problem:
    [ ] N3
7. Review SSS, SAS, ASA, and AAS.
8. Do problems:
    [ ] PROB310
    [ ] N4
    [ ] N5
    [ ] P5
    [ ] P6
    [ ] P7
    [ ] P8
9. Review REFLEXIVE.
10. Do problems:
    [ ] PROB311
    [ ] PROB352
    [ ] PROB353
    [ ] N7
    [ ] P2
    [ ] P3
    [ ] N16
    [ ] N17
    [ ] N15
    [ ] N9
    [ ] N8
    [ ] N10
    [ ] N11
    [ ] N12
    [ ] N13
    [ ] N14

DAY	START TIME	FINISH TIME	TIME SPENT	PROBLEMS COMPLETED
1				
2				
3				
4				
5				
6				

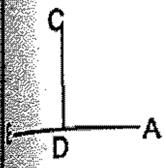
PROB150



GIVENS: $\overline{CA} \perp \overline{DE}$

GOAL: rt$\angle ABE$

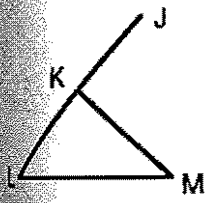
PROB152



GIVENS: rt$\angle CDA$

GOAL: $\overline{DC} \perp \overline{EA}$

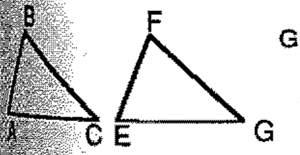
P1



GIVENS: $\angle JKM \cong \angle LKM$

GOAL: rt$\angle LKM$

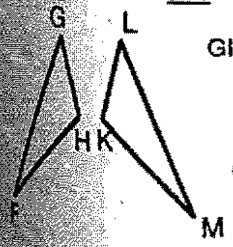
N3



GIVENS: $\triangle ABC \cong \triangle EFG$

GOAL: $\overline{AB} \cong \overline{EF}$

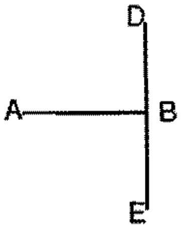
N4



GIVENS: $\angle FGH \cong \angle KLM$
$\overline{GF} \cong \overline{LM}$
$\overline{GH} \cong \overline{LK}$

GOAL: $\angle GFH \cong \angle LMK$

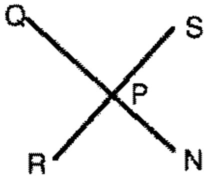
PROB151



GIVENS: $\overline{AB} \perp \overline{ED}$

GOAL: $\angle ABD \cong \angle EBA$

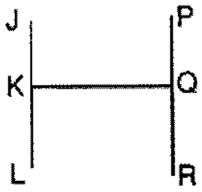
N1



GIVENS: $\angle RPQ \cong \angle SPQ$

GOAL: $\overline{RS} \cong \overline{QN}$

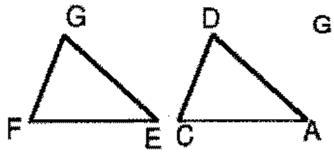
N2



GIVENS: $\overline{KQ} \perp \overline{PR}$
rt$\angle JKQ$

GOAL: $\angle JKQ \cong \angle LKQ$

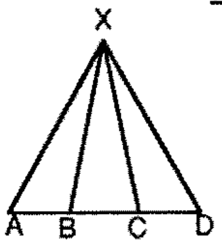
PROB310



GIVENS: $\overline{CD} \cong \overline{FG}$
$\overline{AD} \cong \overline{EG}$
$\overline{CA} \cong \overline{FE}$

GOAL: $\triangle GFE \cong \triangle DCA$

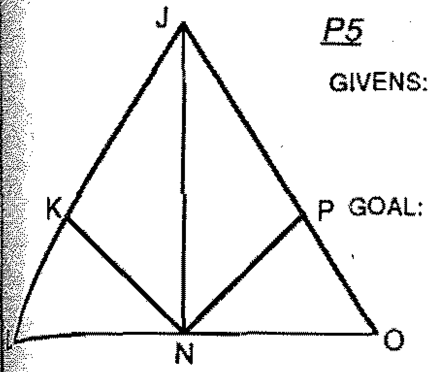
N5



GIVENS: $\angle AXB \cong \angle CXD$
$\overline{BX} \cong \overline{CX}$
$\angle ABX \cong \angle DCX$

GOAL: $\overline{XA} \cong \overline{XD}$

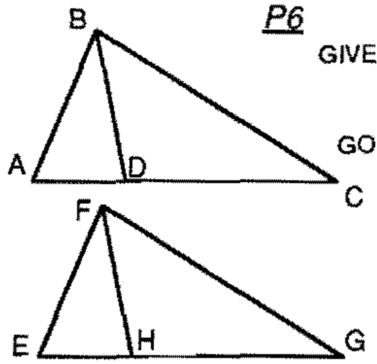
P5



GIVENS: $\triangle JKN \cong \triangle JPN$
$\angle KLN \cong \angle LOP$
$\angle LNK \cong \angle PNO$

GOAL: $\triangle LKN \cong \triangle OPN$

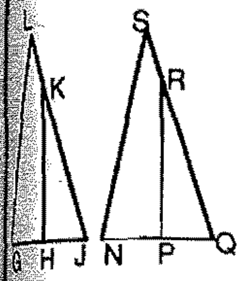
P6



GIVENS: $\triangle ABD \cong \triangle EFH$
$\triangle BCD \cong \triangle FGH$

GOAL: $\triangle ABC \cong \triangle EFG$

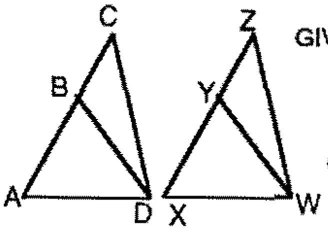
P7



GIVENS: $\angle LGH \cong \angle SNP$
$\overline{GJ} \cong \overline{NQ}$
$\triangle KHJ \cong \triangle RPQ$

GOAL: $\angle JLG \cong \angle QSN$

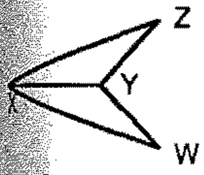
P8



GIVENS: $\triangle BCD \cong \triangle YZW$
$\overline{AC} \cong \overline{XZ}$
$\angle DBA \cong \angle WYX$

GOAL: $\angle ADB \cong \angle XWY$

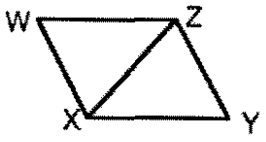
PROB311



GIVENS: $\overline{WY} \cong \overline{YZ}$
$\overline{WX} \cong \overline{XZ}$

GOAL: $\triangle WXY \cong \triangle ZXY$

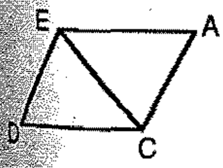
PROB352



GIVENS: $\overline{XW} \cong \overline{ZY}$
$\overline{WZ} \cong \overline{XY}$

GOAL: $\angle XYZ \cong \angle ZWX$

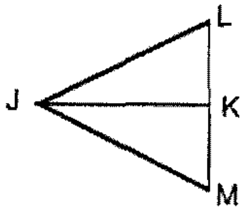
PROB353



GIVENS: $\angle CED \cong \angle ACE$
$\angle AEC \cong \angle DCE$

GOAL: $\overline{DC} \cong \overline{EA}$

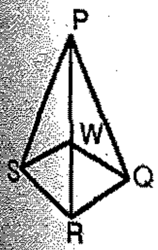
N7



GIVENS: $\overline{JL} \cong \overline{JK}$
$\angle LJK \cong \angle MJK$

GOAL: $\overline{LM} \perp \overline{JK}$

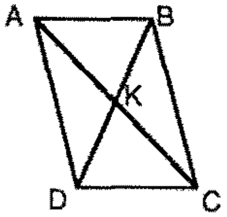
P2



GIVENS: $\triangle PSR \cong \triangle PQR$

GOAL: $\triangle WSR \cong \triangle WQR$

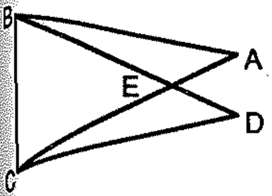
P3



GIVENS: $\triangle DAC \cong \triangle BCA$
$\angle AKD \cong \angle BKC$

GOAL: $\triangle AKD \cong \triangle CKB$

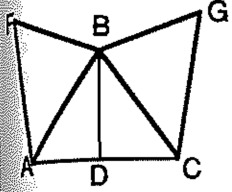
N16



GIVENS: $\angle AEB \cong \angle CED$
$\angle ABC \cong \angle DCB$
$\overline{BA} \cong \overline{CD}$

GOAL: $\triangle ABC \cong \triangle DCE$

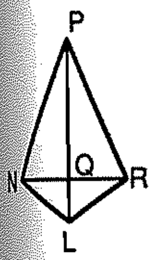
N15



GIVENS: $\overline{AC} \perp \overline{BD}$
$\overline{AD} \cong \overline{DC}$
$\angle FAB \cong \angle GCB$
$\overline{AF} \cong \overline{CG}$

GOAL: $\angle AFB \cong \angle CGB$

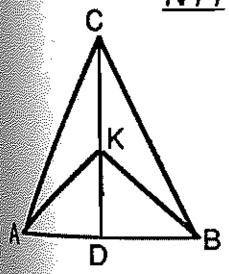
N8



GIVENS: rt$\angle LQR$
$\angle PNR \cong \angle PRN$

GOAL: $\angle NLP \cong \angle RLP$

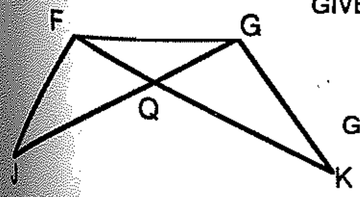
N11



GIVENS: $\overline{AC} \cong \overline{BC}$
$\overline{AK} \cong \overline{BK}$

GOAL: rt$\angle ADC$

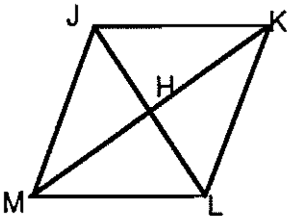
N13



GIVENS: $\overline{JQ} \cong \overline{KQ}$
$\overline{JF} \cong \overline{KG}$
$\angle JFG \cong \angle KGF$

GOAL: $\overline{FQ} \cong \overline{GQ}$

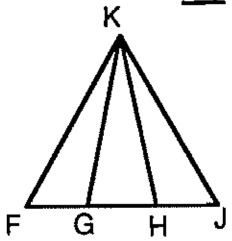
N17



GIVENS: $\overline{JH} \cong \overline{HL}$
$\overline{MH} \cong \overline{HK}$
$\angle JHK \cong \angle MHL$

GOAL: $\triangle JKL \cong \triangle LMJ$

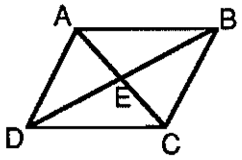
N9



GIVENS: $\overline{KG} \cong \overline{KH}$
$\overline{FH} \cong \overline{GJ}$
$\angle KGJ \cong \angle KHF$
$\angle FGK \cong \angle JHK$

GOAL: $\angle FKG \cong \angle HKJ$

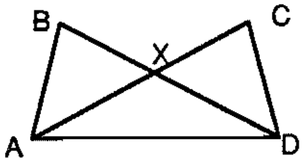
N10



GIVENS: $\overline{DE} \cong \overline{EB}$
$\angle AED \cong \angle BEC$
$\angle ADB \cong \angle CBD$

GOAL: $\overline{AB} \cong \overline{CD}$

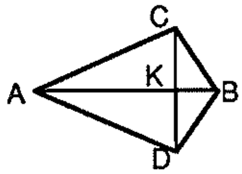
N12



GIVENS: $\angle CAD \cong \angle BDA$
$\angle BAD \cong \angle CDA$

GOAL: $\overline{AB} \cong \overline{CD}$

N14



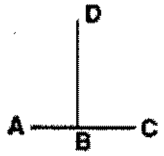
GIVENS: $\overline{AC} \cong \overline{AD}$
$\overline{CK} \cong \overline{DK}$

GOAL: $\angle BCK \cong \angle BDK$

CONCEPT SUMMARY SHEET

PERPENDICULAR-ADJACENT-ANGLES

Configuration:



Whole-statement: $\overline{AC} \perp \overline{BD}$

Part-statements:

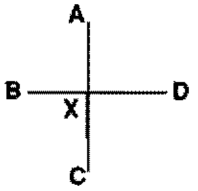
1. rt ∠ABD
2. rt ∠CBD
3. ∠ABD ≅ ∠CBD

Ways-to-prove:

DEF-PERP: {1} {2}
CONG-ADJ-ANGS: {3}

PERPENDICULAR-CROSS

Configuration:



Whole-statement: $\overline{AC} \perp \overline{BD}$

Part-statements:

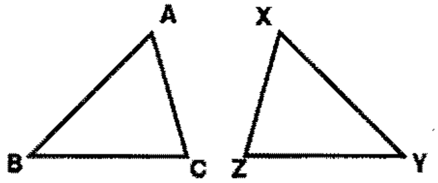
1. rt ∠AXB
2. rt ∠BXC
3. rt ∠CXD
4. rt ∠AXD
5. ∠AXB ≅ ∠BXC
6. ∠BXC ≅ ∠CXD
7. ∠CXD ≅ ∠AXD
8. ∠AXD ≅ ∠AXB

Ways-to-prove:

DEF-PERP: {1} {2} {3} {4}
CONG-ADJ-ANGS: {5} {6} {7} {8}

CONGRUENT-TRIANGLES

Configuration:



Whole-statement: $\triangle ABC \cong \triangle XYZ$

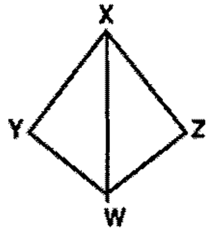
Part-statements:
1. $\overline{AB} \cong \overline{XY}$
2. $\overline{BC} \cong \overline{YZ}$
3. $\overline{CA} \cong \overline{ZX}$
4. $\angle B \cong \angle Y$
5. $\angle C \cong \angle Z$
6. $\angle A \cong \angle X$

Ways-to-prove:
SSS: {1 2 3}
SAS: {1 4 2} {2 5 3} {3 6 1}
ASA: {1 4 6} {2 4 5} {3 5 6}
AAS: {1 4 5} {1 5 6} {2 4 6}
{2 5 6} {3 4 5} {3 4 6}

Other related rules: CORRES-PARTS

CONGRUENT-TRIANGLES-SHARED-SIDE

Configuration:



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

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

Ways-to-prove:
SSS: {1 2}
SAS: {1 4} {2 5}
ASA: {4 5}
AAS: {3 4} {3 5}

Other related rules: CORRES-PARTS
REFLEXIVE

ANGLE Tutor Text

You'll be working with a computerized tutoring system called ANGLE for **A N**ew **G**eometry **L**earning **E**nvironment. This text describes how to use ANGLE to solve geometry proof problems. In ANGLE, you construct proofs in two phases. First you work out a *conceptual plan* for the proof leaving out the picky details. And then in the second phase you *fill in the details* to make your proof complete and rigorous.

Using ANGLE to Solve a Proof Problem

This section will take you step-by-step through the solution of a problem. We'll be doing the first problem on your *Tracking Sheet*, PROB150.

Constructing a Conceptual Plan

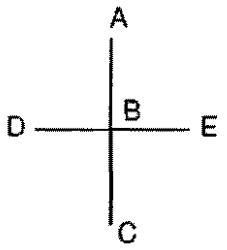
You build a conceptual plan using concepts like the ones shown on the *Concept Summary Sheet* you should have received. Look on the left of the ANGLE screen and notice the list or *menu* of pictures. Four of these pictures are the same as the pictures (configurations) on the Concept Summary Sheet. Below we'll say more about concepts, but for now we'll get you started on PROB150.

Here's what PROB150 looks like in the usual notation:

PROB150:

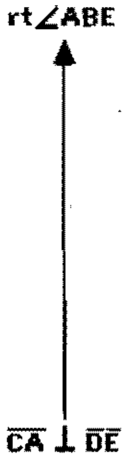
Given: $\overline{CA} \perp \overline{DE}$

Goal: $\mathrm{rt}\angle ABE$



When the problem appears on the screen, you'll notice it looks somewhat different. The problem givens are on the bottom of the screen and the goal is on the top. To construct a proof plan, you need to find one or more concepts that link the givens to the goal. The first few problems you'll do are relatively easy – involving only one concept. The concept you need for PROB150 is the PERPENDICULAR-CROSS concept (the second one on the Concept Summary Sheet). Notice how the configuration of the PERPENDICULAR-CROSS concept and the diagram for PROB150 look very much the same. This is your clue that this concept may be useful for solving the problem.

Whenever you are given a *whole-statement* of a concept, you can prove any of the *part-statements* of it. Look at the PERPENDICULAR-CROSS concept on your Concept Summary Sheet and compare it to PROB150. Notice that the given $\overline{CA} \perp \overline{DE}$ corresponds with the whole-statement of the PERPENDICULAR-CROSS concept and $\mathrm{rt}\angle ABE$ is a part-statement. Thus, you can prove $\mathrm{rt}\angle ABE$ directly from the given. In other words, the plan for PROB150 is short, you simply want to prove $\mathrm{rt}\angle ABE$ using $\overline{CA} \perp \overline{DE}$. Here's how the plan will look when you are done:



To prove a statement in ANGLE requires two easy steps. First, you indicate that $rt\angle ABE$ is the statement you want to prove. Next, you indicate $\overline{CA} \perp \overline{DE}$ as the reason. Here's how:

Justifying:

- Move the mouse pointer to "Justify" in the bottom menu on the left. (This is the *action menu.*) Click the button and "Justify" should highlight – if it doesn't, try again.
- Now, indicate you want to prove $rt\angle ABE$ by moving the mouse to $rt\angle ABE$ and clicking the button. The statement should highlight. If it didn't, try it again, but make sure the tip of mouse pointer is on $rt\angle ABE$.
- Notice that the *action menu* has changed: "Select Reasons" is now highlighted instead of "Justify".

Selecting Reasons:

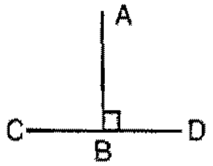
- Mouse-click on $\overline{CA} \perp \overline{DE}$. A line will be drawn from $\overline{CA} \perp \overline{DE}$ to $rt\angle ABE$.
- If you click on $\overline{CA} \perp \overline{DE}$ a second time, the line will disappear. Try it. This is what you do if you make a mistake. Now, click on it once more to get the line back.
- When you have the line connected, mouse-click on "DONE" in the menu underneath the diagram.
- (The "ABORT" option can be used if you decide you want to quit a step you are working on.)

