

**THE IMPACT OF DIFFERENT PROOF STRATEGIES
ON LEARNING GEOMETRY THEOREM PROVING**

by

Noboru Matsuda

BS in Education, Tokyo Gakugei University, 1985

MS in Education, Tokyo Gakugei University, 1988

Submitted to the Graduate Faculty of

Intelligent Systems Program in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2004

UNIVERSITY OF PITTSBURGH
FACULTY OF ARTS AND SCIENCES

This dissertation was presented

by

Noboru Matsuda

It was defended on

November 10, 2004

and approved by

Dr. Peter Brusilovsky (Intelligent Systems Program)

Dr. James G. Greeno (School of Education, University of Pittsburgh)

Dr. Kenneth R. Koedinger (School of Computer Science, Carnegie Mellon University)

Dr. Christian Schunn (Intelligent Systems Program)

Dr. Kurt VanLehn (Intelligent Systems Program)
Dissertation Director

THE IMPACT OF DIFFERENT PROOF STRATEGIES ON LEARNING GEOMETRY THEOREM PROVING

Noboru Matsuda, PhD

University of Pittsburgh, 2004

Two problem solving strategies, forward chaining and backward chaining, were compared to see how they affect students' learning of geometry theorem proving with construction. It has been claimed that backward chaining is inappropriate for novice students due to its complexity. On the other hand, forward chaining may not be appropriate either for this particular task because it can explode combinatorially. In order to determine which strategy accelerates learning the most, an intelligent tutoring system was developed. It is unique in two ways: (1) It has a fine grained cognitive model of proof-writing, which captured both observable and unobservable inference steps. This allows the tutor to provide elaborate scaffolding. (2) Depending on the student's competence, the tutor provides a variety of scaffolding from showing precise steps to just prompting students for a next step. In other words, the students could learn proof-writing through both worked-out examples (by observing a model of proof-writing generated by the tutor) and problem solving (by writing proofs by themselves). 52 students were randomly assigned to one of the tutoring systems. They solved 11 geometry proof problems with and without construction with the aid from the intelligent tutor. The results show that (1) the students who learned forward chaining showed better performance on proof-writing than those who learned backward chaining, (2) both forward and backward chaining conditions wrote wrong proofs equally frequently, (3) both forward and backward chaining conditions seldom wrote redundant or wrong statements when they wrote correct proofs, (4) the major reason for

the difficulty in applying backward chaining lay in the assertion of premises as unjustified propositions (i.e., subgoaling). These results provide theoretical implications for the design of tutoring systems for problem solving.

TABLE OF CONTENTS

1. Introduction.....	1
2. Issues in Learning and Teaching Geometry Theorem Proving.....	3
2.1. Task: Geometry Theorem Proving with a Two-column Proof.....	3
2.2. Theorem Proving with Construction.....	6
2.2.1. Theorem proving with construction as a state-space search.....	6
2.2.2. Finding a useful postulate for construction.....	7
2.2.3. Identifying target segments by partial overlapping	8
2.3. Students' Difficulties in Learning Proof Writing	8
2.4. Teaching and Learning Forward / Backward Chaining.....	9
3. The Advanced Geometry Tutor	14
3.1. Theoretical Implications in Teaching Geometry Theorem Proving	14
3.1.1. Learning from worked-out examples and problem solving.....	15
3.1.2. Articulating tacit knowledge.....	16
3.1.3. Teaching operationalization.....	17
3.1.4. Summary	18
3.2. Overview of the Advanced Geometry Tutor	19
3.3. A Cognitive Task Analysis of Geometry Theorem Proving.....	20
3.4. The Solution Graph.....	22
3.5. The Scaffolding Strategy	23
3.6. The AGT Learning Environment.....	25
3.6.1. GUI components	26
3.6.2. Students' activities and tutor's behavior.....	28
4. Evaluation of AGT.....	29
4.1. Subjects and Design.....	29
4.2. Procedure	29
4.3. Materials	29
4.3.1. The booklet	30
4.3.2. Pre- and post-test.....	30
4.3.3. Proof problems used in the study.....	31
5. Results.....	32
5.1. Learning Time.....	33
5.2. Pre- and Post-Test Scores	34
5.3. Learning the Postulates	39

5.3.1.	Improvement of students' performance in postulate applications	39
5.3.2.	Pre- and post-test difference	42
5.4.	Strategy Used to write Proofs in the Post-test	45
5.5.	Students' Performance in Proof Writing.....	46
5.5.1.	Basic definitions for coding schema.....	47
5.5.2.	Analysis of proofs.....	49
5.5.3.	Analysis of proof statements.....	52
5.5.4.	False subgoaling in backward chaining	56
5.6.	Analysis of Postulate Applications	57
6.	Discussion.....	63
6.1.	Learning Domain Concept does not secure Proof-writing Skills	63
6.2.	Difficulty in Thinking Backwards: Subgoaling.....	64
6.3.	Implication of the Cognitive Load Theory	65
6.4.	Complexity of the Search.....	67
6.5.	Concluding Remarks.....	69
7.	Future Work	70
7.1.	Implications for a Tutor Design.....	70
7.2.	Research Questions for Future Studies	71
	REFERENCE.....	73
	APPENDIX A: An Example of Scaffolding of the Forward AGT.....	76
	APPENDIX B: An Example of Scaffolding of Backward AGT	87
	APPENDIX C: The Geometry Booklet (Backward Tutor)	97
	APPENDIX D: The Geometry Booklet (Forward Tutor).....	107
	APPENDIX E: Test-A (Backward Tutor)	117
	APPENDIX F: Test-B (Backward Tutor).....	125
	APPENDIX G: Test-A (Forward Tutor).....	133
	APPENDIX H: Test-B (Forward Tutor).....	141
	APPENDIX I: Problems used in Tutoring Sessions	149
	APPENDIX J: Learning Curves	151
	APPENDIX K: Example of Proofs written in an Inconsistent Strategy.....	156
	APPENDIX L: Cognitive Model of Proof Writing	159