ANGLE should now tell you that you have "a complete proof plan" and that you should "fill in the details". To get rid of the message window that appears, mouse-click on the OK button.

Filling in the Details

The *Rule Summary Sheet* provides information you'll need in filling in the details. You'll need the rule DEF-PERP to finish up PROB150. Here is a statement of the rule:

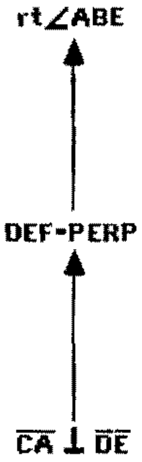
DEF-PERP:



Two lines are perpendicular, $\overline{AB} \perp \overline{CD}$,
if and only if they form a right angle $\text{rt}\angle ABD$.

This rule can be used to make a detailed or "rigorous" connection between a perpendicular line statement, like $\overline{CA} \perp \overline{DE}$, and a right angle statement, like $\text{rt}\angle ABE$. The phrase *if and only if* in the rule indicates that it can be used in either direction: *left-to-right* to go from perpendicular lines to a right angle or *right-to-left* to go from a right angle to perpendicular lines. This is symbolized by the double arrows "<==>" on the Rule Summary Sheet.

In PROB150, you'll use **DEF-PERP** in the left-to-right direction to go from the given perpendicular lines statement to the right angle statement in the goal. You want to insert the rule between these two statements so that your final proof will look like this:



Here's how you add this rule to your proof:

Entering a Rule:

- Mouse-click on "Rule" in the menu to the left.
- A rule menu will appear. Mouse-click on **DEF-PERP** in the menu and a rule will be created to the right.
- Mouse-click on "Exit" at the top of the rule menu.

Moving:

- Mouse-click on **DEF-PERP** and keep the mouse button down.
- With the mouse button down "drag" **DEF-PERP** to the place where you want it by moving the mouse. Place it on top of the line between $\overline{CA} \perp \overline{DE}$ and $\text{rt}\angle ABE$ and then let go of the mouse button.

Inserting:

- Make sure **DEF-PERP** is on top of the line between $\overline{CA} \perp \overline{DE}$ and **rt∠ABE**.
- Mouse-click on "Insert" in the action menu.
- Mouse-click on **DEF-PERP**. It should highlight.
- Mouse-click on "DONE".

If your proof is correct and fully detailed, ANGLE will tell you that your proof is "complete and rigorous". Notice that the lines have thickened. The thinner lines indicate *planning* steps, that is, steps with details left out. The thicker lines indicate detailed steps which are finished.

Get rid of the "good job" message by mouse-clicking on the "OK" button.

Now you should start problem PROB151. Here's how you start a new problem:

Selecting a Problem:

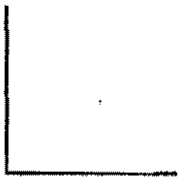
- Move the mouse to the top left corner of the screen to where you see the word "Problem". Put the mouse arrow on Problem and hold down the mouse button. The words "Load problem" should appear just below Problem.
- Keep the mouse button down and move to Load Problem. It should highlight. When it does let go of the mouse button.
- A list of problems will appear, including N1, N10, N11, ... If you don't see this problem list at this time, try the first two steps again. PROB151 does not appear on the screen, because it's down a little lower on the list. To move it into view, click the mouse button on top of the downward pointing arrow. If you went too far, click on the upward pointing arrow.
- Once you see PROB151 move the mouse to it and click the button *twice in a row* fast. This is called *double-clicking*.

It will take a little while for the problem to come up. Read on while you wait.

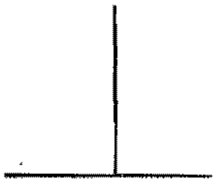
Concept Part-statements

The Properties of Perpendicular Lines

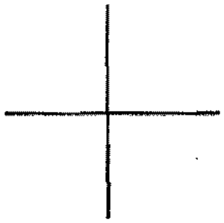
If someone asks you to draw two lines, you can draw them in just about any way you please. However, if someone asks to draw two *perpendicular* lines, you must draw them in certain way, that is, so that they form a right angle. When we say that two lines are perpendicular, we are indicating a particular arrangement or *configuration* of lines not just any arrangement. Basically, perpendicular lines come in three possible configurations depending on how the two lines meet.



RIGHT-ANGLE



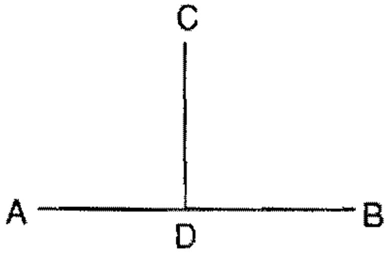
PERPENDICULAR-ADJACENT-ANGLES



PERPENDICULAR-CROSS

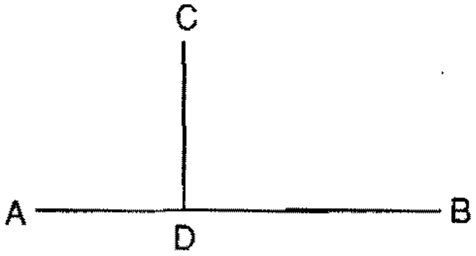
As you are introduced new concepts in this text, you'll be asked to consider the following question: What does the concept tell us about the angles and segments that are a part of the configuration?

For the concept of perpendicularity we ask: What does the fact that two lines are perpendicular tell us about the angles and segments that are formed by the two lines? Let's look at the PERPENDICULAR-ADJACENT-ANGLES configuration first. Say we know that in the following picture $\overline{AB}$ is perpendicular to $\overline{CD}$, in other words, $\overline{AB} \perp \overline{CD}$.



What does it tell us, if anything, about angles $\angle ADC$ and $\angle BDC$ and about segments $\overline{AB}$, $\overline{AD}$, $\overline{DB}$, and $\overline{CD}$? We already know that perpendicular lines form right angles. So, we know these two facts about the configuration: 1) rt$\angle ADC$ and 2) rt$\angle BDC$. Since both angles equal $90°$, it is clear they are equal to each other. Thus, we can add one more fact to our list: 3) $\angle ADC \cong \angle BDC$.

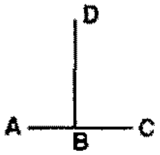
What can we say about the segments in the diagram above, given that we know $\overline{AB} \perp \overline{CD}$? Basically, nothing. Knowing that the lines are perpendicular tells us nothing about the sizes of the segments, nor whether the sizes are related. For example, just because $\overline{AB} \perp \overline{CD}$ we don't know, for example, that $\overline{AD} = \overline{DB}$. After all the diagram could look like this



and it would still be true that $\overline{AB} \perp \overline{CD}$, however, clearly its not true that $\overline{AD} = \overline{DB}$.

In summary, PERPENDICULAR-ADJACENT-ANGLES have 3 properties or *part-statements* as is shown below and on your Concept Summary Sheet.

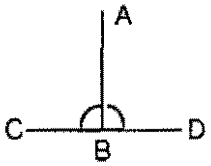
PERPENDICULAR-ADJACENT-ANGLES

Configuration:	
Whole-statement:	$\overline{AC} \perp \overline{BD}$
Part-statements:	1. rt $\angle ABD$ 2. rt $\angle CBD$ 3. $\angle ABD \cong \angle CBD$
Ways-to-prove:	{1} {2} {3}

Now let's do PROB151. This problem is very much like PROB150, but this time the PERPENDICULAR-ADJACENT-ANGLES concept is involved and you'll be proving a different part-statement. Since the goal $\angle ABD \cong \angle EBA$ is a part-statement of $\overline{AB} \perp \overline{ED}$, you can prove it directly as in PROB150. If you don't remember how to do this, refer back to the descriptions of *Justifying* and *Selecting Reasons* above.

After you have a plan, you need to fill in the details. A different rule is involved this time. It's called Congruent Adjacent Angles, which we abbreviate **CONG-ADJ-ANGS**.

CONG-ADJ-ANGS:



Angles formed by connecting lines are congruent, $\angle ABC \cong \angle ABD$, *if and only if* the lines are perpendicular, $\overline{AB} \perp \overline{CD}$.

This rule can be used to make a detailed or "rigorous" connection between a perpendicular line statement, like $\overline{AB} \perp \overline{ED}$, and an angle congruence statement, like $\angle ABD \cong \angle EBA$. Notice that this rule is also an "if and only if" rule. You can use it in the *left-to-right* direction to get from congruent adjacent angles to perpendicular lines or you can use it in the *right-to-left* direction to get from perpendicular lines to prove the adjacent angles are congruent. In PROB151, you are using it in the right-to-left direction.

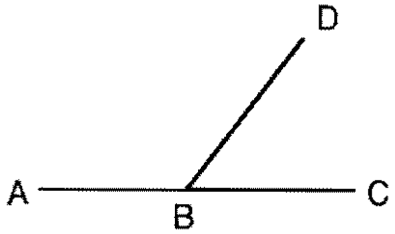
You should now insert **CONG-ADJ-ANGS** into your proof. If you don't remember how, refer to the *Entering a Rule*, *Moving*, and *Inserting* directions above.

[ ] *If you haven't already, finish problem PROB151.*
When you are done, choose problem PROB152 and read on before doing it. If you don't remember how to choose a problem, reread the directions for *Selecting a Problem.*

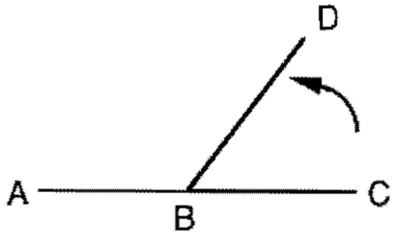
Concept Ways-to-Prove

The previous section discussed the three properties or *part-statements* of the PERPENDICULAR-ADJACENT-ANGLES concept. In this section, we address the question: How many of these part-statements or properties do you need to know to prove two lines are perpendicular?

Look at the following diagram.



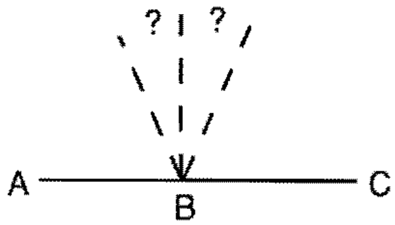
Right now ∠**CBD** is smaller than ∠**ABD**. But consider what happens if line $\overline{\mathbf{BD}}$ is moved:



Try to answer these questions before going on:

1. What happens to the sizes of the two angles?

2. If line $\overline{\mathbf{BD}}$ is moved so that ∠**ABD** and ∠**CBD** are the same size, what will line $\overline{\mathbf{BD}}$ look like? Will it be tilted to the right, tilted to the left, or straight up and down (vertical)? Will it be perpendicular to $\overline{\mathbf{AC}}$?



3. If line $\overline{BD}$ is moved so that $\angle CBD$ becomes a right angle (= 90°), what will line $\overline{BD}$ look like? Will it be tilted to the right, tilted to the left, or straight up and down (vertical)? Will it be perpendicular to $\overline{AC}$?

4. What properties of the diagram above do you need to be told in order to know that $\overline{AC} \perp \overline{BD}$? Do you need to be told all three properties: rt$\angle CBD$, rt$\angle ABD$, and $\angle CBD \cong \angle ABD$? Or is it enough to be told just two of these? Or is it enough to be told only one?

To answer question 1, you should notice that as $\overline{BD}$ moves, $\angle CBD$ gets larger and $\angle ABD$ gets smaller. At some point $\angle CBD$, which started out smaller, will become equal to $\angle ABD$, which started out larger. At the point where they become equal, $\overline{BD}$ will be vertical – if $\overline{BD}$ were tilted to the right $\angle CBD$ would be smaller than $\angle ABD$, if $\overline{BD}$ were tilted to the left $\angle CBD$ would be larger than $\angle ABD$. In other words, as a result of making $\angle CBD \cong \angle ABD$, lines $\overline{BD}$ and $\overline{AC}$ become perpendicular. This is the answer to question 2 and a clue for question 4.

Question 3 should be easy. If we make $\angle CBD$ a right angle, $\overline{BD}$ will be vertical and so, we'll know that $\overline{AC} \perp \overline{BD}$. This answer should also help you with question 4. If you are given all three properties rt$\angle CBD$, rt$\angle ABD$, and $\angle CBD \cong \angle ABD$, clearly you could prove $\overline{AC} \perp \overline{BD}$. But, simply being told *one* of these is enough to prove that $\overline{AC} \perp \overline{BD}$. In question 3, we saw that being told rt$\angle CBD$ is also enough. For the same reason, being told rt$\angle ABD$ is enough too. Lastly, from question 2, we know that $\angle CBD \cong \angle ABD$ is also enough.

Summary

To summarize, you can use any one of the three part-statements of PERPENDICULAR-ADJACENT-ANGLES in order to prove the whole-statement. This fact is indicated in the *ways-to-prove* of the concept (see your Concept Summary Sheet). The {1} in the ways-to-prove of PERPENDICULAR-ADJACENT-ANGLES indicates that you can prove $\overline{AC} \perp \overline{BD}$ if you know rt $\angle ABD$. A {1 2} in the ways-to-prove of PERPENDICULAR-ADJACENT-ANGLES would indicate that you need both right angles, rt$\angle ABD$ and rt $\angle CBD$, to prove $\overline{AC} \perp \overline{BD}$. But, as we saw above, you just need to know one of them.

Back to Doing Problems

Look at the goal of PROB152, $\overline{DC} \perp \overline{EA}$, and notice that it is a whole-statement of PERPENDICULAR-ADJACENT-ANGLES. Whenever you want to prove the whole-statement of a concept, you should look at the concepts *ways-to-prove*. Since the given statement rt$\angle CBD$ is way-to-prove of $\overline{DC} \perp \overline{EA}$, you can justify it directly from this given. Do this as described above in *Justifying* and *Selecting Reasons.*

To fill in the details, you need to find a rule which gets you from a right angle to perpendicular lines. Look on your Rule Summary Sheet. What rule will work? **CONG-ADJ-ANGS** doesn't work because it connects perpendicular lines and an angle congruence statement, however, **DEF-PERP** does work. Recall that **DEF-PERP** can be used in either direction: you can prove right angles from perpendicular lines or, as in this problem, perpendicular lines from right angles.

Insert **DEF-PERP** into your proof. If you forgot how, see the sections above on *Entering a Rule*, *Moving*, and *Inserting*.

[ ] *Finish PROB152.*
Choose problem N1 and read on.

Again, problem N1 has a short plan – you can get the goal directly from the given. This time the problem diagram contains a PERPENDICULAR-CROSS configuration and a different one of the ways-to-prove is involved. Notice that the part-statements and the ways-to-prove of the PERPENDICULAR-CROSS concept are essentially the same as the part-statements and ways-to-prove of the PERPENDICULAR-ADJACENT-ANGLES concept.

[ ] *Do N1 now.*
Choose problem P1.

This is your first two step problem. When the problem comes up, try proving the goal directly from the given. ANGLE will tell you that you can only prove **rt∠MKL** with a concept (or the whole-statement of a concept). The given $\angle JKM \cong \angle MKL$ is not a concept but a part-statement. However, both of these statements are part-statements of the PERPENDICULAR-ADJACENT-ANGLES concept $\overline{LJ} \perp \overline{MK}$. So, the given can be used to prove $\overline{LJ} \perp \overline{MK}$ and then, $\overline{LJ} \perp \overline{MK}$ can be used to prove the goal. The final proof plan will look something like this:

$$\begin{tikzcd}
\mathrm{rt}\angle MKL \\
\overline{LJ} \perp \overline{MK} \arrow[u] \\
\angle JKM \cong \angle MKL \arrow[u]
\end{tikzcd}$$

You need to learn how to make the concept $\overline{LJ} \perp \overline{MK}$. When the problem is ready, do the following.

Making a Concept:

- Mouse-click on the PERPENDICULAR-ADJACENT-ANGLES configuration in the menu on the left – it's the 5th item down from the top.
- Now click on the segments in the diagram that make up this configuration. Click anywhere near the middle of a segment to select. The segments will highlight as you pick them.

- Highlight all the segments that make up the PERPENDICULAR-ADJACENT-ANGLES configuration, in this case every segment in the diagram but $\overline{LM}$.
- If you make a mistake, you can click on a segment a second time. This will erase the highlighting of that segment.
- When you've highlighted only the segments, $\overline{LK}$, $\overline{KJ}$, and $\overline{KM}$, click on "DONE" in the menu below the diagram. A concept will be created, complete with the whole-statement and a miniature configuration. If you get an error message, try again.

To finish the proof plan, you need to move the concept into the middle of the screen. Moving works just like it does with rules – see the directions above for *Moving* if you need to. Next, *Justify* $\overline{LJ} \perp \overline{MK}$ just like you did for the goal statement in the problems above. Similarly, you need to justify the goal rt∠MKL and select $\overline{LJ} \perp \overline{MK}$ as the reason.

To fill in the details, you need to add rules on both steps. In other words, you need a rule to go from ∠JKM ≅ ∠MKL to $\overline{LJ} \perp \overline{MK}$ and a rule to go from $\overline{LJ} \perp \overline{MK}$ to rt∠MKL.

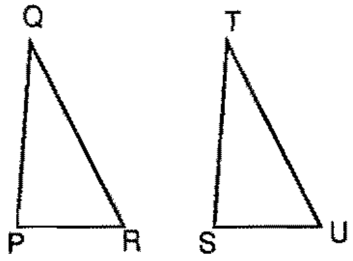
[ ] *Finish P1..*
Choose problem N2.

In the problems so far, there's only been one given statement and only one concept configuration in the diagram. In problem N2, there are two given statements and two examples of the PERPENDICULAR-ADJACENT-ANGLES configuration in the diagram. You need to figure out which of these two is relevant to proving the problem goal.

[ ] *Do N2 now.*
Choose problem N3.

Triangle Congruence

The remaining concepts and rules you will learn with ANGLE are related to triangle congruence. Just as segments and angles can be congruent, so can triangles. Triangles are congruent if they have the same size and shape. The following two triangles are congruent:



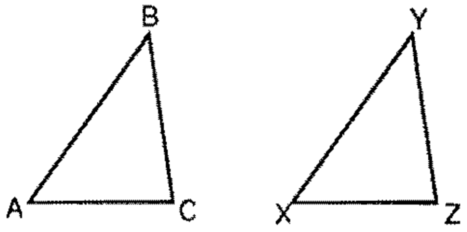
If two triangles are congruent, it is possible to place one on top of the other so that the first exactly covers the second. For $\triangle PQR$ to be congruent to $\triangle STU$, the following points need to be in correspondences:

$P \sim S \qquad Q \sim T \qquad R \sim U$

We indicate this triangle congruence in geometry with the statement $\triangle PQR \cong \triangle STU$. Alternatively we can say $\triangle QRP \cong \triangle TUS$ or $\triangle QPR \cong \triangle TSU$ or any of the other 6 possibilities where the points that correspond are in the same position. Note that $\triangle PRQ \cong \triangle STU$ is not a correct way to specify the congruence between these triangles because this indicates that $\overline{PR} \cong \overline{ST}$ which is clearly not true (look at the diagram above).

What Are the Properties of Congruent Triangles?

What does the fact that two triangles are congruent tell you about the segments and angles formed by the triangles? Consider the triangles below:

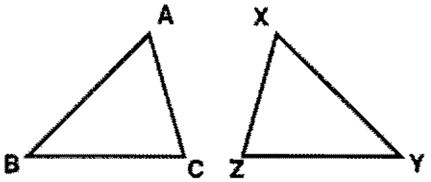


What can you say about the segments, $\overline{AB}$, $\overline{BC}$, $\overline{CA}$, $\overline{XY}$, $\overline{YZ}$, and $\overline{ZX}$, in these triangles? Given that you are told $\triangle ABC \cong \triangle XYZ$, can you tell whether any of these segments are equal to each other? Compare the size of $\overline{AB}$ with the size of each of the other five segments. You should notice that $\overline{XY}$ is the same size. Perhaps this is not a surprise to you, since if you slide $\triangle XYZ$ on top of $\triangle ABC$, $\overline{AB}$ and $\overline{XY}$ fit exactly on top of each other. So do $\overline{BC}$ and $\overline{YZ}$ as well as $\overline{CA}$ and $\overline{ZX}$. In other words, the segments of CONGRUENT-TRIANGLES have these three properties or part-statements: 1) $\overline{AB} \cong \overline{XY}$, 2) $\overline{BC} \cong \overline{YZ}$, and 3) $\overline{CA} \cong \overline{ZX}$.

What about the angles? The three corresponding angles of each triangle are also the same size. Thus: 4) $\angle B \cong \angle Y$, 5) $\angle C \cong \angle Z$, and 6) $\angle A \cong \angle X$ are also part-statements of the CONGRUENT-TRIANGLES concept.

Here's the CONGRUENT-TRIANGLES concept showing the six part-statements. We'll discuss the ways-to-prove below.

CONGRUENT-TRIANGLES

Configuration:	
Whole-statement:	$\triangle ABC \cong \triangle XYZ$
Part-statements:	1. $\overline{AB} \cong \overline{XY}$ 2. $\overline{BC} \cong \overline{YZ}$ 3. $\overline{CA} \cong \overline{ZX}$ 4. $\angle B \cong \angle Y$ 5. $\angle C \cong \angle Z$ 6. $\angle A \cong \angle X$

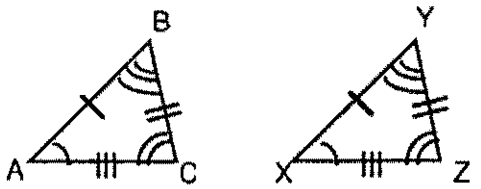
Ways-to-prove:	SSS: {1 2 3}
	SAS: {1 4 2} {2 5 3} {3 6 1}
	ASA: {1 4 6} {2 4 5} {3 5 6}
	AAS: {1 4 5} {1 5 6} {2 4 6}
	{2 5 6} {3 4 5} {3 4 6}

Other related rules:	CORRES-PARTS

Look at problem N3. Since the goal $\overline{AB} \cong \overline{EF}$ is a part-statement of the given $\triangle ABC \cong \triangle EFG$, your plan simply involves proving the goal using the given.

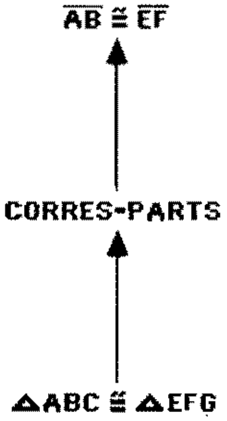
To fill in the details you'll need to use the Corresponding Parts rule, abbreviated as **CORRES-PARTS**. This rule is also called CPCTP for "**C**orresponding **P**arts of **C**ongruent **T**riangles are **C**ongruent".

CORRES-PARTS



If two triangles are congruent, $\triangle ABC \cong \triangle XYZ$, *then* all the corresponding sides are congruent: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{CA} \cong \overline{ZX}$, and all the corresponding angles are congruent: $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\angle CAB \cong \angle ZXY$.

This rule is not an *if and only if* rule – it can only be used in the left-to-right direction to prove segments or angles congruent from congruent triangles. Here's how your final proof for N3 should look:


$\overline{AB} \cong \overline{EF}$
CORRES-PARTS
$\triangle ABC \cong \triangle EFG$

$\overline{AB} \cong \overline{EF}$ is proven from $\triangle ABC \cong \triangle EFG$ using CORRES-PARTS as the rule. CORRES-PARTS **cannot** be used in the right-to-left direction to prove triangles congruent from congruent segments and angles. The rules below are for that purpose.

[] *Finish N3 by inserting the* CORRES-PARTS *rule.*
Choose problem PROB310 and read on.

Ways to Prove Triangles Congruent

Look on your Rule Summary Sheet and you'll notice that the next 4 rules can all be used to prove triangles congruent. These four rules indicate the different combinations of CONGRUENT-TRIANGLES part-statements which are enough to prove the triangles congruent. Basically, you need to know at least three part-statements to prove triangles congruent. But, not just any three will do. What three work?

Combinations of that work:
3 sides (SSS),
2 sides and 1 angle **if** the angle is *included* (SAS), or
1 side and 2 angles (ASA or AAS).

Combinations that don't work:
- <3 part-statements,
- 2 sides and 1 angle that is not *included*,
- 3 angles.

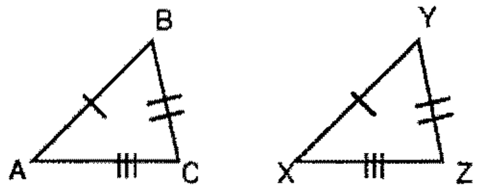
The combinations that work are summarized in the *ways-to-prove* of the CONGRUENT-TRIANGLES concept above and on your Concept Summary Sheet. Notice that there is only 1 combination of three sides, 3 combinations of two sides and an included angle, 3 combinations of two angles and an included side, and 6 combinations of two angles and a non-included side.

You can do PROB310 with a one step plan since the givens are the 3 segment part-statements of the goal $\triangle GFE \cong \triangle DCA$. In other words, *Justify* the goal statement and select all three givens as reasons. Don't forget to click on "DONE".

Filling in the Details of Triangle Congruence Plans

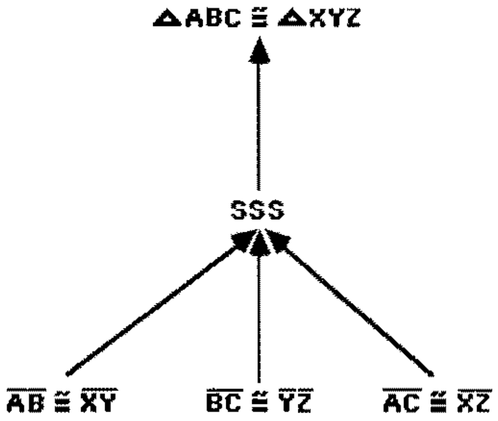
There are four rules for proving triangles congruent. The first is the side-side-side postulate, which we will abbreviate sss:

SSS:



If three sides of one triangle, $\overline{AB}$, $\overline{BC}$, and $\overline{AC}$, are congruent to the corresponding sides of another triangle, $\overline{XY}$, $\overline{YZ}$, and $\overline{XZ}$, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

Here is how sss is used as part of a proof:


$\triangle ABC \cong \triangle XYZ$
SSS
$\overline{AB} \cong \overline{XY}$
$\overline{BC} \cong \overline{YZ}$
$\overline{AC} \cong \overline{XZ}$

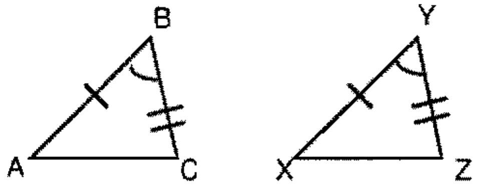
To finish up PROB310, you need to insert the sss rule into your plan. Place sss on any one of the three lines going up to $\triangle GFE \cong \triangle DCA$. *REPEAT*: You only need to put it on **one** of the three lines. Then choose *Insert* as usual and click on **sss**. Don't forget to click on DONE.

[ ] *Finish PROB310 if you haven't..*
Choose problem N4 and read on.

As part of your solution plan for problem N4, you need to construct a TRIANGLE-CONGRUENCE concept for the triangles that appear in the diagram. Mouse-click on the TRIANGLE-CONGRUENCE configuration in the menu of the left (the 9th item down from the top) and then, mouse-click on the segments in the diagram that make up these triangles. See the directions for *Making a Concept* above if you need help.

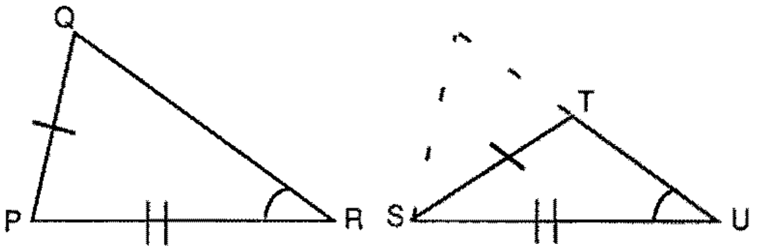
When you're ready to fill in the details you'll need a new rule for proving triangle congruence. Instead of using just sides, this rule uses a combination of angles and sides:

SAS:



If two sides and the *included* angle of one triangle, $\overline{AB}$, $\overline{BC}$, and $\angle ABC$, are congruent to the corresponding parts, $\overline{XY}$, $\overline{YZ}$, and $\angle XYZ$, of another triangle, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\angle ABC \cong \angle XYZ$, *then* the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

Note that the side-angle-side postulate requires that the angles be contained between the two congruent segments. The figure below shows why this is critical.

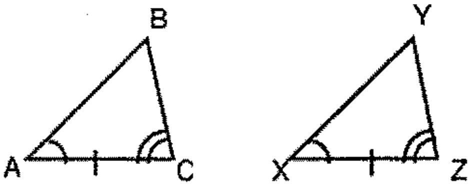


Even though there are two sets of corresponding sides, $\overline{PQ} \cong \overline{ST}$ and $\overline{PR} \cong \overline{SU}$, and a set of corresponding angles, $\angle PRQ \cong \angle SUT$, clearly these triangles are not congruent. (Note: the dashed lines indicate how $\triangle PRQ$ would look on top of $\triangle SUT$.) In order for **SAS** to apply, *make sure the angle is at the point where the two congruent segments meet.* If $\angle RPQ \cong \angle UST$ had been given in the picture above, then you could prove the triangles congruent by **SAS**.

[ ] *Do problem N4.*
Choose problem N5 and read on.

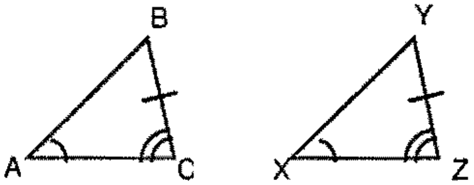
Only under the circumstances indicated above is it possible to prove triangles congruent when you have *two sides* and *an angle* as congruent corresponding parts. However, no special circumstances are necessary when you have *two angles* and *a side* as congruent corresponding parts. The two rules **ASA** and **AAS** are used in these situations.

ASA:



If two angles and the included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{BC}$, are congruent to the corresponding parts, $\angle XYZ$, $\angle YZX$, and $\overline{YZ}$, of another triangle, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{BC} \cong \overline{YZ}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

AAS:



If two angles and a non-included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{AB}$, are congruent to the corresponding sides of another triangle, $\angle XYZ$, $\angle YZX$, and $\overline{XY}$, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{AB} \cong \overline{XY}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

You now have four rules, **SSS**, **SAS**, **ASA**, and **AAS**, that you can use to prove triangles congruent.

[ ] *Finish problem N5.*
Choose problem P5 and read on.

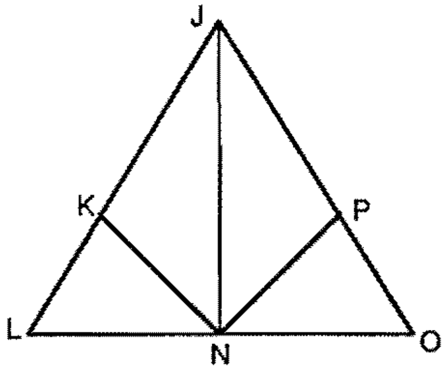
Overlapping Concepts

In the next few problems we'll be exploring cases where two concepts *overlap*. Look in the diagram for problem P5 below.

P5:

Given: $\triangle JKN \cong \triangle JPN$
$\angle KLN \cong \angle LOP$
$\angle LNK \cong \angle PNO$

Goal: $\triangle LKN \cong \triangle OPN$



Try to figure how concepts $\triangle JKN \cong \triangle JPN$ and $\triangle LKN \cong \triangle OPN$ overlap. In other words, what part-statement of $\triangle JKN \cong \triangle JPN$ is also a part-statement of $\triangle LKN \cong \triangle OPN$. On the back of your *Tracking Sheet*, write down the 6 part-statements of $\triangle JKN \cong \triangle JPN$. After you've done that write down the 6 part-statements of $\triangle LKN \cong \triangle OPN$. Circle the statement that appears in both lists. Ask Ken to check this when you're done.

Finding such overlapping part-statements is usually a big clue to doing a proof. In P6, $\overline{KN} \cong \overline{PN}$ provides you a way to get from $\triangle JKN \cong \triangle JPN$ to $\triangle LKN \cong \triangle OPN$. You can prove $\overline{KN} \cong \overline{PN}$ using $\triangle JKN \cong \triangle JPN$ and then, use $\overline{KN} \cong \overline{PN}$ along with the other part-statements you're given to prove $\triangle LKN \cong \triangle OPN$.

In order to prove $\overline{KN} \cong \overline{PN}$, you'll first need to create this statement. Here's how.

Making a Segment Congruence Statement:

- Mouse-click on the item at the bottom of the menu on the left, just above the action menu. It contains three segments with one, two, and three markings on them.
- Mouse-click on $\overline{KN}$ in the problem diagram. "$\overline{KN} \cong$" should appear in the box below the diagram. Now, mouse-click on $\overline{PN}$ and $\overline{KN} \cong \overline{PN}$ should now appear in the box.
- Mouse-click on "DONE".
- NOTE: If you want to create a segment that is made up of two smaller segments, click on one segment, keep the mouse down, and drag over the other segment.

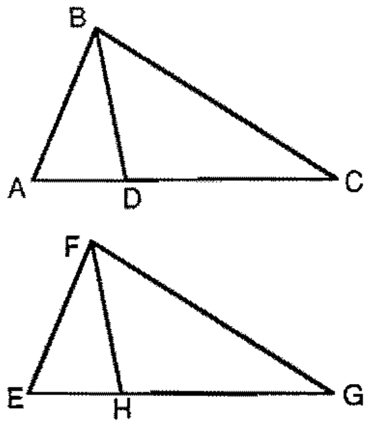
You can move this statement into the proof and justify it just like a concept. In this case, you want to *Justify* $\overline{KN} \cong \overline{PN}$ and then pick $\triangle JKN \cong \triangle JPN$ as the reason.

[] *Finish problem P5.*
Choose problem P6 and read on.

P6:

Given: △JKN ≅ △JPN
∠KLN ≅ ∠LOP
∠LNK ≅ ∠PNO

Goal: △ABC ≅ △EFG



The proof for problem P6 involves 3 overlapping CONGRUENT-TRIANGLES concepts. Write down the part-statements for each of the three triangle congruences. Figure out how the two given triangles overlap with the goal triangles. You should find 2 overlaps between △ABD ≅ △EFH and △ABC ≅ △EFG, and 2 overlaps between △BCD ≅ △FGH. and △ABC ≅ △EFG. Have Ken check these.

In order to do problem P6, you'll need to create the angle congruence statements you find in the overlap between the triangles. Here's how.

Making an Angle Congruence Statement:

- In the menu on the left, mouse-click on the item just above the item for congruent segments. It contains three angles with one, two, and three markings on them.
- To indicate an angle, you need to mouse-click near the vertex right about where you would draw a mark with a pencil to indicate angle congruence. When you've selected an angle, it should appear in the box below the diagram.
- When you've selected two angles, mouse-click on "DONE".
- NOTE: If you want to create an angle that is made up of two smaller angles, click on one angle, keep the mouse down, and drag over the other angle.

Work on problem P6. Make the overlapping statements you found and prove them using the given triangles. Then, figure out how you can use these statements to prove the goal triangle. *Warning*: Remember there is no SSA rule – if you have two sides and an angle, the angle must be the *included* angle.

[ ] *Finish problem P6.*
Choose problem P7 and read on.

Sometimes you need to *search* for the key concept or concepts that are needed to form a proof plan. In problem P7, think about what triangles you could prove congruent in order to prove the goal ∠JLG ≅ ∠QSN. When you've found those triangles, create the concept for them – use the menu on the left and mouse-click on the segments which make up the two triangles.

Now, think about how the triangle-congruence you want to prove overlaps with the triangle-congruence you are given, $\triangle KHJ \cong \triangle RPQ$.

If you're having trouble with figuring out how to do the proof, you can ask ANGLE for a hint:

Requesting Hints from ANGLE:

- Go up to the "Info" menu, mouse-click on "Info" and hold the mouse button down. Slide down to "Hint" and let go of the mouse button.
- Read the message that should appear in the upper right corner of the screen.
- If you want a stronger hint, try "Hint" again. You can keep getting more and more specific hints and if you are really stuck the tutor will eventually tell you what to do.

[ ] *Finish problem P7.*
Choose problem P8 and read on.

On harder problems, like P8, it is often useful to try to prove triangles congruent even if you don't know how they will help you prove the goal. So, in this problem diagram, find any two triangles which look congruent and make the corresponding concept. Try to prove it. If you can't, try to prove some other triangles congruent. Then when you've done that, go back to the first triangles if you need to, and try to prove them congruent using what you've learned.

Remember, it helps to figure out how the corresponding segments and angles of proven congruent triangles *overlap* with the corresponding parts of triangles you are trying to prove congruent.

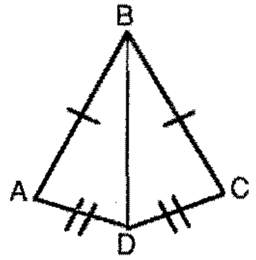
Ask for a hint, if you're having trouble. See "Requesting Hints from ANGLE" above.

[ ] *Finish problem P8.*
Choose problem PROB311 and read on.