LIST OF TABLES

Table 2.1: Comparison of search complexity	12
Table 2.2: Search complexity with construction supported by backward chaining	13
Table 4.1: Proof problems used in the study.....	32
Table 5.1: 2x2 Contingency Table on the correctness of the proofs	51
Table 5.2: 2 x 2 Contingency tables on on-path and off-path proof statements	53
Table 5.3: 2 x 2 Contingency table on on-path and off-path statements comparing construction vs. non-construction problems	54
Table 5.4: 2 x 2 Contingency tables on on-path and off-path statements comparing construction vs. non-construction problems in correct and incorrect proofs.....	55
Table 5.5: A 2 x 3 Contingency table on the type of propositions	59
Table 5.6: A 2 x 4 Contingency table on the type of justification	61
Table 5.7: A 2 x 4 Contingency table on the use of premises	62

LIST OF FIGURES

Figure 2.1: An example of a proof table	5
Figure 3.1: Cognitive model of backward inference with construction.....	22
Figure 3.2: The Advanced Geometry Tutor.....	26
Figure 3.3: The equation builder.....	28
Figure 5.1: Average time spent on a problem.....	33
Figure 5.2: Pre- and post-test scores	35
Figure 5.3: Average subscores on fill-in-blank questions	36
Figure 5.4: Average subscores on proof-writing questions (excluding question #5)	36
Figure 5.5: ATI analysis for proof-writing scores on the post-test.....	37
Figure 5.6: Mean scores on proof-writing for non-construction and construction problems	38
Figure 5.7: Average duration for postulate applications.....	40
Figure 5.8: Average number of errors made during single postulate application.....	41
Figure 5.9: Difference between tutor conditions in accuracy of postulate applications.....	43
Figure 5.10: Difference in accuracy of postulate applications between tests	44
Figure 5.11: Number of proof written in opposite strategy (Max=52).....	46
Figure 5.12: Classification of incorrect proofs	49
Figure 5.13: Portion of incorrect proofs on the post-test.....	50
Figure 5.14: Classification of proof statements	52
Figure 5.15: Type of propositions.....	59
Figure 5.16: Type of justification	60
Figure 5.17: Usage of premise	62

PREFACE

I thank my advisor Kurt VanLehn for his kind and considerable intellectual support. Thanks also my committee members Peter Brusilovsky, Christian Schunn, James Greeno, and Ken Koedinger. I have learned a lot from every single meeting with those great scholars.

Also, I thank Sara Masters, Eri Seta, and Kwangsu Cho for their patience during early pilot-testing as well as very many constructive comments on the software. Especially, without Sara's thorough correction in the computerized tutor's broken English, the experiment could not run properly.

Thanks to my friends in Kurt VanLehn and Micki Chi's research groups for intellectual and personal support throughout my grad life, especially Chas Murray, Chad Lane, Mike Ringenberg, Min Chi, and Stephanie Siler.

Finally and most importantly, I cordially thank to my beloved wife, Chizuru Matsuda, and beautiful kids, Rina and Reo. Without Chizuru's devoted support and deep understanding, I could not even complete this study. The kids always made me happy and eased all sort of pains. I thank my parents for their love and support.

1. Introduction

Geometry theorem proving is one of the most challenging subjects for students to learn. When it requires *construction* as a part of a proof, the difficulty of the problem drastically increases. The term “construction” here means drawing additional lines and points onto the problem figure using a compass and a straightedge. In this study, we attempted to determine effective instruction to teach proof writing for geometry theorems that require construction. The target students in this study are at an intermediate level. They know the geometry knowledge necessary to write some proofs and learn more postulates, but their problem solving strategy is not fully stabilized, hence they need a tutor’s aid while they practice problem solving.

The difficulties of geometry theorem proving with construction may lie in a lack of knowledge about selecting construction, namely, the lack of an *algorithm* to find appropriate construction. Indeed, in an educational context, construction is thought to be “creative” and “intuitive” hence best taught as heuristics (see for example, Polya, 1957).

When people encounter a problem that seems unfamiliar like finding a proof with construction, they tend to use the so-called *weak methods*. Two major weak methods for geometry theorem proving are *forward chaining* and *backward chaining* (Newell & Simon, 1972). When applied to theorem proving, the former calculate a *deductive closure*, which is a set of true propositions that hold within a given configuration of the theorem to be proven. Starting from given propositions, which constitute an initial database, forward chaining finds all propositions that are logically derived from the database, adds them to the database, and repeats this cycle until the to-be-proved goal is eventually added to the database. Backward chaining, on the other hand, starts from the goal to be proved, identifies premises that support the goal, and proves that those premises also hold in the given problem configuration. Construction can be

taken place at anytime during these weak methods. For forward chaining, one can draw arbitrarily many lines to draw new geometric objects (such as triangles, quadrangles, etc) and draw arbitrary many inferences from the modified configuration, which in turn blows up the database. For backward chaining, one can pick an arbitrary postulate that has a consequence that matches with the goal to prove, draw segments so that the postulate would apply, and subgoal the postulate's premises, which could lead one to infinite subgoaling.

Because these two methods are the most straightforward strategies, the students might benefit from using them as a vehicle for learning geometry theorem proving. However, there has been no theoretical account provided to predict which problem-solving strategy facilitates the students' ability to write geometry proofs (see Section 2.4 for a review on teaching and learning these weak methods). Thus, the current study addresses the following research questions:

- (1) Given a fixed set of training problems including construction problems, which problem-solving strategy, forward vs. backward chaining, facilitates students' ability to write proofs?
- (2) If there is a difference in the learning gain between the forward and backward chaining groups, then what seems to cause that difference?

To answer these questions, we built two versions of an intelligent tutoring system; one teaches forward chaining and the other teaches backward chaining. We then assigned students to each tutoring condition, let them learn theorem proving under the assistance of the intelligent tutor, and compared their performance on pre- and post-tests as well as during the tutoring sessions.

The contributions of the current dissertation include (1) gaining a better understanding of the educational benefits of teaching geometry theorem proving with a certain problem solving

strategy (i.e., forward or backward chaining) particularly as it relates to construction, (2) revealing student's difficulties in working forwards and backwards, (3) providing guidance for future designs of learning environments to support students in learning geometry theorem proving.

Section 2 provides a review of the task, namely, geometry theorem proving with construction. It also discusses students' difficulties in learning proof writing, and the theoretical implications in teaching geometry theorem proving. Section 2 also provides a review on teaching and learning forward / backward chaining. Section 3 then shows the structure of our geometry tutor, the Advanced Geometry Tutor (AGT). It first summarizes cognitive theories that we know best to design out tutor. Detailed explanations of the underlying cognitive model, scaffolding strategy, and graphic user interface follow. Section 4 explains an experiment conducted to evaluate the effectiveness of the tutor, and Section 5 shows its results. We then discuss general lessons learned through this study in Section 6. Finally, in Section 7, we discuss implications for the design of a tutoring system for problem solving and other issues for future works.

2. Issues in Learning and Teaching Geometry Theorem Proving

This section first introduces a target task, geometry theorem proving with construction. It then discusses the students' difficulties in learning this task. We then summarize pros and cons in teaching and learning forward / backward chaining.

2.1. Task: Geometry Theorem Proving with a Two-column Proof

The target domain is elementary Euclidean geometry. In this study, we deal only with proofs of equality and congruence that do not involve arithmetic operations (i.e., sums and

multiplications).¹ This restriction is required by GRAMY, an automated geometry theorem prover that is capable of construction (Matsuda & VanLehn, 2004), which was built as a part of the Advanced Geometry Tutor project (see Section 2.2 for details of GRAMY). The problems (i.e., geometry theorems) used in this study may require construction *as a part of the proof*. Although GRAMY is capable of finding all kinds of construction that can be done with a compass and a straightedge, AGT only deals with constructions that can be done by connecting two existing points.

A problem consists of (1) a set of given propositions, (2) a proposition to be proved, and (3) a diagram called the problem figure that represents generic configuration of the problem. We call a problem figure that is originally given to a problem the *initial problem figure* to discriminate it from a problem figure after some constructions were took place. In this study, the term “postulate” refers to statements that are known to be true such as definitions, axioms, and proved theorems. A postulate consists of premises and a consequence that are represented as propositions. Each postulate is associated with a generic diagram that represents topological information that may or may not be represented in the premises and the consequence. This generic diagram is called the *configuration* of the postulate.

The students are taught cognitive skills for composing two-column proofs as shown in Figure 2.1. A two-column proof is represented as a table where a row corresponds to a *proof statement*, which consists of a *proposition* and a *justification*. A proposition is a geometric assertion that appears in the left column in the proof table. Every proposition must be justified by providing a valid justification on the right column in the table. A proposition is justified in

¹ This restriction implies that the proofs of inequalities, ratios, and coincident intersection (i.e., to prove that three or more segments intersect at one point) are also excluded from the present study.

one of two ways. (1) Providing an associated keyword in the justification cell for an *obvious proposition*. Obvious propositions are true propositions whose truth value do not depend other propositions. For example, “Given” is the associated keyword for given propositions, “VerAng” for vertical angles, “Identical” for congruence of identical elements (say, $\angle ABC = \angle CBA$), and so on. (2) Specifying the name of the postulate that logically derives the proposition. For the latter case, the *premises* of the postulate must be also mentioned by listing their line numbers.

	Proposition	Justification
1.	$\angle ADC = \angle BDC$	GIVEN
2.	$AD = BD$	GIVEN
3.	$CD = CD$	IDENTICAL
4.	$\triangle DCA \cong \triangle DCB$	SAS (3, 2, 1)
5.	$AC = BC$	CPCTC (4)
6.		
7.		

Figure 2.1: An example of a proof table

The proof shown in Figure 2.1 was written by forward chaining: the proof table starts with givens at the top and ends with the to-be-proved goal ($AC=BC$) at the bottom. When composing a proof forwards, the students are provided with a proof table that only contains givens at the first few rows (the first 2 rows in the case of Figure 2.1). Those rows have “Given” as their justifications. The students are then supposed to extend the proof table by asserting a new proposition into an empty proposition cell. They then need to provide a justification for the new proposition by filling in an empty justification cell. The students continue this process until a top level goal is eventually asserted into the proof table.

When composing a proof backwards, the students are provided with a proof table that only contains the to-be-proved goal at the first row. The goal has an empty justification. Students are then supposed to fill in empty justifications. To use a postulate as a justification, they must

assert the postulate's name into the empty justification cell next to the proposition being justified. The student then assert premises of the justification into the empty proposition cells one at a row starting immediately beneath the row that contains the proposition being justified. Finally, the student writes the line numbers of the premises into the justification cell of the being-justified proposition. They continue this process until all the empty cells are filled in.

In some case, a universal proof must be conditional, thus requiring different problem figures that are consistent with the given propositions. However, in most cases, classroom instruction only requires students to find a proof for a particular problem figure and does not ask them to generate conditional proofs, so that is all GRAMY does. That is, GRAMY finds proofs and constructions that hold for the give problem figure. So does AGT; the tutor only finds a proof, often with a construction, for the initial problem figure and ask students to write that particular proof.