Congruent Triangles That Share a Side

Planning with the CONGRUENT-TRIANGLES-SHARED-SIDE Concept

In the following diagram it is given that $\overline{AB} \cong \overline{BC}$ and $\overline{AD} \cong \overline{CD}$.

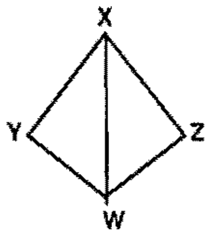


Are the two triangles, $\triangle ABD$ and $\triangle CBD$, congruent? Even though you're only told two statements about them and usually you need at least three? Yes, the triangles are congruent – you could fold one exactly on top of the other. How is it that these triangles are definitely congruent even though you're only given two statements about them?. The answer is that since $\overline{BD}$ is a side of both triangles, so the triangles actually have three corresponding sides which are congruent.

In ANGLE there is a special concept for congruent triangles that share a side:

CONGRUENT-TRIANGLES-SHARED-SIDE

Configuration:



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

Part-statements:

1. $\overline{XY} \cong \overline{XZ}$
2. $\overline{YW} \cong \overline{ZW}$
3. $\angle Y \cong \angle Z$
4. $\angle YXW \cong \angle ZXW$
5. $\angle XWY \cong \angle XWZ$

Ways-to-prove:

SSS: {1 2}
SAS: {1 4} {2 5}
ASA: {4 5}
AAS: {3 4} {3 5}

Other related rules:

CORRES-PARTS
REFLEXIVE

When two triangles share a side, you only need two segment or angle congruence statements to prove them congruent – as long as they are the right two. These are indicated in the *ways-to-prove* of the CONGRUENT-TRIANGLES-SHARED-SIDE concept. Notice that the first way-to-prove, {1 2}, corresponds with the situation above where the two segment part-statements are used to prove the triangles congruent. This is also the situation in PROB311.

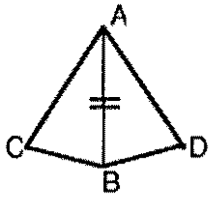
Working out a plan for problem PROB311 is quite simple. You can prove the goal $\triangle WXY \cong \triangle ZXY$ directly from the givens because the goal is a CONGRUENT-TRIANGLES-SHARED-SIDE concept and the givens correspond with the first of the ways-to-prove of this concept. In other words, *Justify* $\triangle WXY \cong \triangle ZXY$ and pick $\overline{WY} \cong \overline{YZ}$ and $\overline{WX} \cong \overline{XZ}$ as reasons.

Filling in the Details Using the REFLEXIVE Rule

At the detail level, you need to prove these triangles congruent by SSS. However, you only have two congruent side statements on the screen. You need to construct the reflexive statement $\overline{XY} \cong \overline{XY}$ and prove it by the REFLEXIVE rule. Here's the rule:

REFLEXIVE:

If segment $\overline{AB}$ appears in the diagram,
then $\overline{AB} \cong \overline{AB}$.



Here's how you use it in problem PROB311. First you need to make the reflexive statement – this is just like "Making a Segment Congruence Statement" above:

Making a Reflexive Statement:

- Mouse-click on the segment congruence menu item (at the bottom on the left, just above the action menu).
- Mouse-click on the segment in the diagram. " $\overline{XY} \cong$ " should appear in the box below the diagram. Mouse-click on the segment a second time. $\overline{XY} \cong \overline{XY}$ should now appear in the box.
- Mouse-click on "DONE" and then move the statement into the middle of the screen.

Now you need to prove this statement.

Proving a Reflexive Statement:

- Select REFLEXIVE from the rule menu and move it so that it is below $\overline{XY} \cong \overline{XY}$.
- Now *Justify* $\overline{XY} \cong \overline{XY}$ by choosing *Justify* from the action menu and mouse-clicking on $\overline{XY} \cong \overline{XY}$.
- Pick the REFLEXIVE rule as the reason by mouse-clicking on it.
- Mouse-click on "DONE".

You don't need any premises going into the REFLEXIVE rule as long as the segment appears in the diagram. Everything else in your proofs must be built up from the givens. In other words, for your proof to be complete, *everything but the givens and the REFLEXIVE rule should have thick lines going into it.*

The last thing you need to do to finish PROB311, is add the SSS rule.

Adding a Rule to Prove a CONGRUENT-TRIANGLES-SHARED-SIDE concept:

- Get **SSS** from the rule menu and move it up top of one of the two lines going to $\triangle WXY \cong \triangle ZXY$. Make sure it is higher on the screen the reflexive statement $\overline{XY} \cong \overline{XY}$.
- Now, *Insert* the rule by choosing *Insert* and mouse-clicking on **SSS**. **But**, before you choose "DONE" mouse-click on $\overline{XY} \cong \overline{XY}$ to add it as the third premise for **SSS**.
- Mouse-click on "DONE".

[ ] *Finish problem PROB311.*
Choose problem PROB352 and read on.

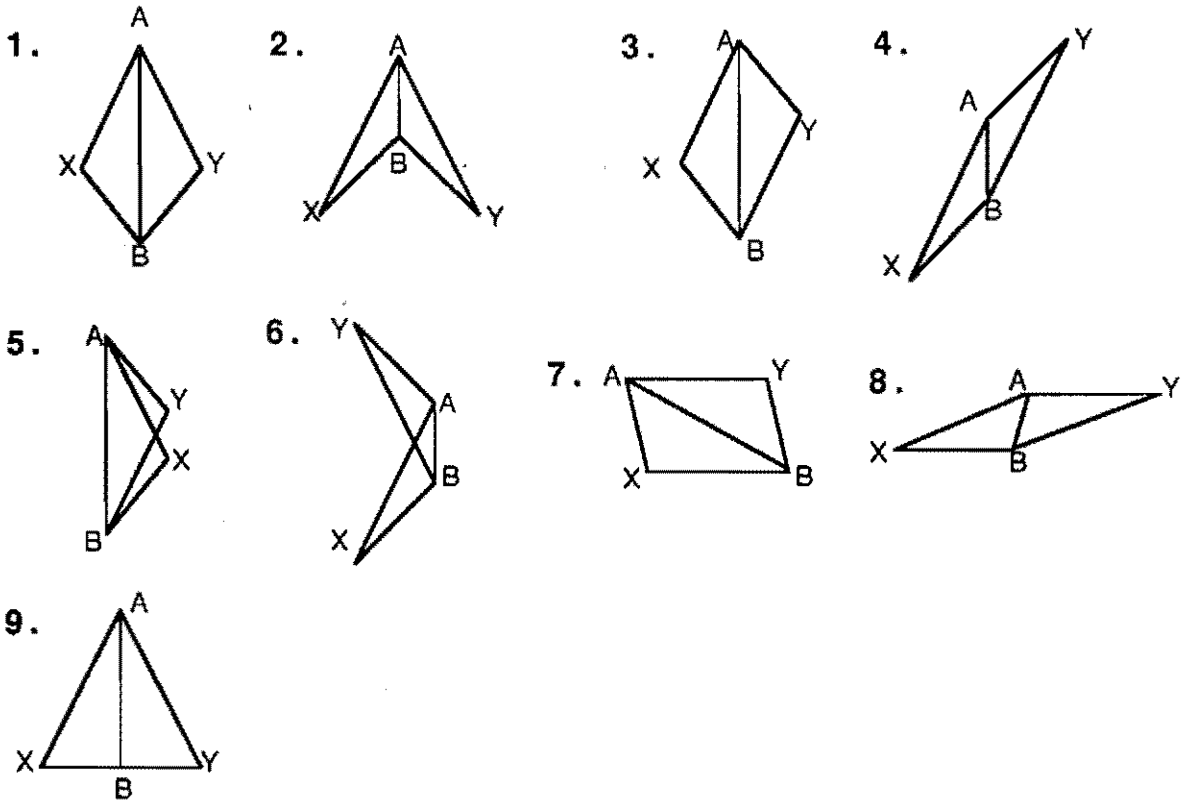
In this problem, you'll have to construct a CONGRUENT-TRIANGLES-SHARED-SIDE concept. Pick the CONGRUENT-TRIANGLES-SHARED-SIDE configuration in the menu on the left – it's the picture just below the CONGRUENT-TRIANGLES configuration. Now indicate the segments in the diagram that make up $\triangle WXZ \cong \triangle YZX$.

Finish the proof plan by using this concept as the intermediate step between the givens and the goals. Then fill in the details. Remember to add the reflexive statement, justify it, and then select it as an extra premise when inserting the rule for $\triangle WXZ \cong \triangle YZX$.

[ ] *Finish problem P352.*
Choose problem PROB353 and read on.

As shown below CONGRUENT-TRIANGLES-SHARED-SIDE configurations can come in many forms beside the one shown above.

CONGRUENT-TRIANGLES-SHARED-SIDE configurations:



The triangles can be on opposite sides of the shared side as in configurations 1-4 and 7-9 or on the same side of the shared side as in 5 and 6. The points on shared side can correspond with themselves as in configurations 1, 2, and 9 where ▲XAB ≅ ▲YAB or can they can correspond with each other as in the other configurations where ▲XBA ≅ ▲YAB. The triangles involved may have acute angles only (1, 3, 5, and 7), an oblique angle (2, 4, 6, and 8), or a right angle (9).

The diagram for PROB311 is like configuration 2, while the diagrams for PROB352 and PROB353 are like 7. The problems below will have other types of configurations. Watch out for configurations 5 and 6 which are particularly hard to see.

[] *Finish problem P353.*
Choose problem N7 and read on.

As we discussed above, a good strategy for doing difficult problems is looking for overlapping part-statements between triangle concepts you've proven and ones you want to prove. This is also true of other concepts. Overlaps can also occur between a CONGRUENT-TRIANGLE-SHARED-SIDE concept and one of the perpendicular concepts. This happens in problem N7 where the diagram looks like configuration 9 above. Figure out how the PERPENDICULAR-ADJACENT-ANGLES concept and the CONGRUENT-TRIANGLE-SHARED-SIDE concept overlap in this problem. What part-statement is a property of both concepts?

[] *Finish problem N7.*

Choose problem P2 and read on.

For this problem, repeat the exercise of identifying the part-statements which appear both in the given triangle congruence statement and the goal triangle congruence statement. Have Ken check what you come up with. You should find two statements that are part-statements of both triangle congruences.

[ ] *Finish problem P2.*
Choose problem P3 and read on.

Do overlap exercise, just as with P6

[ ] *Finish problem P3.*
Choose problem N16 and read on.

One of the more difficult forms of the CONGRUENT-TRIANGLES-SHARED-SIDE configuration is the case where the two triangles are on the same side of the shared segment – configurations 5 and 6 above. This is the case in problem N16. What is particularly hard about this configuration is seeing the corresponding angles. Write down the three corresponding angle statements for triangles $\triangle ABC$ and $\triangle DCB$ in problem N16. Have Ken check these.

[ ] *Finish problem N16.*
Choose problem N17 and read on.

Remember:

1. *Look in the diagram* for triangles which look congruent – try to prove that they are.
2. *Identify overlaps* between triangles you want to prove congruent and ones you've already proven.

[ ] *Finish problem N17.*
Choose problem N15 and read on.

Review the PERPENDICULAR-ADJACENT-ANGLES concept.

How many part-statements do you need to prove the CONGRUENT-TRIANGLES-SHARED-SIDE concept?

What triangles could you prove congruent in order to prove the goal $\angle BFA \cong \angle CGB$ of problem N15?

[ ] *Finish problem N15.*
Choose problem N9 and read on.

The triangles you need in this problem are a little difficult to see. Counting all the triangles in the diagram may help you to notice some triangles that you didn't at first. If you are having trouble, ask ANGLE for help. See "Requesting Hints from ANGLE" above.

[ ] *Finish problem N9.*
Choose problem N8 and read on.

Remember:

1. *Look in the diagram* for examples of perpendicularity or triangle congruence concepts.
2. *Identify overlaps* between concepts you've proven and ones you want to prove.

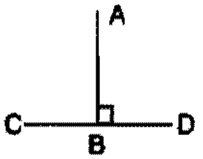
[ ] Finish problem N8 and keep these things in mind as you do the remaining problems on your tracking sheet. If you finish all the problems, talk to Ken.

GEOMETRY PROOF TUTOR TEXT

You'll be working with a computerized tutoring system called the Geometry Proof Tutor or *GPT*, for short. This text describes how to use GPT to solve geometry proof problems.

In the first problem, you'll be using the rule **DEF-PERP**. Here is a statement of the rule:

DEF-PERP:



Two lines are perpendicular, $\overline{AB} \perp \overline{CD}$,
if and only if they form a right angle rt∠ABD.

This rule and the others you'll be using appear on the *Rule Summary Sheet* you should have been given. You can refer to this sheet when doing problems. Let's start our first problem on GPT.

Using GPT to Solve a Proof Problem

This section will take you step-by-step through the solution of a problem. We'll be doing the first problem on your *Tracking Sheet*, PROB150.

Selecting a Problem:

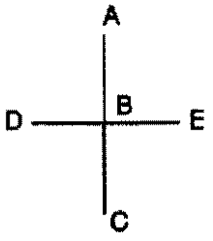
- Look on the computer screen and you should see a *problem menu*, that is, a list of problems from which you can choose. Locate PROB150 in the menu.
- Move the mouse so that the arrow on the screen is pointing to PROB150 in the menu.
- Hold down the the left mouse button and do not let go. PROB150 should highlight in the diagram.
- If PROB150 is not highlighted, keep the left button down and move the mouse until it is. Then let go of the button to select PROB150.

It will take a little while for the problem to come up. Read on, while you wait. Here's what PROB150 looks like in the usual notation:

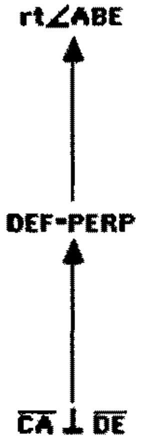
PROB150:

Given: $\overline{CA} \perp \overline{DE}$

Goal: rt∠ABE



When GPT is ready you'll notice the problem appears on the computer screen in a different form. The givens are on the bottom and the goal is on the top Your job is to link the givens to the goal, using geometry rules and statements. When you are done, your proof will look something like this:



At the bottom of the screen you should notice the question "What statements are you working from?". The tutor wants to know what given or proven statements you are going to use as the premise for a geometry rule.

Selecting Premises:

- Move the mouse pointer to the given statement, $\overline{CA} \perp \overline{DE}$, and press and release the *left* button. A box should appear around the statement. If a box didn't appear, try it again, but hold the mouse button down a little longer this time.
- Now, move the mouse up a little bit and press and hold down the *right* button. A menu should appear.

Slide the mouse up so that the "Done selecting" option is highlighted and then, let go of the button.

You may have wait a second or two, but than the question: "What is the rule?" should appear at the bottom of the screen. The tutor wants you to type in the rule that follows from the premise(s) you've selected.

Typing the Rule:

- Use the keyboard to type in the rule, DEF-PERP.
- When you are done typing, press the ENTER key. Its on the right side of the keyboard and has two arrows on it pointing down and to the left. The rule should appear on the screen with a line from the premise, $\overline{CA} \perp \overline{DE}$, to it.

Finally, GPT wants the conclusion that follows from applying the rule you typed to the premise you selected. The question "What is the conclusion?" should appear at the bottom of the screen. There are actually four possible conclusions of applying DEF-PERP to $\overline{CA} \perp \overline{DE}$: rt∠ABD, rt∠DBC, rt∠CBE, and rt∠EBA. But since rt∠EBA is the goal, you should enter it.

Entering the Conclusion:

- Move the mouse up to the menu in the top-left corner of the screen and find the item for right angles: rt∠.
- Click the *left* button on top of rt∠. You should see this symbol appear at the bottom of the screen. If you don't, try it again, but hold the mouse down a little longer this time.
- Move to the problem diagram to indicate the points E, B, and A:
  - Click the left button on point E in the problem diagram. You should now see rt∠E on the screen.
  - Left-click on point B in the diagram.
  - Left-click on point A.
- You should see rt∠EBA on the screen. If you make a mistake you can use the RUBOUT or CLEAR INPUT options at the bottom of the menu. Press the ENTER key when you are done.

After you press ENTER, GPT should respond by drawing a line from DEF-PERP to rt∠EBA and then printing "SUCCESS" to indicate that you have completed the proof.

Now you should load PROB152 (if you forgot how, see the note above on **Selecting a Problem**). Since it takes a while for GPT to prepare a problem, you should read on while you wait.

While PROB150 illustrated the use of the DEF-PERP rule in the *left-to-right* direction, that is, from perpendiculars to a right angle, PROB152 illustrates the use DEF-PERP in the *right-to-left* direction, that is, from a right angle to perpendiculars.

"If and Only If" Rules

The phrase "if and only if" in the Definition of Perpendicular Lines indicates that this rule works in both directions. If you are given or have proven a perpendicular statement on the screen, then you can use DEF-PERP (in the *left-to-right* direction) to prove that any one of the angles formed is a right angle. This is what you did in PROB150.

Going in the other direction, if you have a right angle statement on the screen, you can use DEF-PERP (in the *right-to-left* direction) to prove that the lines making up this angle are perpendicular. You will use DEF-PERP in this way in PROB152.

Look at DEF-PERP on your rule summary sheet. The double arrows "<=>" indicate that this rule works in both directions. This is not the case for all rules, for example, look at the rule CORRES-PARTS on the sheet. This rule can only be used in the left-to-right direction. This is indicated by a single headed arrow "==>".

If forgot how to **Select Premises**, **Type the Rule**, or **Enter the Conclusion**, use the directions above. In doing this problem, you can use a *short-cut* for entering the conclusion. Whenever the conclusion is the problem goal, you can simply point to it and indicate "DONE SELECTING".

Enter the Conclusion Short-Cut:

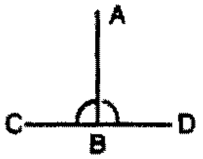
- Make sure you've selected premises and typed a rule. If you've done so GPT should be asking you, "What is the conclusion?".
- Left-click the mouse on the goal statement, $\overline{DC} \perp \overline{EA}$.

- Hold done the right button, choose "DONE SELECTING", and then let go of the button.

[ ] *If you haven't already, do problem PROB152 now.*
When you are done, chose the next problem, N1

The next rule you'll use in problems is called Congruent Adjacent Angles, which we abbreviate **CONG-ADJ-ANGS**.

CONG-ADJ-ANGS:



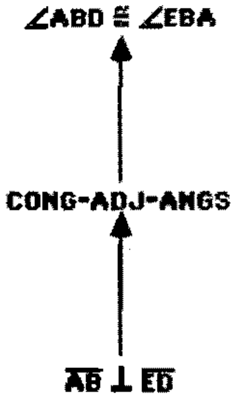
Angles formed by connecting lines are congruent, $\angle ABC \cong \angle ABD$,
if and only if the lines are perpendicular, $\overline{AB} \perp \overline{CD}$.

Notice that this rule is also an "if and only if" rule. You can use it in the *left-to-right* direction to get from congruent adjacent angles to perpendicular lines or you can use it in the *right-to-left* direction to get from perpendicular lines to prove the adjacent angles are congruent.

Use **CONG-ADJ-ANGS** in the left-to-right direction in problem N1.

[ ] *Do N1 now.*
Choose problem PROB151 next and read on.

The picture below illustrates the use of **CONG-ADJ-ANGS** in the right-to-left direction:



[ ] *Do PROB151 now.*
Choose problem P1.

This is your first two step problem. After selecting a premise and typing a rule, this time you won't be able to use the short-cut for entering the conclusion, since the conclusion of the first step, $\overline{LJ} \perp \overline{MK}$, is not the goal. You'll need to enter this statement as described above in **Enter the Conclusion**.

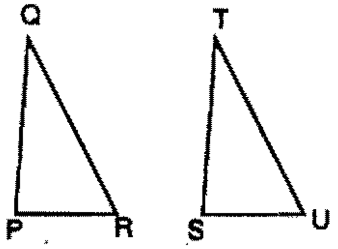
[ ] *Do P1 now.*
Choose problem N2.

In the problems so far, there's only been one given statement and you always start by selecting it. In the next problem, N2, there are two given statements and you need to figure out whether to select both or just one of them. (*Hint*: Both the rules you've learned so far can only have one premise.)

[ ] *Do N2 now.*
Choose problem N3.

Triangle Congruence

All the remaining rules you will learn with GPT are related to triangle congruence. Just as segments and angles can be congruent, so can triangles. Triangles are congruent if they have the same size and shape. The following two triangles are congruent:



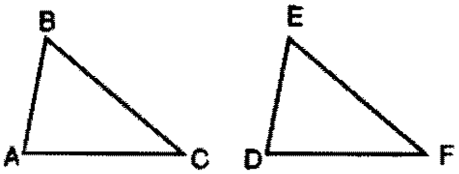
If two triangles are congruent, it is possible to place one on top of the other so that the first exactly covers the second. For $\triangle PQR$ to be congruent to $\triangle STU$, the following points need to be in correspondences:

$$P \sim S \qquad Q \sim T \qquad R \sim U$$

We indicate this triangle congruence in geometry with the statement $\triangle PQR \cong \triangle STU$. Alternatively we can say $\triangle QRP \cong \triangle TUS$ or $\triangle QPR \cong \triangle TSU$ or any of the other 6 possibilities where the points that correspond are in the same position. Note that $\triangle PRQ \cong \triangle STU$ is not a correct way to specify the congruence between these triangles because this indicates that $\overline{PR} \cong \overline{ST}$ which is clearly not true (look at the diagram above).

Proving Parts of Triangles Congruent

By definition we know that if 2 figures are congruent their corresponding parts are congruent. Consider the two triangles below.

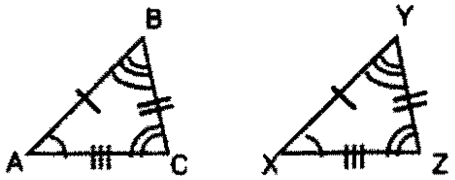


If $\triangle ABC \cong \triangle DEF$ then we can conclude any of the following:

$\overline{AB} \cong \overline{DE}$	$\angle A \cong \angle D$
$\overline{BC} \cong \overline{EF}$	$\angle B \cong \angle E$
$\overline{CA} \cong \overline{FD}$	$\angle C \cong \angle F$

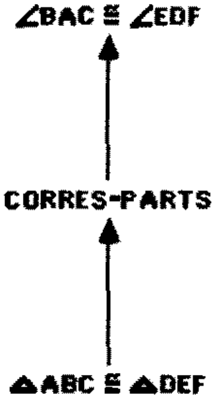
To do so you need to use the Corresponding Parts rule, abbreviated as CORRES-PARTS. This rule is also called CPCTP for "Corresponding Parts of Congruent Triangles are Congruent".

CORRES-PARTS



If two triangles are congruent, $\triangle ABC \cong \triangle XYZ$, *then* all the corresponding sides are congruent: $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{CA} \cong \overline{ZX}$, and all the corresponding angles are congruent: $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\angle CAB \cong \angle ZXY$.

This rule is not an *if and only if* rule – it can only be used in the left-to-right direction to prove segments or angles congruent from congruent triangles. Here's an example showing how $\angle BAC \cong \angle EDF$ can be proven using CORRES-PARTS as the rule and $\triangle ABC \cong \triangle DEF$ as the premise:



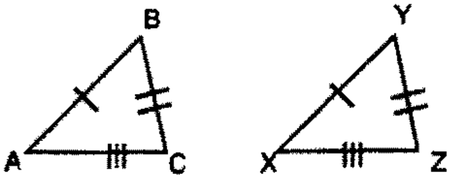
CORRES-PARTS*cannot* be used in the right-to-left direction to prove triangles congruent from congruent segments and angles. The rules below are for that purpose.

[ ] *Do problem N3.*
Choose problem PROB310 and read on.

Proving Triangles Congruent

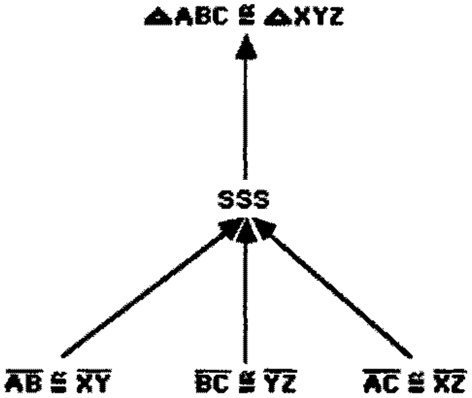
Look on your rule summary sheet and you'll notice that the next 4 rules can all be used to prove triangles congruent. The fisrt is the side-side-side postulate, which we will abbreviate SSS:

SSS:



If three sides of one triangle, $\overline{AB}$, $\overline{BC}$, and $\overline{AC}$, are congruent to the corresponding sides of another triangle, $\overline{XY}$, $\overline{YZ}$, and $\overline{XZ}$, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

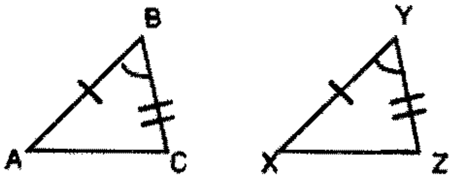
Here is how SSS is used as part of a proof:


$\triangle ABC \cong \triangle XYZ$
SSS
$\overline{AB} \cong \overline{XY}$
$\overline{BC} \cong \overline{YZ}$
$\overline{AC} \cong \overline{XZ}$

[ ] *Do problem PROB310.*
Choose problem N4 and read on.

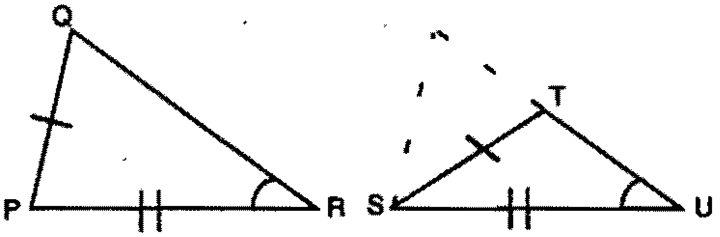
Another rule for determining triangle congruence uses a combination of angles and sides:

SAS:



If two sides and the *included* angle of one triangle, $\overline{AB}$, $\overline{BC}$, and $\angle ABC$, are congruent to the corresponding parts, $\overline{XY}$, $\overline{YZ}$, and $\angle XYZ$, of another triangle, that is, $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\angle ABC \cong \angle XYZ$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

Note that the side-angle-side postulate requires that the angles be contained between the two congruent segments. The figure below shows why this is critical.

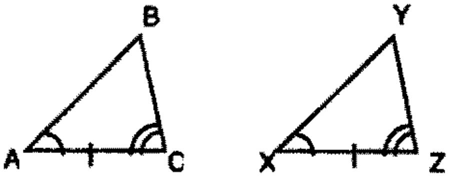


Even though there are two sets of corresponding sides, $\overline{PQ} \cong \overline{ST}$ and $\overline{PR} \cong \overline{SU}$, and a set of corresponding angles, $\angle PRQ \cong \angle SUT$, clearly these triangles are not congruent. (Note: the dashed lines indicate how $\triangle PRQ$ would look on top of $\triangle SUT$.) In order for SAS to apply, make sure the angle is at the point where the two segments meet. If $\angle RPQ \cong \angle UST$ had been given in the picture above, then you could prove the triangles congruent by SAS.

[ ] *Do problem N4.*
Choose problem N5 and read on.

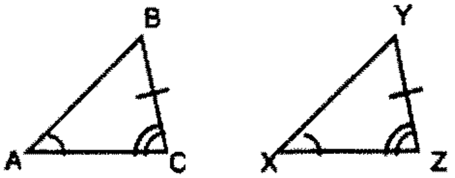
Only under the circumstances indicated above is it possible to prove triangles congruent when you have *two sides* and *an angle* as congruent corresponding parts. However, no special circumstances are necessary when you have *two anlges* and *a side* as congruent corresponding parts. The two rules ASA and AAS are used in these situations.

ASA:



If two angles and the included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{BC}$, are congruent to the corresponding parts, $\angle XYZ$, $\angle YZX$, and $\overline{YZ}$, of another triangle, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{BC} \cong \overline{YZ}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

AAS:



If two angles and a non-included side of one triangle, $\angle ABC$, $\angle BCA$, and $\overline{AB}$, are congruent to the corresponding sides of another triangle, $\angle XYZ$, $\angle YZX$, and $\overline{XY}$, that is, $\angle ABC \cong \angle XYZ$, $\angle BCA \cong \angle YZX$, and $\overline{AB} \cong \overline{XY}$,
then the triangles are congruent: $\triangle ABC \cong \triangle XYZ$.

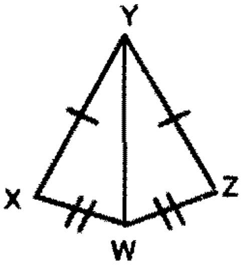
You now have four rules, **SSS**, **SAS**, **ASA**, and **AAS**, that you can use to prove triangles congruent.

[ ] *Do problem N5.* Do P5 - P8

Choose problem PROB311 and read on.

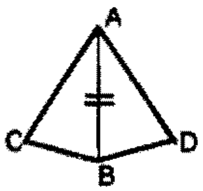
The REFLEXIVE Rule

In the following diagram it is given that $\overline{XY} \cong \overline{YZ}$ and $\overline{XW} \cong \overline{WZ}$:



Can you apply **SSS** to this picture? Well, its the right idea but technically you cannot since **SSS** requires 3 sets of corresponding sides and you've only been given 2. Where do you get the third set? Well, clearly $\overline{YW}$ is congruent to itself. But, the question is how do you prove $\overline{YW} \cong \overline{YW}$? The **REFLEXIVE** rule is just for this purpose:

REFLEXIVE:



If segment $\overline{AB}$ appears in the diagram,
then $\overline{AB} \cong \overline{AB}$.

To show that two segments are congruent by the REFLEXIVE rule you do not need any statements as premises; it can be used as long as the segment appears in the diagram. Thus, when you are selecting premises for the REFLEXIVE rule you should select the diagram as the premise. Let's do it with PROB311:

Using the REFLEXIVE rule:

- Move the mouse over to the problem diagram and click the left button. A box should appear around the diagram. If you don't see a box, try again, but hold the mouse button down a little longer.
- Now indicate you are "Done selecting" in the usual way by holding the right button down letting go when the "Done selecting" option is highlighted.
- Type in the rule, REFLEXIVE and press ENTER.
- Enter the conclusion $\overline{XY} \cong \overline{XY}$.

When you are done with PROB311, this is how it should look on your screen:

[ ] *Finish PROB311.*
Choose problem PROB352.

Getting Help

If you know what you want to do and can't seem to get GPT to do it, feel free to ask Ken for help. If you're having trouble with figuring out how to do the proof, you can ask GPT.

Requesting Hints from GPT:

- Hold down the *right* button and wait for the menu to pop up.
- When it does, move the mouse (still holding the right button done!) past the "Done selecting" option up to the "Explain" option. Let go of the right button when "Explain" is highlighted.
- If you want a stronger hint, try "Explain" again. You can keep getting more and more specific hints and if you are really stuck the tutor will eventually show you what to do.

Finish as many of the remaining problems on the *tracking sheet* as you can.

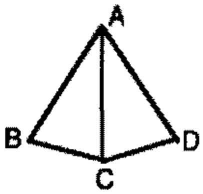
Proof Construction Test: A

Name:______________ Date:____________
Id#:___ Start time:____________
Circle one: Pre Post Finish time:____________

For each problem write a proof of the goal using the givens and any of the rules on the *Rule Summary Sheet* You may assume that any points which appear colinear (on the same line) in the problem diagram are actually colinear. Here's an example problem and solution:

0. Given: $\overline{AB} \cong \overline{AD}$
$\overline{BC} \cong \overline{DC}$

Goal: $\angle BAC \cong \angle DAC$

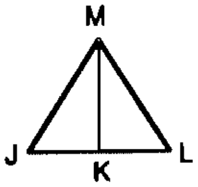


PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{AD}$	1. Given
2. $\overline{BC} \cong \overline{DC}$	2. Given
3. $\overline{AC} \cong \overline{AC}$	3. REFLEXIVE
4. $\triangle ACB \cong \triangle ACD$	4. SSS
5. $\angle BAC \cong \angle DAC$	5. CORRES-PARTS

1. Given: $\overline{JM} \cong \overline{LM}$
$\angle JMK \cong \angle LMK$

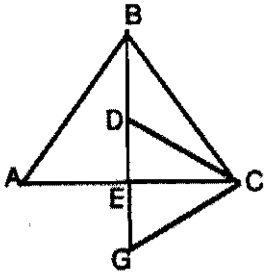
Goal: rt$\angle MKL$



PROOF:

Statements:	Reasons:
1. $\overline{JM} \cong \overline{LM}$	1. Given
2. $\angle JMK \cong \angle LMK$	2. Given

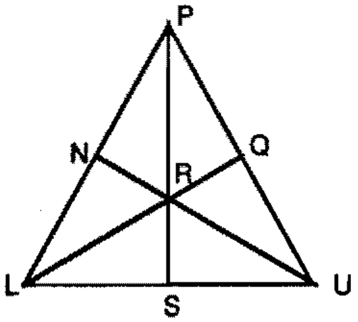
2. Given: $\overline{AB} \cong \overline{BC}$
$\angle ABG \cong \angle CBG$
$\overline{DE} \cong \overline{EG}$
Goal: $\angle DCE \cong \angle GCE$



PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{BC}$	1. Given
2. $\angle ABG \cong \angle CBG$	2. Given
3. $\overline{DE} \cong \overline{EG}$	3. Given

3. Given: rt$\angle PSU$
$\overline{LN} \cong \overline{QU}$
$\angle LPS \cong \angle UPS$
Goal: $\angle LNU \cong \angle UQL$



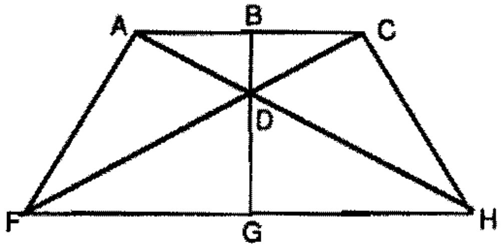
PROOF:

Statements:	Reasons:
1. rt$\angle PSU$	1. Given
2. $\overline{LN} \cong \overline{QU}$	2. Given
3. $\angle LPS \cong \angle UPS$	3. Given

EXTRA CREDIT

Remember: your reasons should only be rules which appear on the rule summary sheet.

4. Given: $\overline{AB} \cong \overline{BC}$
rt$\angle GBC$
$\angle FAH \cong \angle HCF$
$\angle ADF \cong \angle CDH$
Goal: $\overline{AH} \cong \overline{FC}$



PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{BC}$	1. Given
2. rt$\angle GBC$	2. Given
3. $\angle FAH \cong \angle HCF$	3. Given
4. $\angle ADF \cong \angle CDH$	4. Given

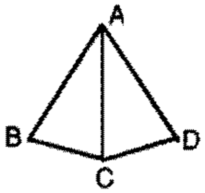
Proof Construction Test: B

Name:________________ Date:______________
Id#:___ Start time:______________
Circle one: Pre Post Finish time:______________

For each problem write a proof of the goal using the givens and any of the rules on the *Rule Summary Sheet* You may assume that any points which appear colinear (on the same line) in the problem diagram are actually colinear. Here's an example problem and solution:

0. Given: $\overline{AB} \cong \overline{AD}$
$\overline{BC} \cong \overline{DC}$

Goal: $\angle BAC \cong \angle DAC$

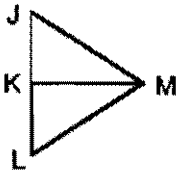


PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{AD}$	1. Given
2. $\overline{BC} \cong \overline{DC}$	2. Given
3. $\overline{AC} \cong \overline{AC}$	3 REFLEXIVE
4. $\Delta ABC \cong \Delta ADC$	4. SSS
5. $\angle BAC \cong \angle DAC$	5. CORRES-PARTS

1. Given: rt∠JKM
∠MJL ≅ ∠MLK

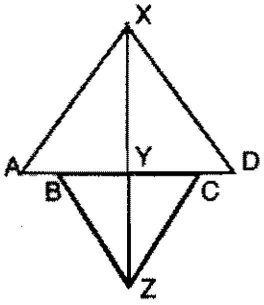
Goal: $\overline{JM} \cong \overline{LM}$



PROOF:

Statements:	Reasons:
1. rt∠JKM	1. Given
2. ∠MJL ≅ ∠MLK	2. Given

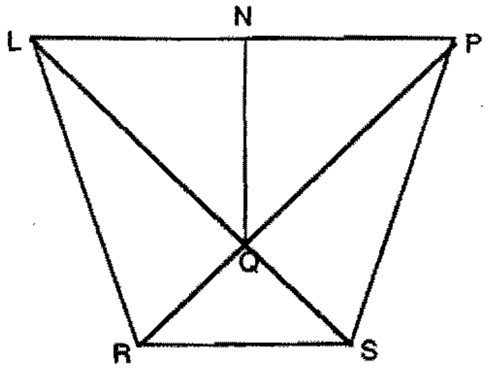
2. Given: $\overline{BY} \cong \overline{CY}$
$\angle AXY \cong \angle DXY$
$\angle XAY \cong \angle XDY$
Goal: $\angle BZY \cong \angle CZY$



PROOF:

Statements:	Reasons:
1. $\overline{BY} \cong \overline{CY}$	1. Given
2. $\angle AXY \cong \angle DXY$	2. Given
3. $\angle XAY \cong \angle XDY$	3. Given

3. Given: $\angle LQN \cong \angle PQN$
$\angle RLP \cong \angle SPL$
$\overline{LR} \cong \overline{PS}$
Goal: rt$\angle LNQ$



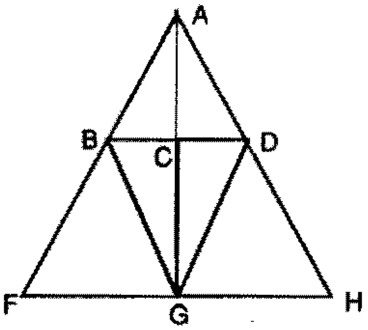
PROOF:

Statements:	Reasons:
1. $\angle LQN \cong \angle PQN$	1. Given
2. $\angle RLP \cong \angle SPL$	2. Given
3. $\overline{LR} \cong \overline{PS}$	3. Given

EXTRA CREDIT

Remember: your reasons should only be rules which appear on the rule summary sheet.