2.2. Theorem Proving with Construction

This section describes a procedure for theorem proving with construction that has been developed through working with GRAMY. The procedure is believed to be comprehensible for students and is embedded in the cognitive model utilized by AGT.

2.2.1. Theorem proving with construction as a state-space search

Geometry theorem proving with construction can be viewed as a state-state search. The initial state holds a set of propositions assumed to be true (givens), a proposition to be proved (the goal), and the initial problem figure. To change a state, one can either apply a postulate forwards to assert a new proposition or backwards to make new subgoals, or apply construction operators to add new segments and points to the problem figure. The goal state holds a proof that is a sequence of postulate applications and constructions. In sum, theorem proving with

construction can be formalized as a state-space search where a postulate application (with or without construction) is the basic mean of state transition.

2.2.2. Finding a useful postulate for construction

If students are to apply a postulate, they must be able to overlap the configuration of the postulate onto the problem figure. This overlapping must be done so that the premises and consequence of the postulate are *quantitatively satisfied*, which by definition means that the relations stated in the proposition are consistent with the measurements of the corresponding geometric elements in the problem figure. For example, a proposition $XY = WZ$ in the postulate is quantitatively satisfied if two segments, say, AB and CD , in the problem figure on which XY and WZ are overlapped respectively are approximately the same length.

Since a proof is a sequence of postulate applications, if a proof exists for a problem that requires a construction, there exists at least one postulate application in the proof that does not perfectly overlap with the initial problem figure. The key idea behind our construction technique is to find such a postulate.

Let's call a postulate *useful* if its consequence unifies with the goal to prove and all premises that match the problem figure are quantitatively satisfied but some premises might not match the problem figure. For example, when a goal is to justify an angle congruence $\angle ABC = \angle DEF$, the postulate CPCTC (Corresponding Parts of Congruent Triangles are Congruent), which says "if $\Delta xyz \equiv \Delta uvw$ then $\angle xyz = \angle uvw$," is useful as long as the problem figure points bound to x , y , z , u , v , and w form quantitatively congruent triangles. The appropriate triangles may not completely exist in the problem figure. For instance, the student may have to draw the segment connecting x to y . So, the question of how to make appropriate construction is reduced to the question of how to find a useful postulate. Finding a useful postulate is indeed rather

straightforward. One can first pick a postulate that has a consequence that matches the goal to be proved and then test if the postulate partially overlaps the problem figure.²

2.2.3. Identifying target segments by partial overlapping

Once a useful postulate is found and a partial overlap is identified, the rest of the procedure is fairly straightforward. Since we only deal with construction for connecting existing points, the partial overlapping must be the one that has all the points in the postulate bound to some point in the problem figure. For each of the missing segments, one can then simply connect their endpoints.

2.3. Students' Difficulties in Learning Proof Writing

Geometry theorem proving is a challenging subject for students. In a large scale classroom evaluation with 1520 students, Senk (1985) showed that only 20% of the students could do complex proofs at the end of a year-long geometry class, that 30% could find proofs only for problems that were similar to the ones in textbook, and that 25% could only do trivial proofs.

Studies have suggested many difficulties that the students suffer when learning proof writing. They include lacking a commonsense meaning of proofs as mathematical argumentation (Dreyfus, 1999), a difficulty in communicating in *mathematical language* (Laborde, 1990; Landa, 1975), in transforming descriptive (conceptual) knowledge into operational (procedural) knowledge (Dreyfus & Hadas, 1987; Greeno, 1983; Tubridy, 1992), and in applying abstract and formal reasoning (Algarabel & Dasi, 1996; Lovell, 1971; Renner &

² In general, there may be multiple postulates that are useful for a particular goal. One must learn a selection schema (i.e., a search control). AGT, however, does not teach search control explicitly. Students just follow the shortest proof that the tutor provided. See Sections 3.2 for details on how AGT helps students solve problems.

Stafford, 1976), search skills (Schoenfeld, 1985); and misconceptions (Chaiyasang, 1989; Schoenfeld, 1988).

Especially remarkable issue is that the students fail to write proofs even when they have conceptual mastery of geometric propositions and postulates. Koedinger (1990) reported that the students had only 35.5% accuracy on proof writing at a pre-test even though they showed 67% accuracy on the test items for judgment of geometric statements. Chaiyasang (1989) showed that less than 15% of the students could achieve “good” in proof writing even though they are ranked as the van Hiele level 4, which means that they understood geometric concepts necessary to write a proof.

These studies show that teaching geometric concepts is not enough to have students master proof writing. Students apparently need deliberate practice in writing proofs. The question is how to make such practice effective and efficient. The next section provides theoretical insight into a desired learning environment for geometry theorem proving by comparing two strategies: forward and backward chaining.

2.4. Teaching and Learning Forward / Backward Chaining

As mentioned earlier, the primary interest in the current study is to investigate the difference between teaching *forward chaining* (FC) and *backward chaining* (BC) as a strategy for theorem proving. This section provides literature review on the studies that address this issue by highlighting two different aspects of teaching and learning those strategies: cognitive theories and computational theories.

First of all, majority of cognitive studies on problem-solving performance of novices showed that they tend to prefer FC to BC. For example, Koedinger mentioned about classroom experience with a geometry proof tutor that provided hints based on BC where “[t]he average

student found this very confusing” hence “such hints were eliminated so that the current version of the tutor [called ANGLE] only tutors forward chaining” (Koedinger, 1991). Anderson *et al.* (1993) also reported that “all but gifted students had great difficulty with backward reasoning facility (p.172).” In a study of a LISP tutor, GIL, Trafton and Reiser (1991) observed that on the post test, 95% of the steps taken by the students who had been trained for both FC and BC (so called “Free” tutor condition) were FC. Also, in a study comparing performance of novices and experts solving Physics problems, Priest and Lindsay reported that they observed “[all the participants] show[ed] the same overwhelming bias towards the employment of forward inference” (1992, p.401). They concluded that a potential theoretical account for the novice-to-expert shift in the performance is not due to shift in problem-solving strategy (e.g., from BC to FC), but a shift from so called unguided FC to schematic FC. These studies support teaching FC rather than BC for students learning problem solving skills.