4. Given: $\overline{FG} \cong \overline{GH}$
$\overline{BC} \cong \overline{CD}$
$\angle FBG \cong \angle GDH$
$\angle BGF \cong \angle DGH$
Goal: $\angle ABC \cong \angle ADC$



PROOF:

Statements:	Reasons:
1. $\overline{FG} \cong \overline{GH}$	1. Given
2. $\overline{BC} \cong \overline{CD}$	2. Given
3. $\angle FBG \cong \angle GDH$	3. Given
4. $\angle ABC \cong \angle ADC$	4. Given

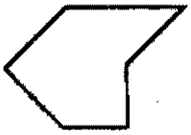
Name Date:

HIDDEN FIGURES TEST — CF-1 (Rev.)

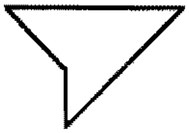
This is a test of your ability to tell which one of five simple figures can be found in a more complex pattern. At the top of each page in this test are five simple figures lettered A, B, C, D, and E. Beneath each row of figures is a page of patterns. Each pattern has a row of letters beneath it. Indicate your answer by putting an X through the letter of the figure which you find in the pattern.

NOTE: There is only one of these figures in each pattern, and this figure will always be right side up and exactly the same size as one of the five lettered figures.

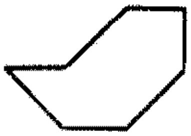
Now try these 2 examples.



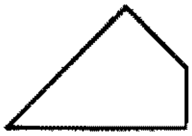
A



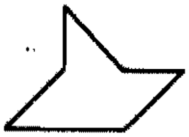
B



C

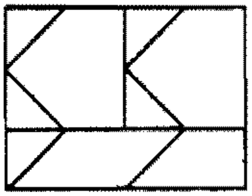


D



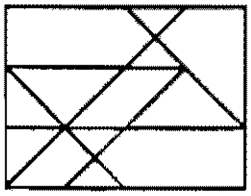
E.

I



A B C D E

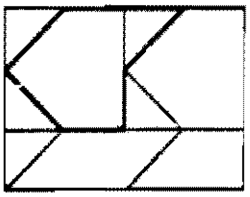
II



A B C D E

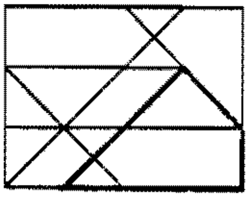
The figures below show how the figures are included in the problems. Figure A is in the first problem and figure D in the second.

I



[✓ A] B C D E

II



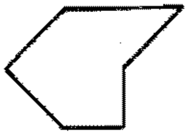
A B C [✓ D] E

Your score on this test will be the number marked correctly minus a fraction of the number marked incorrectly. Therefore, it will not be to your advantage to guess unless you are able to eliminate one or more of the answer choices as wrong.

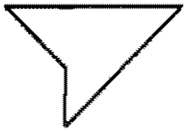
You will have 12 minutes for each of the two parts of this test. Each part has 2 pages. When you have finished Part 1, STOP. Please do not go on to Part 2 until you are asked to do so.

DO NOT TURN THIS PAGE UNTIL ASKED TO DO SO.

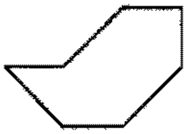
Part 1 (12 minutes)



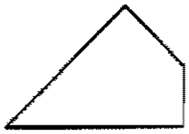
A



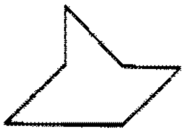
B



C

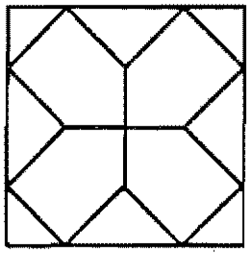


D



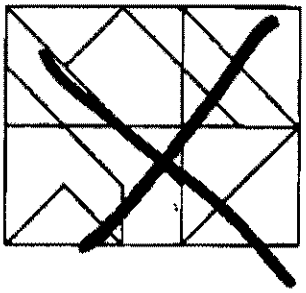
E

1\.



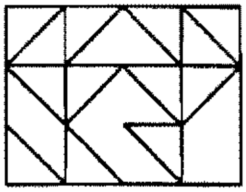
A B C D E

2\.



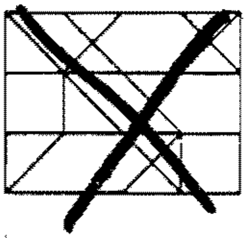
A B C D E

3\.



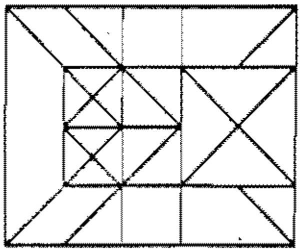
A B C D E

4\.



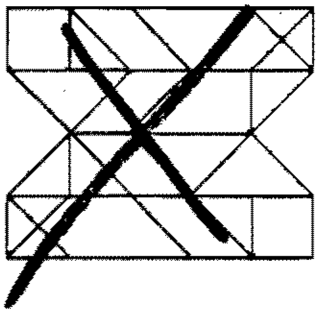
A B C D E

5\.



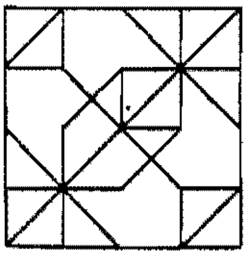
A B C D E

6\.



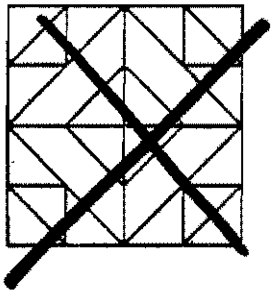
A B C D E

7\.



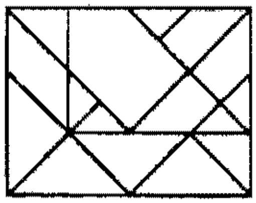
A B C D E

8\.



A B C D E

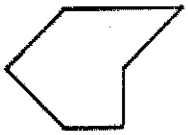
9\.

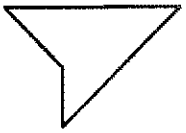


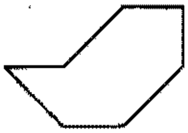
A B C D E

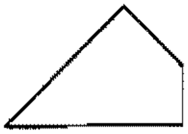
GO ON TO THE NEXT PAGE

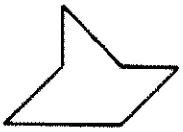
Part 1 (continued)


A

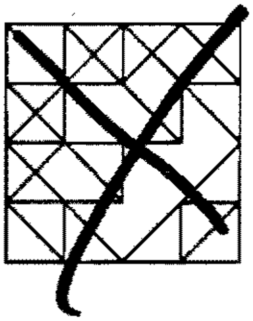

B


C


D

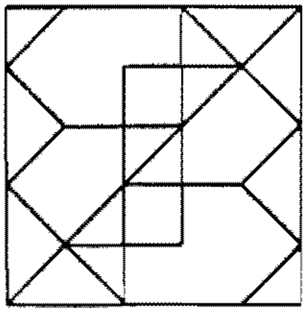

E

10.



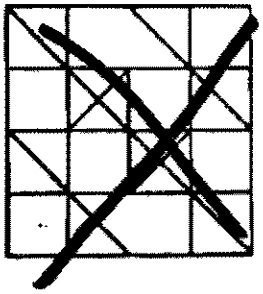
A B C D E

11.



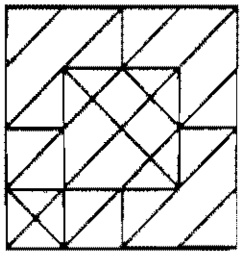
A B C D E

12.



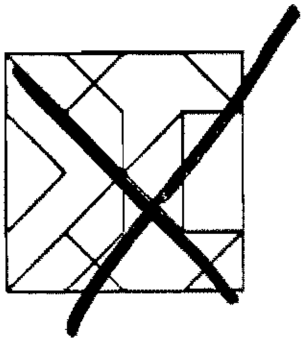
A B C D E

13.



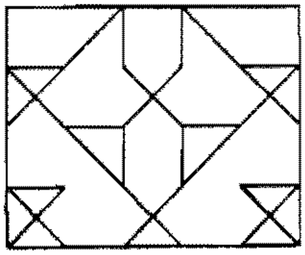
A B C D E

14.



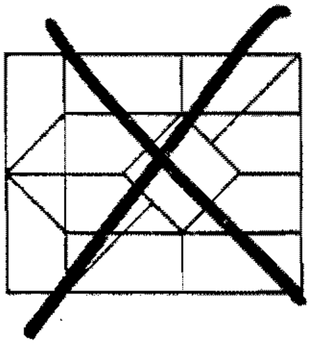
A B C D E

15.



A B C D E

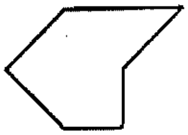
16.

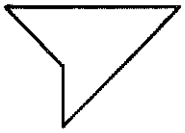


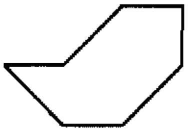
A B C D E

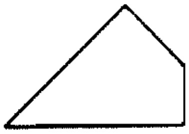
DO NOT TURN THIS PAGE UNTIL ASKED TO DO SO STOP.

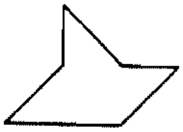
Part 2 (12 minutes)


A

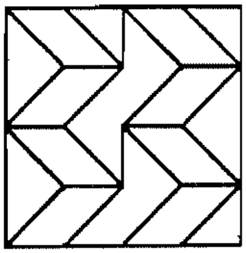

B


C


D

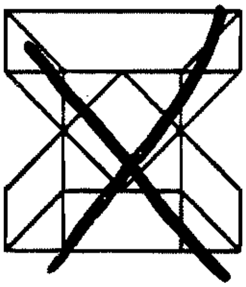

E

17.



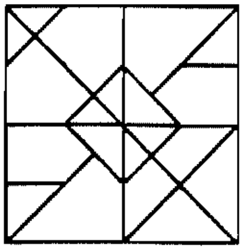
A B C D E

18.



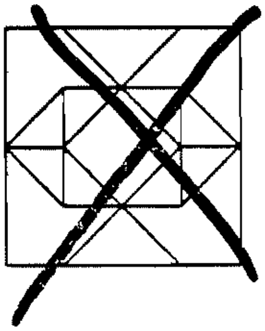
A B C D E

19.



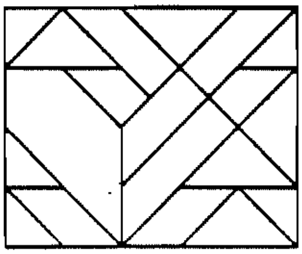
A B C D E

20.



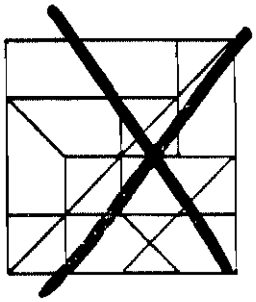
A B C D E

21.



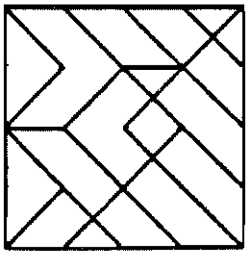
A B C D E

22.



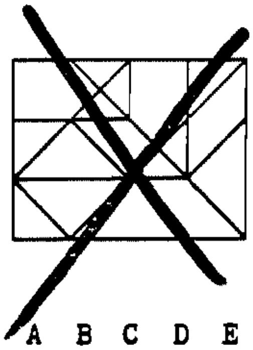
A B C D E

23.



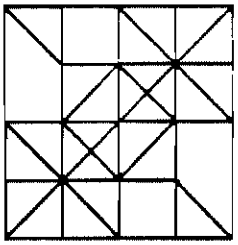
A B C D E

24.



A B C D E

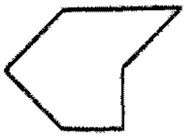
25.

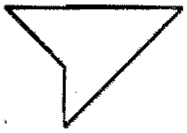


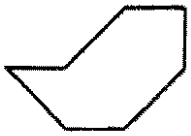
A B C D E

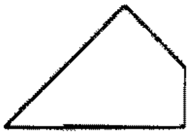
GO ON TO THE NEXT PAGE

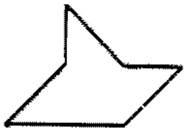
Part 2 (continued)

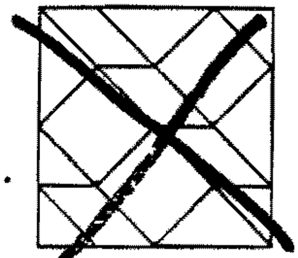

A


B

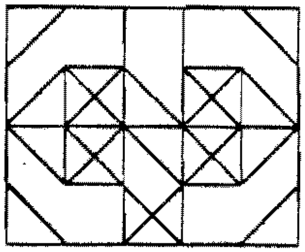

C


D

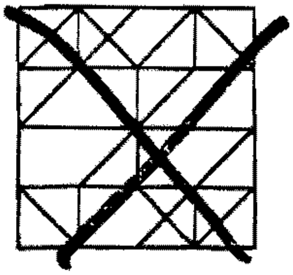

E

26. 

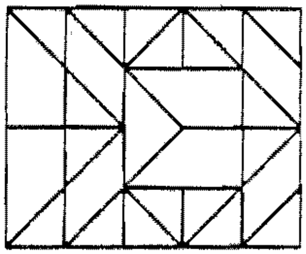
A B C D E

27. 

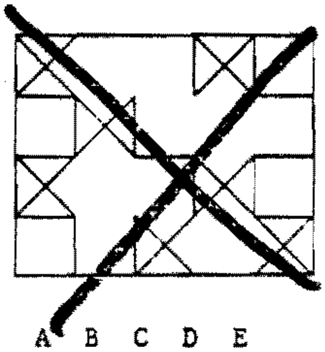
A B C D E

28. 

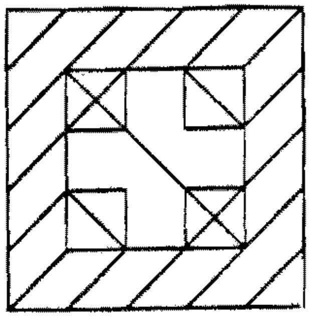
A B C D E

29. 

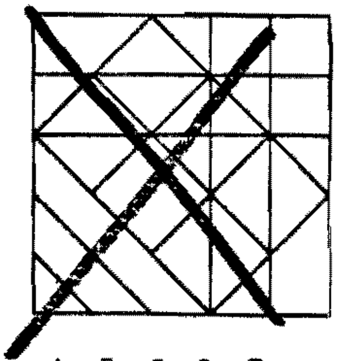
A B C D E

30. 

A B C D E

31. 

A B C D E

32. 

A B C D E

DO NOT GO BACK TO PART 1, AND
DO NOT GO ON TO ANY OTHER TEST UNTIL ASKED TO DO SO.

STOP

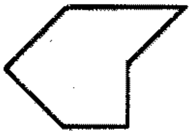
Name Date:

HIDDEN FIGURES TEST — CF-1 (Rev.)

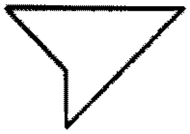
This is a test of your ability to tell which one of five simple figures can be found in a more complex pattern. At the top of each page in this test are five simple figures lettered A, B, C, D, and E. Beneath each row of figures is a page of patterns. Each pattern has a row of letters beneath it. Indicate your answer by putting an X through the letter of the figure which you find in the pattern.

NOTE: There is only one of these figures in each pattern, and this figure will always be right side up and exactly the same size as one of the five lettered figures.

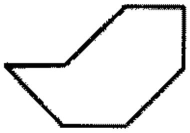
Now try these 2 examples.



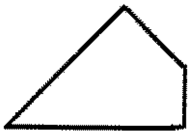
A



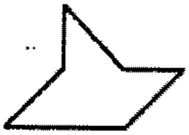
B



C

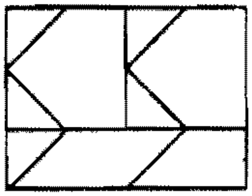


D



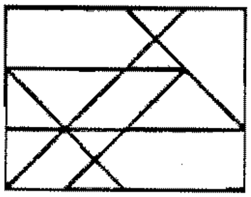
E.

I



A B C D E

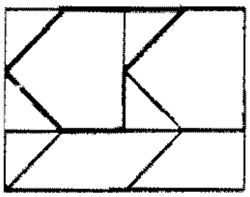
II



A B C D E

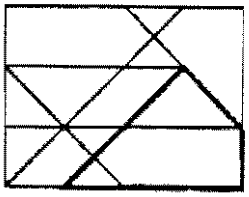
The figures below show how the figures are included in the problems. Figure A is in the first problem and figure D in the second.

I



[✓ A] B C D E

II



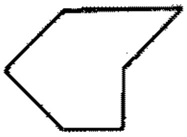
A B C [✓ D] E

Your score on this test will be the number marked correctly minus a fraction of the number marked incorrectly. Therefore, it will not be to your advantage to guess unless you are able to eliminate one or more of the answer choices as wrong.

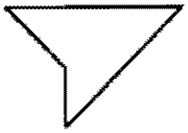
You will have 12 minutes for each of the two parts of this test. Each part has 2 pages. When you have finished Part 1, STOP. Please do not go on to Part 2 until you are asked to do so.

DO NOT TURN THIS PAGE UNTIL ASKED TO DO SO.