On the other hand, there are some conflicting studies that claim that the novices rely on backward chaining whereas the experts prefer to forward chaining. When observing novices solving Kinematics problems, for example, Larkin *et al* found that the novices tend to apply backward chaining (Larkin, McDermott, Simon, & Simon, 1980). “The management of goals and subgoals [is] deciding periodically what to do next” (Larkin et al., 1980, p.1338). Unlike the experts, the novices do not apply schematic knowledge (M. T. H. Chi, Feltovich, & Glaser, 1981), hence tend to rely on analytic goal-directed strategy. This strategy apparently affects the experts’ performance as well so that the “experts work forwards only on easy problems” (Larkin et al., 1980, p.1338). There is also a computational model of novice-expert shift from backward to forward chaining called EUREKA (Elio & Scharf, 1990). The EUREKA model predicts that the strategy change occurs when the content of the problem solving principles (or schemas, if

you will) changed so that discriminating features of problem description reflect more fundamental physics principles. These studies support that it may be natural for the students to use backward chaining to learn geometry theorem proving.

Above studies do not directly compare difference in teaching FC and BC. Not so many studies have been conducted that directly address the difference between teaching FC and BC. However, when those strategies were compared, the results were rather neutral. In the GIL study, Trafton and Reiser (1991) compared strict FC, strict BC, and bidirectional conditions. In the strict FC/BC conditions, the students could only use FC or BC. In the bidirectional condition, the students could use both strategies freely. After solving 14 training problems, all conditions tied on the post-test, although the BC students made more errors and require more time to write a program during the training session than other conditions. Scheines and Sieg (1994) also compared learning gains for students learning logic proof by either strict FC, strict BC, and bidirectional conditions. In this study, the students used computerized tutor (Carnegie Proof Tutor, or CPT) for 5 weeks. Quite similar to the GIL study, there was no significant difference between FC and BC conditions, but the bidirectional chaining condition outperformed the other conditions only on the hard problems.

Although the overall performance on the post-test did not differ over the tutor (i.e., the strategy) conditions, there may be a difference in certain sub skills. Through the GIL project, Reiser *et al.* (1994) replicated the result of not having main effect in the tutor condition (i.e., BC vs. FC vs. bidirectional), but also found that the students in the bidirectional condition scored significantly higher on debugging (but not repairing) tasks than BC and FC students.

If the theories found in above studies could apply to the current study, then we might also hit a null effect on the tutor conditions. However, there is a reason that learning with FC and BC

would differ from each other especially for geometry theorem proving with construction, which is inspired by a computational model of geometry theorem proving. A study with GRAMY (Matsuda & VanLehn, 2004) revealed that forward chaining is much more efficient than backward chaining for geometry theorem proving that *do not* require construction. Table 2.1 shows a comparison of search complexity between forward and backward chaining for proof problems without construction performed by GRAMY. GRAMY applies forward chaining to calculate a deductive closure, which is called *exhaustive forward chaining*. As shown in the table, exhaustive forward chaining is superior to backward chaining for all non-construction problems with different complexities (i.e., the length of the shortest proof). On the other hand, due to high branching factors, backward chaining blew up for the hard problems. Especially, backward chaining could not find a proof for P011 and P005 hence was terminated manually. These findings suggest that one must be taught forward chaining when learning geometry theorem proving when it does not involve constructions.

Table 2.1: Comparison of search complexity

Problem	Proof Length	Forward Chaining				Backward Chaining				
		Depth	Time	Space		Depth	Time	Space		
				#Prop.	#State			#Prop.	#State	ABF
P001	3	2	1.54	22	6	2	1.54	2	30	5.03
P008	4	2	4.40	34	7	3	4.56	3	57	6.16
P010	6	4	9.40	57	15	5	8.84	2	7104	6.84
P006	7	2	29.55	146	6	6	31.03	4	360	2.26
P004	10	5	28.23	111	42	9	106.56	5	1267388	3.89
P011	40	12	41.19	198	2259	14+	-	6	40378455+	-
P005	55	10	37.02	199	498	9+	-	7	53324115+	-

ABF: Average Branching Factor

What about theorem proving *with* construction? Forward chaining, even exhaustive forward chaining, suffers from search explosion when finding proofs that involve construction. For example, for a simple geometric theorem regarding a simple quadrangle (i.e., a theorem

involves only four segments), there are about 78 different “meaningful” constructions³ possible every time one draws a segment to the problem figure with a compass and a straightedge. On the other hand, when constructions are supported by backward chaining, the number of constructions drastically decreases. Table 2.2 shows a search complexity to find proofs with construction supported by backward chaining. Although ratio of successful construction to unsuccessful ones are still quite low, the table implies that backward chaining must be taught for geometry theorem proving with construction. The essential difference is that BC only draws a line (or other construction) that will make a useful postulate match, whereas FC’s construction is unconstrained. Any possible construction may be drawn at any FC step. Therefore, here, again, is a conflict between FC and BC; FC is quite efficient for proofs without construction, but BC must be taught for proofs with construction.