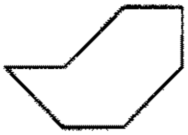
Part 1 (12 minutes)



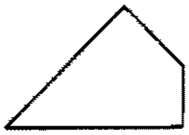
A



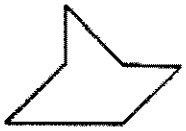
B



C

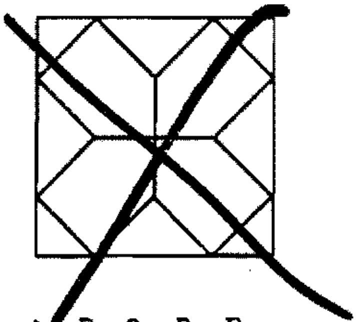


D



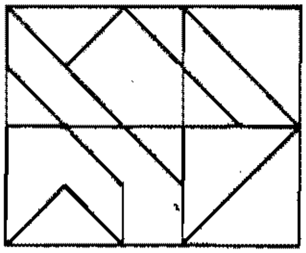
E

1.



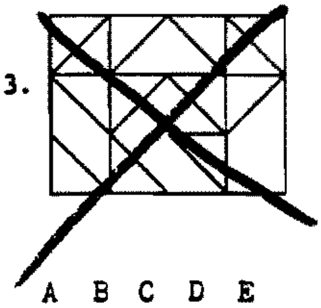
A B C D E

2.



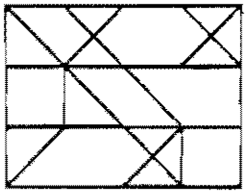
A B C D E

3.



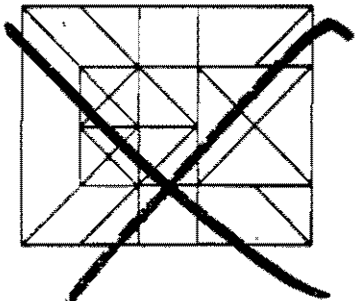
A B C D E

4.



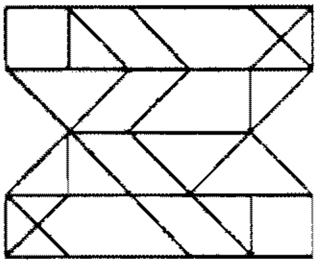
A B C D E

5.



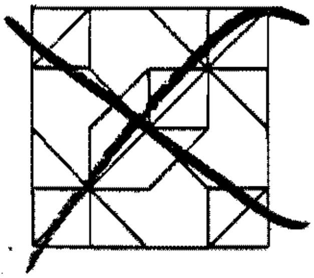
A B C D E

6.



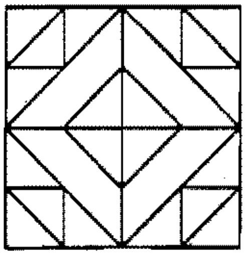
A B C D E

7.



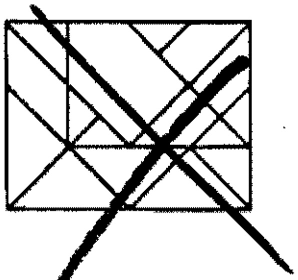
A B C D E

8.



A B C D E

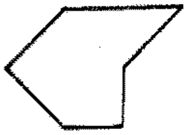
9.

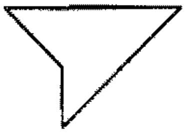


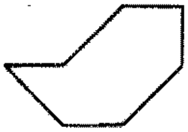
A B C D E

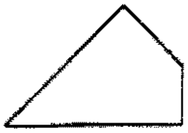
GO ON TO THE NEXT PAGE

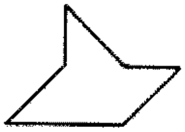
Part 1 (continued)


A

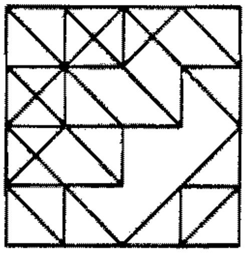

B


C


D

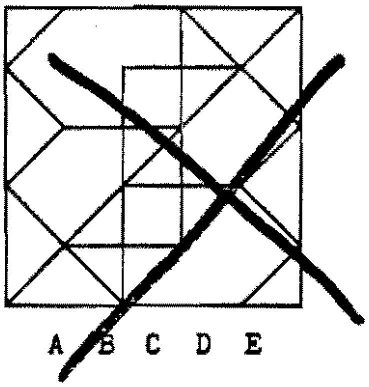

E

10.



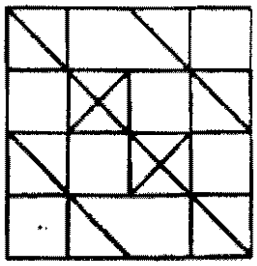
A B C D E

11.



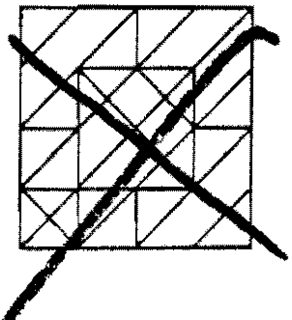
A B C D E

12.



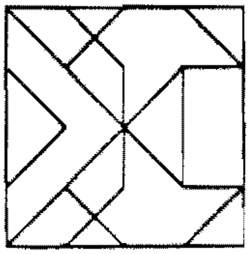
A B C D E

13.



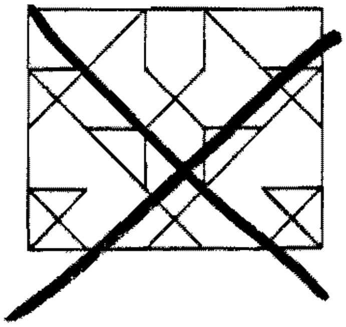
A B C D E

14.



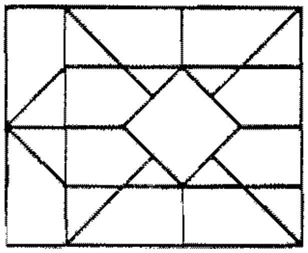
A B C D E

15.



A B C D E

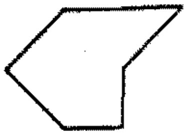
16.

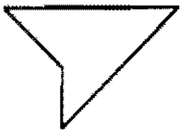


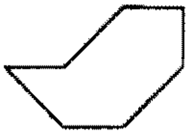
A B C D E

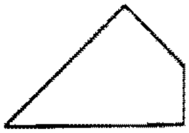
DO NOT TURN THIS PAGE UNTIL ASKED TO DO SO STOP.

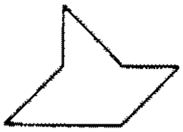
Part 2 (12 minutes)

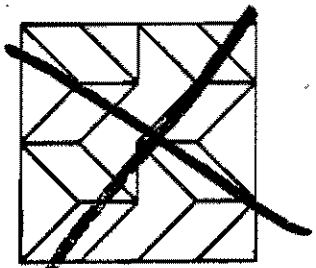

A


B

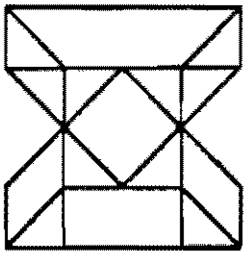

C


D

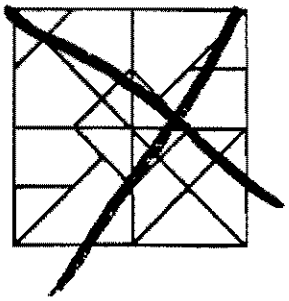

E

17. 

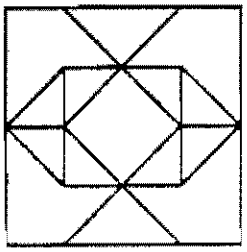
A B C D E

18. 

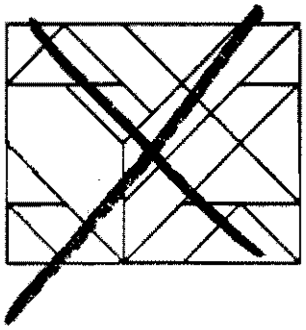
A B C D E

19. 

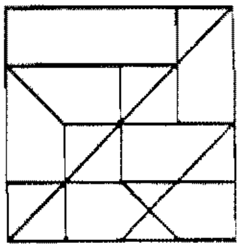
A B C D E

20. 

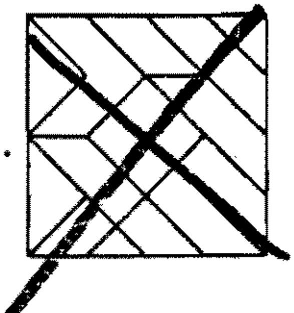
A B C D E

21. 

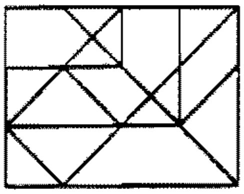
A B C D E

22. 

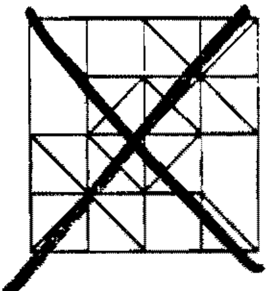
A B C D E

23. 

A B C D E

24. 

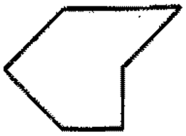
A B C D E

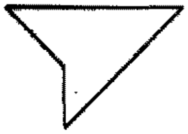
25. 

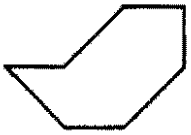
A B C D E

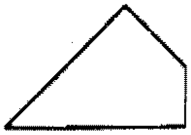
GO ON TO THE NEXT PAGE

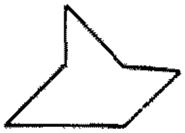
Part 2 (continued)


A

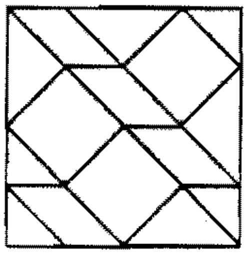

B


C


D

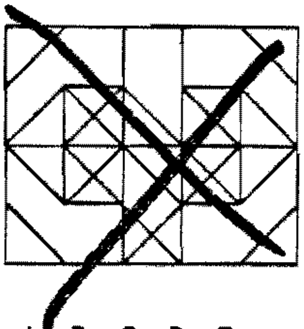

E

26.



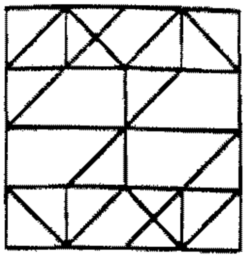
A B C D E

27.



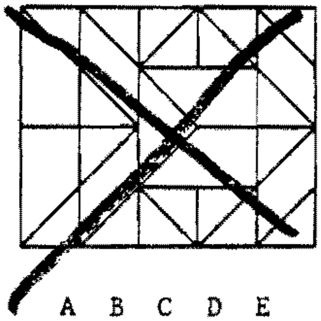
A B C D E

28.



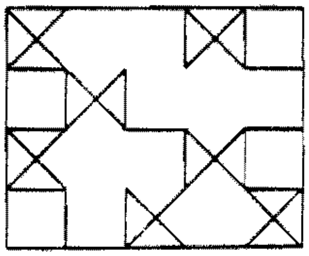
A B C D E

29.



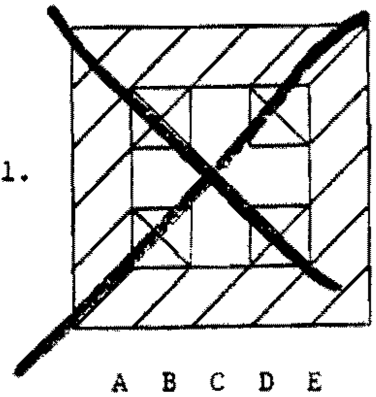
A B C D E

30.



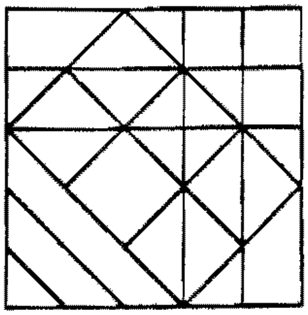
A B C D E

31.



A B C D E

32.



A B C D E

DO NOT GO BACK TO PART 1, AND
DO NOT GO ON TO ANY OTHER TEST UNTIL ASKED TO DO SO.

STOP.

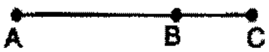
Truth Judgment Test: A

Name:______________ Date:____________

Id#:___ Start time:____________

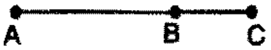
Circle one: Pre Post Finish time:____________

Below is series of yes-no questions which require you to do some geometric reasoning. The questions all request you to imagine certain relationships about a problem diagram, for example, in the diagram below imagine that $m\overline{AC} = 20$ and $m\overline{AB} = 5$.



Then you'll be asked whether some other relationship must be true, for example, in the situation above would $m\overline{BC}$ have to be 15, that is, must $m\overline{BC} = 15$? In this case the answer would be YES, since the two parts of $\overline{AC}$ must add up to 20. The only thing you can assume about the diagram is that all lines which appear straight are, in other words, *all points which appear colinear can be assumed to be colinear. However, the diagram may be misleading in other ways.* For example, in the diagram above, it doesn't look like $\overline{BC}$ is longer than $\overline{AB}$ even though the givens $m\overline{AC} = 20$ and $m\overline{AB} = 5$ indicate it must be. Don't let the diagram sway your decision. Its the relationships you are given that count. Here's the example above written in shorter form.

a. If $m\overline{AC} = 20$ and $m\overline{AB} = 5$, must $m\overline{BC} = 15$?

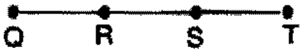


[✓ YES]

CAN'T TELL

In the next example, you CAN'T TELL for sure that $m\overline{RS} = 5$ since the information you are given $m\overline{QS} = 10$ and $m\overline{ST} = 5$ doesn't say anything about the exact position of point R between Q and S — for example, $m\overline{QR}$ could be 1 and $m\overline{RS}$ could be 9.

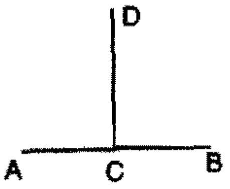
b. If $m\overline{QS} = 10$ and $m\overline{ST} = 5$, must $m\overline{RS} = 5$?



YES

[✓ CAN'T TELL]

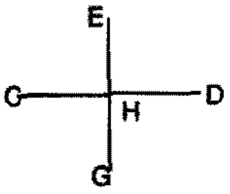
1. If $\angle BCD \cong \angle DCA$, must $\overline{AB} \perp \overline{CD}$?



YES

CAN'T
TELL

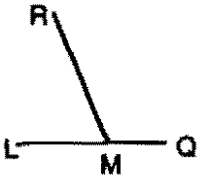
2. If rt$\angle EHD$, must $\overline{CH} \cong \overline{HD}$?



YES

CAN'T
TELL

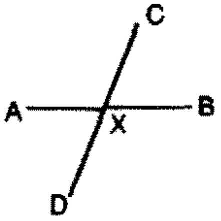
3. If $\overline{LM} \cong \overline{MQ}$, must $\overline{LQ} \perp \overline{RM}$?



YES

CAN'T
TELL

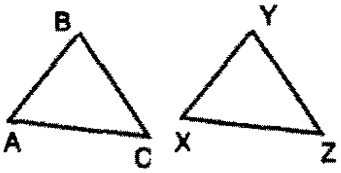
4. If rt$\angle AXC$, must $\angle AXC \cong \angle CXB$?



YES

CAN'T
TELL

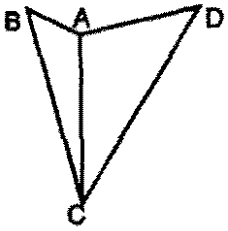
5. If $\overline{AB} \cong \overline{XY}$, $\overline{BC} \cong \overline{YZ}$, and $\overline{AC} \cong \overline{XZ}$, must $\angle ABC \cong \angle XYZ$?



YES

CAN'T
TELL

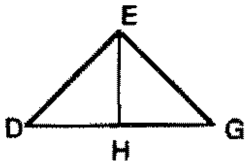
6. If $\angle ABC \cong \angle ADC$ and $\angle ACB \cong \angle ACD$, must $\triangle ABC \cong \triangle ADC$?



YES

CAN'T
TELL

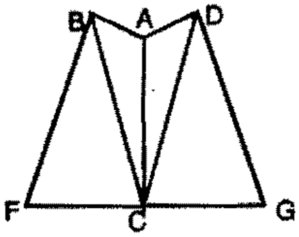
7. If $\triangle DEH \cong \triangle GEH$, must rt$\angle EHG$?



YES

CAN'T TELL

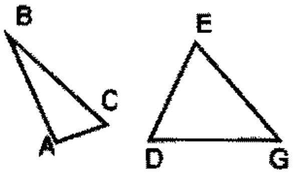
8. If $\angle BCF \cong \angle DCG$, $\angle CFB \cong \angle CGD$, and $\overline{BF} \cong \overline{DG}$, must $\angle BAC \cong \angle DAC$?



YES

CAN'T TELL

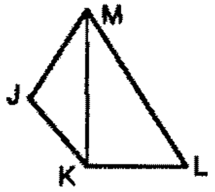
9. If $\overline{AB} \cong \overline{DE}$, $\angle ACB \cong \angle EGD$, and $\overline{BC} \cong \overline{EG}$, must $\overline{AC} \cong \overline{DG}$?



YES

CAN'T TELL

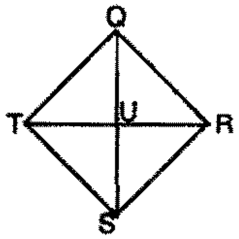
10. If rt$\angle MKL$ and $\overline{JK} \cong \overline{LK}$, must $\angle JMK \cong \angle LMK$?



YES

CAN'T TELL

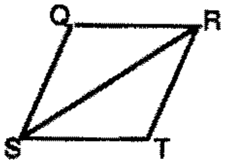
11. If $\angle TQU \cong \angle RQU$ and $\angle QTU \cong \angle QRU$, must $\angle STU \cong \angle SRU$?



YES

CAN'T TELL

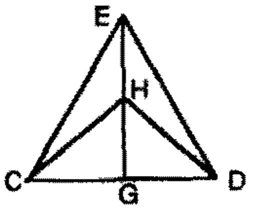
12. If $\angle QRS \cong \angle TSR$ and $\overline{QS} \cong \overline{ST}$, must $\triangle QRS \cong \triangle TSR$?



YES

CAN'T TELL

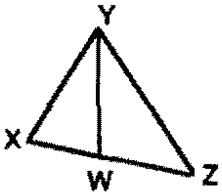
13. If rt$\angle CGE$ and $\overline{CG} \cong \overline{GD}$, must $\triangle ECH \cong \triangle EDH$?



YES

CAN'T TELL

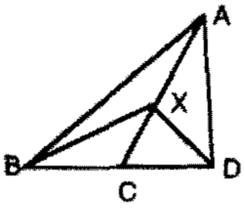
14. If $\triangle XYW \cong \triangle ZYW$, must $\overline{XW} \cong \overline{YW}$?



YES

CAN'T TELL

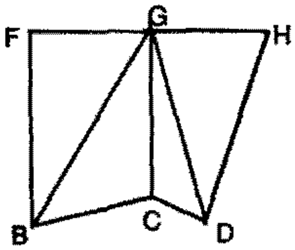
15. If rt$\angle BCA$ and $\overline{AX} \cong \overline{CX}$, must $\triangle ABX \cong \triangle ADX$?



YES

CAN'T TELL

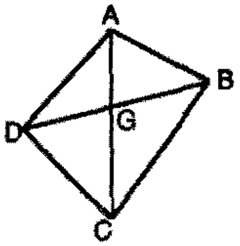
16. If $\angle FGB \cong \angle HGD$, $\angle GFB \cong \angle GHD$, $\overline{BC} \cong \overline{DC}$, and $\overline{BF} \cong \overline{DH}$, must $\angle GCB \cong \angle GCD$?



YES

CAN'T TELL

17. If $\angle DCG \cong \angle BCG$ and $\angle CDG \cong \angle CBG$, must $\angle DAG \cong \angle DCG$?



YES

CAN'T TELL

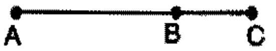
Truth Judgment Test: B

Name:_______________ Date:___________

Id#:___ Start time:___________

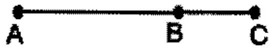
Circle one: Pre Post Finish time:___________

Below is series of yes-no questions which require you to do some geometric reasoning. The questions all request you to imagine certain relationships about a problem diagram, for example, in the diagram below imagine that $m\overline{AC} = 20$ and $m\overline{AB} = 5$.



Then you'll be asked whether some other relationship must be true, for example, in the situation above would $m\overline{BC}$ have to be 15, that is, must $m\overline{BC} = 15$? In this case the answer would be YES, since the two parts of $\overline{AC}$ must add up to 20. The only thing you can assume about the diagram is that all lines which appear straight are, in other words, *all points which appear colinear can be assumed to be colinear. However, the diagram may be misleading in other ways.* For example, in the diagram above, it doesn't look like $\overline{BC}$ is longer than $\overline{AB}$ even though the givens $m\overline{AC} = 20$ and $m\overline{AB} = 5$ indicate it must be. Don't let the diagram sway your decision. Its the relationships you are given that count. Here's the example above written in shorter form.

a. If $m\overline{AC} = 20$ and $m\overline{AB} = 5$, must $m\overline{BC} = 15$?

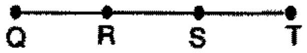


[✓ YES]

CAN'T TELL

In the next example, you CAN'T TELL for sure that $m\overline{RS} = 5$ since the information you are given $m\overline{QS} = 10$ and $m\overline{ST} = 5$ doesn't say anything about the exact position of point R between Q and S — for example, $m\overline{QR}$ could be 1 and $m\overline{RS}$ could be 9.

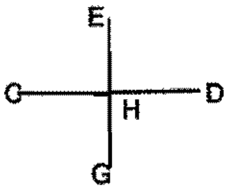
b. If $m\overline{QS} = 10$ and $m\overline{ST} = 5$, must $m\overline{RS} = 5$?



YES

[✓ CAN'T TELL]

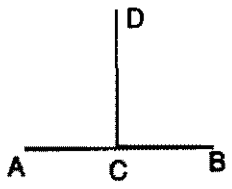
1. If rt$\angle EHD$, must $\angle CHE \cong \angle DHE$?



YES

CAN'T TELL

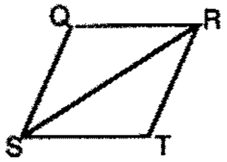
2. If $\overline{AC} \cong \overline{CB}$, must $\overline{AB} \perp \overline{CD}$?



YES

CAN'T TELL

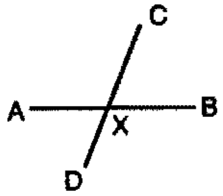
3. If $\angle QRS \cong \angle TSR$ and $\angle QSR \cong \angle TRS$, must $\triangle QRS \cong \triangle TSR$?



YES

CAN'T TELL

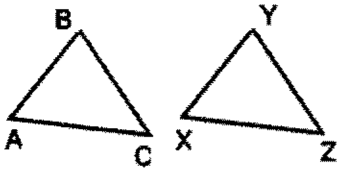
4. If rt$\angle AXC$, must $\overline{CX} \cong \overline{XD}$?



YES

CAN'T TELL

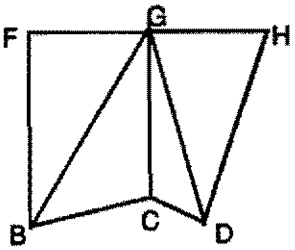
5. If $\overline{AB} \cong \overline{XY}$, $\angle ACB \cong \angle XZY$, and $\overline{BC} \cong \overline{YZ}$, must $\angle ABC \cong \angle XYZ$?



YES

CAN'T TELL

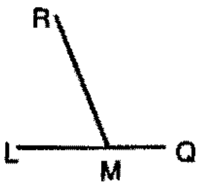
6. If $\angle FGB \cong \angle HGD$, $\angle GFB \cong \angle GHD$, and $\overline{BF} \cong \overline{DH}$, must $\angle GCB \cong \angle GCD$?



YES

CAN'T TELL

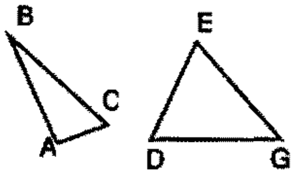
7. If $\angle QMR \cong \angle LMR$, must $\overline{LQ} \perp \overline{RM}$?



YES

CAN'T TELL

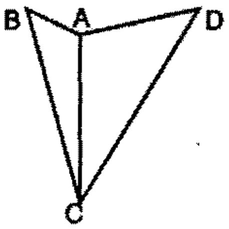
8. If $\overline{AB} \cong \overline{DE}$, $\overline{BC} \cong \overline{EG}$, and $\overline{AC} \cong \overline{EG}$, must $\angle ABC \cong \angle DEG$?



YES

CAN'T TELL

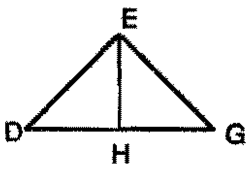
9. If $\angle BCA \cong \angle DCA$ and $\overline{BC} \cong \overline{AD}$, must $\triangle ABC \cong \triangle ADC$?



YES

CAN'T TELL

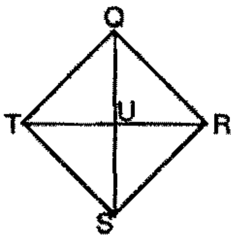
10. If $\triangle DEH \cong \triangle GEH$, must $\overline{DH} \cong \overline{EH}$?



YES

CAN'T TELL

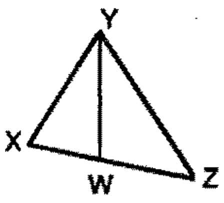
11. If $\angle TQU \cong \angle RQU$ and $\angle QTU \cong \angle QRU$, must $\angle TSU \cong \angle TQU$?



YES

CAN'T TELL

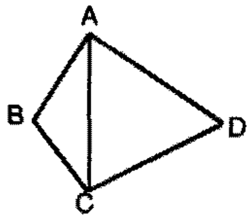
12. If $\triangle XYW \cong \triangle ZYW$, must rt$\angle YWX$?



YES

CAN'T TELL

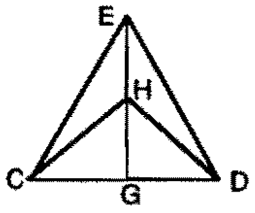
13. If rt$\angle ACD$ and $\overline{BC} \cong \overline{DC}$, must $\angle BAC \cong \angle DAC$?



YES

CAN'T TELL

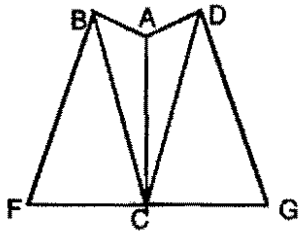
14. If rt$\angle CGE$ and $\overline{EH} \cong \overline{GH}$, must $\triangle ECH \cong \triangle EDH$?



YES

CAN'T TELL

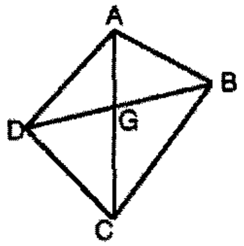
15. If $\angle BCF \cong \angle DCG$, $\angle CFB \cong \angle CGD$, $\overline{BA} \cong \overline{DA}$, and $\overline{BF} \cong \overline{DG}$, must $\angle BAC \cong \angle DAC$?



YES

CAN'T TELL

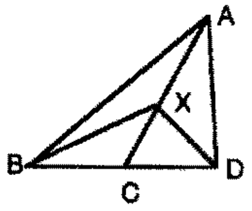
16. If $\angle DCG \cong \angle BCG$ and $\angle CDG \cong \angle CBG$, must $\angle ADG \cong \angle ABG$?



YES

CAN'T TELL

17. If rt$\angle BCA$ and $\overline{BC} \cong \overline{CD}$, must $\triangle ABX \cong \triangle ADX$?



YES

CAN'T TELL

Proof Checking Test: A

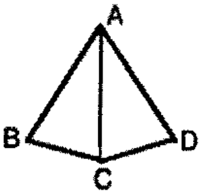
Name:______________ Date:____________
Id#:___ Start time:____________
Circle one: Pre Post Finish time:____________

For each proof that follows check each line to see if it follows from the preceeding lines. If it does, put "OK" after the line. If it does not, put "doesn't follow" after the line and indicate why. In the example below, line 3 is OK, but line 4 is not. There must be a congruent triangle statement preceeding the use of **CORRES-PARTS**, but the statement **$\triangle ABC \cong \triangle ADC$** is missing.

NOTE: There may be more than one error in these proofs.

0. Given: $\overline{AB} \cong \overline{AD}$
$\overline{BC} \cong \overline{DC}$

Goal: $\angle BAC \cong \angle DAC$

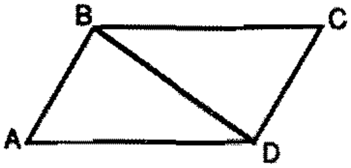


PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{AD}$	**1. Given**
2. $\overline{BC} \cong \overline{DC}$	**2. Given**
3. $\overline{AC} \cong \overline{AC}$	**3. REFLEXIVE** ok
4. $\angle BAC \cong \angle DAC$	**4. CORRES-PARTS** doesn't follow: missing $\triangle ABC \cong \triangle ADC$

1. Given: $\overline{AD} \cong \overline{BC}$
$\overline{AB} \cong \overline{DC}$
$\angle ABD \cong \angle CDB$

Goal: $\angle BAD \cong \angle DCB$

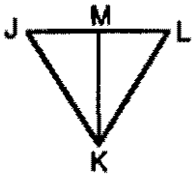


PROOF:

Statements:	Reasons:
1. $\overline{AD} \cong \overline{BC}$	1. Given
2. $\overline{AB} \cong \overline{DC}$	2. Given
3. $\angle ABD \cong \angle CDB$	3. Given
4. $\triangle ABD \cong \triangle CDB$	4. SAS
5. $\angle BAD \cong \angle DCB$	5. CORRES-PARTS

2. Given: rt $\angle JMK$
$\overline{JK} \cong \overline{LK}$

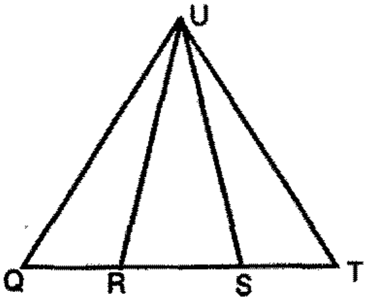
Goal: $\triangle JKM \cong \triangle LKM$



PROOF:

Statements:	Reasons:
1. rt $\angle JMK$	1. Given
2. $\overline{JK} \cong \overline{LK}$	2. Given
3. $\angle JMK \cong \angle LMK$	3. CONG-ADJ-ANGS
4. $\overline{MK} \cong \overline{MK}$	4. REFLEXIVE
5. $\triangle JKM \cong \triangle LKM$	5. SAS

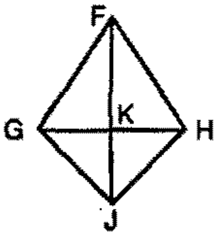
3. Given: ∠QUS ≅ ∠TUR
$\overline{QS} \cong \overline{TR}$
∠URS ≅ ∠USR
∠QUS ≅ ∠TUR
Goal: ∠URQ ≅ ∠UST



PROOF:

Statements:	Reasons:
1. ∠QUS ≅ ∠TUR	1. Given
2. $\overline{QS} \cong \overline{TR}$	2. Given
3. ∠URS ≅ ∠USR	3. Given
4. ∠QUS ≅ ∠TUR	4. Given
5. ∠UQR ≅ ∠UTS	5. AAS
6. $\overline{QU} \cong \overline{TU}$	6. CORRES-PARTS
7. △UQR ≅ △UTS	6. AAS
8. ∠URQ ≅ ∠UST	7. CORRES-PARTS

4. Given: $\overline{GF} \cong \overline{FH}$
∠GFK ≅ ∠HFK
Goal: ∠JGK ≅ ∠JHK



PROOF:

Statements:	Reasons:
1. $\overline{GF} \cong \overline{FH}$	1. Given
2. ∠GFK ≅ ∠HFK	2. Given
3. $\overline{FK} \cong \overline{FK}$	3. REFLEXIVE
4. △GFK ≅ △HFK	4. ASA
5. ∠FKG ≅ ∠FKH	5. CORRES-PARTS
6. ∠JKG ≅ ∠JKH	6. CONG-ADJ-ANGS
7. $\overline{KJ} \cong \overline{KJ}$	7. REFLEXIVE
8. △GKJ ≅ △HKJ	8. SAS
9. ∠JGK ≅ ∠JHK	9. CORRES-PARTS

Proof Checking Test: B

Name:______________
Id#:___
Circle one: Pre Post

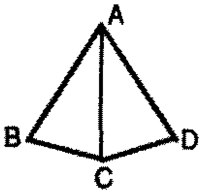
Date:___________
Start time:___________
Finish time:___________

For each proof that follows check each line to see if it follows from the preceeding lines. If it does, put "OK" after the line. If it does not, put "doesn't follow" after the line and indicate why. In the example below, line 3 is OK, but line 4 is not. There must be a congruent triangle statement preceeding the use of **CORRES-PARTS**, but the statement **$\triangle ABC \cong \triangle ADC$** is missing.

NOTE: There may be more than one error in these proofs.

0. Given: $\overline{AB} \cong \overline{AD}$
$\overline{BC} \cong \overline{DC}$

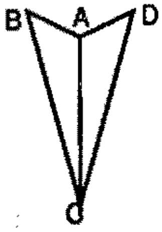
Goal: $\angle BAC \cong \angle DAC$



PROOF:

Statements:	Reasons:
1. $\overline{AB} \cong \overline{AD}$	**1. Given**
2. $\overline{BC} \cong \overline{DC}$	**2. Given**
3. $\overline{AC} \cong \overline{AC}$	**3. REFLEXIVE** OK
4. $\angle BAC \cong \angle DAC$	**4. CORRES-PARTS** doesn't follow; missing $\Delta ABC \cong \Delta ADC$

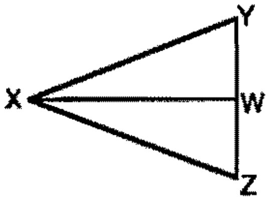
1. Given: $\overline{BC} \cong \overline{DC}$
$\overline{AB} \cong \overline{AD}$
$\angle BAC \cong \angle DAC$
Goal: $\angle ABC \cong \angle ADC$



PROOF:

Statements:	Reasons:
1. $\overline{BC} \cong \overline{DC}$	1. Given
2. $\overline{AB} \cong \overline{AD}$	2. Given
3. $\angle BAC \cong \angle DAC$	3. Given
4. $\triangle ABC \cong \triangle ADC$	4. SAS
5. $\angle ABC \cong \angle ADC$	5. CORRES-PARTS

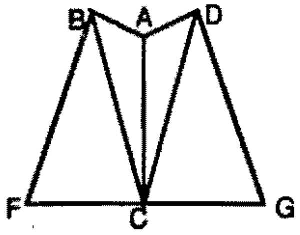
2. Given: $\overline{YW} \cong \overline{WZ}$
$\angle YXW \cong \angle ZXW$
Goal: rt $\angle XWZ$



PROOF:

Statements:	Reasons:
1. $\overline{YW} \cong \overline{WZ}$	1. Given
2. $\angle YXW \cong \angle ZXW$	2. Given
3. $\overline{XW} \cong \overline{XW}$	3. REFLEXIVE
4. $\triangle YXW \cong \triangle ZXW$	4. SAS
5. $\angle XWY \cong \angle XWZ$	5. CORRES-PARTS
6. rt $\angle XWZ$	6. DEF-PERP

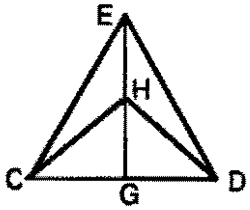
3. Given: $\angle BCF \cong \angle DCG$
$\angle CFB \cong \angle CGD$
$\overline{BA} \cong \overline{DA}$
$\overline{BF} \cong \overline{DG}$
Goal: $\angle BAC \cong \angle DAC$



PROOF:

Statements:	Reasons:
1. $\angle BCF \cong \angle DCG$	1. Given
2. $\angle CFB \cong \angle CGD$	2. Given
3. $\overline{BA} \cong \overline{DA}$	3. Given
4. $\overline{BF} \cong \overline{DG}$	4. Given
5. $\triangle BCF \cong \triangle DCG$	5. ASA
6. $\overline{BC} \cong \overline{DC}$	6. CORRES-PARTS
7. $\overline{AC} \cong \overline{AC}$	7. REFLEXIVE
8. $\angle BAC \cong \angle DAC$	8. SSS

4. Given: rt$\angle CGE$
$\overline{CG} \cong \overline{GD}$
Goal: $\angle ECH \cong \angle EDH$



PROOF:

Statements:	Reasons:
1. rt$\angle CGE$	1. Given
2. $\overline{CG} \cong \overline{GD}$	2. Given
3. $\overline{EG} \cong \overline{EG}$	3. REFLEXIVE
4. $\angle CGE \cong \angle DGE$	4. CONG-ADJ-ANGS
5. $\triangle CGE \cong \triangle DGE$	5. ASA
6. $\overline{CE} \cong \overline{DE}$	6. CORRES-PARTS
7. $\overline{EH} \cong \overline{EH}$	7. REFLEXIVE
8. $\triangle CEH \cong \triangle DEH$	8. SSS
9. $\angle ECH \cong \angle EDH$	9. CORRES-PARTS