Table 2.2: Search complexity with construction supported by backward chaining

Problem	First Proof				All Proofs				
	Length	State	Prop.	Time	All Const	Suc. Const	State	Prop.	Time
P132	3	4	12	11	40	40	40	1342	76
P127	3	296	70	1529	101	7	189	12252	2381
P123	4	15	38	35	5	1	9	200	35
P109	5	198	91	322	76	3	149	2414	512
P101	5	313	72	1764	109	8	205	13075	2573
P116	6	130	79	109	76	1	151	3303	392
P131	6	492	63	2156	122	16	228	12880	2027
P111	6	81	146	544	135	14	256	18229	3967
P115	8	26	65	61	55	1	109	2785	329
P112	8	23	79	151	181	14	348	21584	3908
P117	9	48	178	54	17	1	33	1642	205
P129	10	13	349	278	36	1	71	15095	5268
P142	10	13	127	84	95	7	183	16251	3356
P128	11	93	103	770	146	2	290	20255	2951
P108	14	112	92	185	49	1	97	2137	432
P144	19	85	152	146	61	6	112	2691	472

³ The meaningful constructions involve drawing a parallel line, drawing a perpendicular line, drawing a median line, drawing an extension line, etc.

In sum, students might learn geometry theorem proving better with FC, but it could only be an efficient strategy for proofs *without* construction. BC might be much more efficient for theorem proving with construction, but it could be too challenging for students to learn. Indeed, as discussed in Section 3.3, backward chaining is more complicated than forward chaining in terms of the number of inference steps to be performed. Backward chaining also involves more tacit inference steps than forward chaining. Therefore, it is not surprising that the students showed difficulties in learning backward chaining hence resulted in poor performance on post-test. Yet, we are lacking theoretical support to determining which one of these strategies facilitates students learning the target task – geometry theorem proving with construction.

3. The Advanced Geometry Tutor

This chapter describes the architecture of AGT. First we discuss theoretical issues in designing a tutor. A brief survey on cognitive theories of teaching and learning problem solving is given to provide insight into an effective and efficient tutor. We then describe details of AGT.

3.1. Theoretical Implications in Teaching Geometry Theorem Proving

This section reviews studies on teaching and learning problem solving, especially those that have theoretical implications in designing our intelligent tutor. More specifically, we focus on the following factors that might have a significant impact on students' learning: (a) learning from worked-out examples and problem solving, (b) articulating tacit inference steps, and (c) teaching operationalization. The following sections explain these instruments and why we think they will be effective.

3.1.1. Learning from worked-out examples and problem solving

Most tutoring systems ask students to solve problems. These tutoring systems assume that the students have learned the knowledge necessary to solve problems and yet they need to stabilize their knowledge through practice. However, as VanLehn (1998) mentioned, when the students reach an impasse, they often go back to the examples and do analogical reasoning from them. The advantage of learning with worked-out examples has been observed in many studies (M. T. Chi, Bassok, Lewis, Reimann, & Glaser, 1989; Sweller & Cooper, 1985; Zhu & Simon, 1987). Nonetheless, most computer tutors do not allow students this kind of retrograde reference to the examples.

The worked-out examples can be written in a textbook or generated by the tutor on the fly. Namely, based on the underlying cognitive model of proof writing, the tutor can perform all the inference steps, no matter if they are observable or not, and explicitly show them to the students. This type of tutor's aid for novice students is called modeling in a context of cognitive apprenticeship learning, which involves modeling, coaching, scaffolding, and fading (Collins, Brown, & Newman, 1989).

At the beginning of the tutoring session, when the students are not familiar with solving problems, the effect of modeling would be maximized because it provides students with opportunities to learn domain principles. As learning proceeds, however, the benefit of modeling may decrease (Kalyuga, Chandler, Tuovinen, & Sweller, 2001). Hence the tutor must gradually elicit more steps from the students. This switching from worked-out example to problem solving, which is often called *fading* (Collins et al., 1989), has been investigated in several of studies (Renkl, Atkinson, & Grose, 2004; Renkl, Atkinson, Maier, & Staley, 2002).

These studies emphasize the importance of fading, but what if the students eventually get stuck after the worked-out examples are faded away? The tutor must resume providing modeling

again for such situation. That is, problem solving must be replaced with worked-out example once again when students show poor performance. In sum, the studies mentioned above suggest that modeling is useful for students who are just beginning to learn proof writing. The amount of scaffolding must be controlled based on the student's competence level.

3.1.2. Articulating tacit knowledge

As discussed in the previous section, the benefits of worked-out examples can be seen in many studies. However, as mentioned in VanLehn et al. (1992), many inferences taken in solving a problem are not displayed in worked-out examples. Asking students to self-explain worked-out examples increases the learning in part because they are filling the missing information (M. T. Chi et al., 1989; Renkl, Stark, Gruber, & Mandl, 1998; VanLehn et al., 1992).

Let us define an *inference step* as a primitive component of the problem-solving procedure. An inference step can be either mental, which is not observable, or physical, which is observable. The objective of tutoring in this study is to make students acquire all those inference steps necessary to write proofs. A cognitive task analysis of geometry theorem proving, presented in Section 3.3, demonstrates that only about 30% of the inference steps correspond to observable steps, that is, some kind of operation with a GUI component (i.e., to press a button, to enter an equation, etc). The remaining 70% are unobservable, and might be a major source of the failure of learning for low competent students, who can not uncover such tacit knowledge even when they try to self-explain the example's steps.

Once a cognitive model has been created that involves detailed inference steps then we can utilize a well-known effective tutoring strategy, *model tracing*, which assesses students' competence on each inference step while they solve problems and provides appropriate feedback so that students can learn correct problem solving skills (Anderson, Boyle, Corbett, & Lewis,

1990; Koedinger, 1991; Koedinger & Anderson, 1993). The Geometry Proof Tutor is one of the most successful tutoring systems for geometry theorem proving (Anderson et al., 1990). The model tracing tutor represents a model of theorem proving as a set of production rules. Each production rule is a unit of instruction for the model tracing tutors. The tutor monitors whether students can apply a particular production rule in a particular situation, and if they fail, the tutor gives instruction on how to apply the production rule. An empirical evaluation showed that when students use the Geometry Proof Tutor individually, their performance was more than one standard deviation higher than that of the traditional classroom instruction (Anderson, Corbett, Koedinger, & Pelletier, 1995).

Although one can build a cognitive model of problem solving at an extremely fine grain size including all perceptual and motor skills, we have yet to know what exactly is the right grain size for such a cognitive model. Because there are several studies that support the importance of teaching geometric postulates in a conditional form, which clearly emphasizes premises and consequence (Dreyfus & Hadas, 1987; Greeno, 1983; Tubridy, 1992), we have included an inference step that articulates premises and the consequence of a postulate being applied. The next section discusses this issue.

3.1.3. Teaching operationalization

One of the unobservable inference steps that have been occasionally reported to be particularly important for proof writing is to transform geometry statements written in a declarative form into a conditional form. For example, a theorem taught in a declarative form “two base-angles of an isosceles triangle are equal” must be translated into a conditional form “if a goal is to prove $\angle ABC = \angle ACB$ in a triangle ABC , then set the goal to prove that $AB = AC$.” We hereafter call such transformations *operationalization*.

Several successful methods teach geometric postulates as conditional statements, which have conditions as an IF-part and consequences as a THEN-part. Tubridy (1992) taught students conditional statements in conjunction with the so-called three-part-format, which consists of (1) a configuration of the postulate, (2) a consequence of the postulate, and (3) a verbal explanation of the postulate. The evaluation showed that the Tubridy's instructional strategy led low- and middle-level students to better performance on proof writing. Greeno (1983) emphasized IF-THEN structure of conditional statements when he taught students proof-checking where the students judge the correctness of written proofs. After four 1-hour training sessions, the students showed better performance on proof writing. Dreyfus and Hadas (1987) articulated six principles for teaching geometry theorem proving: (1) a theorem has no exceptions, (2) even "obvious" statements have to be proved, (3) a proof must be general, (4) *the assumption of a theorem must be clearly identified and distinguished from the conclusion*, (5) the converse of a correct statement is not necessarily correct, and (6) complex figures consist of basic components whose identification may be indispensable in a proof. The fourth principle emphasizes the premises and consequence of postulates. Two hours of instruction per week for one full school year produced a significant effect on proof writing for the mid-year and post-year tests.

These studies suggest that students need to learn operationalization explicitly as a part of the cognitive skills for proof writing. Hence we assume that the underlying cognitive model of proof writing should include operationalization as an inference steps.

3.1.4. Summary

We have reviewed empirical studies that provide evidence for a learning environment to be effective and efficient. The survey in the previous section suggests that an ideal learning environment for our target students (those who are at an intermediate level) should provide

modeling that fades away as the student get familiar with proof writing, but also fades in when necessary. Modeling must articulate problem-solving steps in great details including inference steps that are unobservable as well as those that transform geometric postulates in declarative form into conditional form, namely operationalization.

3.2. Overview of the Advanced Geometry Tutor

AGT teaches how to compose a proof. More precisely, it teaches how to complete a proof table, which requires asserting propositions, justifications (i.e., postulate names), and premises. There are two versions of AGT: the *forward chaining tutor* (FC tutor for short) teaches only a forward inference procedure. On the other hand, the *backward chaining tutor* (BC tutor) teaches only a backward inference procedure. In either direction, a postulate application may or may not involve construction.

In order to write a proof statement, the student must make several inference steps, such as selecting a postulate, matching its configuration to the problem figure, testing its conditions, etc. AGT has a cognitive model that represents which inference steps are required for which kind of proof statement. The tutor requires the students to follow the exact sequence of inference steps in the cognitive model every time the student asserts a proof statement into the proof table. When a student made an inappropriate (or erroneous) inference step, the tutor immediately provides feedback. The content of immediate feedback serves as a hint. When a student makes multiple unsuccessful attempts of the same inference step, the tutor provides more and more specific feedback. After the student made a certain number of false trials on a particular inference step, the tutor gives the so-called *bottom-out hint* (Koedinger & Anderson, 1993), which both performs the inference step for the student and provides a specific instruction on what to do.

One of the most prominent features of AGT is that it starts the tutoring session by showing how to compose a proof table. That is, the tutor first provides the students with the worked-out examples of proof writing. The tutor then gradually decreases the amount of modeling (i.e., fading takes place) and starts asking students to perform the inference steps by themselves. The degree of modeling is determined based on the student's performance on each of the inference steps.

3.3. A Cognitive Task Analysis of Geometry Theorem Proving

This section presents a cognitive task analysis of backward and forward reasoning for proof writing. The unit of analysis is a *postulate application*, namely, the assertion of a proof statement. The study with GRAMY revealed that for the most of the problems used in the textbooks, construction can be implemented as a substep of a postulate application (i.e., a construction procedure can be described as a part of the inference steps that assert a proof statement into the proof table). Hence our model of proof writing delineates the cognitive skills of applying postulates with and without construction.

The cognitive model of applying a single postulate is comprised of a hierarchy of *goal-subgoal* relations whose top level goal is making a backward or forward postulate application (see Figure 3.1). Each goal in the model corresponds to a single inference step involved in a postulate application. The leaves of the hierarchical model represent *operations* that the students must perform. Some of them are the observable manipulations upon a proof table (e.g., to enter a propositions), whereas others are unobservable mental steps (e.g., to see if a proposition about to be asserted is already in the proof or not).

For the sake of tutoring, separate goal hierarchies were developed for postulate application with and without construction and with either forward or backward chaining (e.g., backward-

inference and backward-inference-with-construction). Also, a proposition with “obvious” justifications such as “Given” or “Identical,” is modeled with a unique top-level goal (e.g., forward-obvious). Another example of a proposition with an obvious justification is asserting two angles that form vertical angles, which in AGT, can be justified by simply stating “VerAng.”

As an illustration of one of the goal hierarchies, Figure 3.1 shows a cognitive model of backward chaining with construction. The italicized steps are observable ones. Backward chaining consists of two major goals: “Select a proposition to justify” and “Apply a postulate backwards.” The former corresponds to selecting an unjustified proposition in the proof table. The latter step has three sub-steps: “Select a postulate,” “Construction,” and “Execute the postulate.” These three sub-steps are further broke down as follows.

When selecting a postulate, students must check that the selected postulate overlaps with the problem figure and that it is indeed *effective*, namely, its consequence matches the proposition to be justified.

If needed, construction takes place immediately after selecting a postulate. As described in Section 2.2, construction is done to complete a partial overlapping between the postulate configuration and the problem figure. Hence the construction consists of two substeps: (1) Finding missing segments and (2) constructing the missing segments.

Finally, to execute the postulate, students first identify premises to be asserted (Instantiate premises), verify if they already appear in the proof table (Check Duplication), and assert only those premises that are not in the proof table as goals, namely, propositions that need to be justified.

Backward inference construction

- Select a proposition to justify*
- Apply a postulate backwards
 - Select a postulate
 - Pick a postulate*
 - Overlap configurations
 - Transform the postulate into a conditional form
- Construction
 - Find missing segments
 - Construct missing segments*
- Execute the postulate
 - Instantiate premises
 - Check Duplication
 - Assert premises as unjustified propositions*

Figure 3.1: Cognitive model of backward inference with construction

As can be seen in Figure 3.1, to make a single backward chaining postulate application with construction, students do nine operations where four operations are observable and five are unobservable. APPENDIX L lists all other cognitive models used in AGT.

3.4. The Solution Graph

Prior to a tutoring session, the tutor invokes GRAMY to find the shortest proof for the target problem. It then builds a *solution graph* from the output of GRAMY. This section describes how solution graphs are built from proofs.

The solution graph represents a sequence of *proof steps*, each of which corresponds to asserting a proof statement into the proof table. For the proof steps that do not involve construction, a step consists of (1) a proposition to be asserted, (2) the name of the postulate that justifies the proof step, and (3) the premises that support the postulate application. For proof

steps that require construction, a step also contains information about the construction, namely the segments to be constructed.

The backward chaining tutor builds a solution graph by traversing the proof depth first from the goal. A proof step for construction is asserted into the solution graph as a node immediate before an application of the postulate that requires construction.

The forward chaining tutor builds a solution graph by traversing the proof bottom-up. Namely, starting from given propositions in the proof, it makes a proof step that contains all propositions in the proof that are *immediate consequences* of a set of propositions in a solution graph built so far. A proposition p is an immediate consequence of a set of propositions \mathbf{P} , if p can be derived by single postulate application with a subset of \mathbf{P} . A proof step for construction is asserted into the solution graph when no proposition can be an immediate consequence without construction.

3.5. The Scaffolding Strategy

The tutor simply follows the solution graph in order to provide scaffolding for students to complete a proof table. Scaffolding only focuses on helping students follow inference steps necessary to perform proof steps. In other words, scaffolding provided by AGT is carefully designed to provide local feedback and hints on each postulate application, not for a global search strategy.

As mentioned in Section 3.3, there are different types of proof statements: with and without construction, and the obvious proof steps. Those proof statements are embedded into the solution graph as proof steps hence there are different types of proof steps in the solution graph. Each type of proof step in the solution graph is associated with a particular scaffolding dialogue. Thus the scaffolding dialogue is hierarchical and its entire structure is identical to the goal

hierarchy for the corresponding proof step. Associated with each inference step in the goal hierarchy is a *dialogue script* that defines tutor's reaction to the student's input.

To control the amount of scaffolding on an inference step, the behavior of the step's dialogue script depends on the student's competence level for the step. There are three levels of scaffolding:

Show-tell: the tutor tells students what to do and actually performs the step.

Tell: the tutor tells students what to do, but asks the student to perform the step.

Prompt: the tutor only prompts the student to perform the step.

The student's competence level for a step is maintained as follows. When the student correctly performs an inference step, the tutor increases the competence level. Conversely, when the student commits an error on an inference step, then the competence level of that step is decreased.

When the scaffolding level is "Tell" or "Prompt," the tutor asks the student to perform the step, and if the student's response is wrong, the tutor immediately provides feedback and asks the student to enter a correct input. At first, the tutor says that the student has made an error and provides minimal feedback (e.g., "Try again"). If the student repeatedly fails to perform the inference step correctly, the tutor provides more specific feedback until it eventually reaches bottom-out hint, which is equivalent to the show-tell scaffolding.

There is no way for students to seek help even when they get stuck. The students must do some kind of action to receive the tutor's feedback. In other words, an incorrect response at the impasse triggers a tutor's help that varies according to the student's competence level.

For example, for an inference step for construction the tutor would say "Draw segments so that the postulate has a perfect match with the problem figure." When the student still fails to

draw correct segments, the tutor lowers the competence level of that inference step and then provides a “Tell” dialogue, which generates a feedback message like “Draw new segments by connecting two points.” If the students yet can not make a correct construction, then the tutor provides more specific “Show-Tell” dialogue that would say “connect points A and B.” Note that this sequence roughly corresponds to a sequence of hints that starting from a general idea and becoming more concrete until very specific instruction (bottom-out hint).

APPENDIX A and APPENDIX B show an example of scaffolding dialogue provided by the forward and backward AGT respectively.

In sum, cognitive skills of proof writing are modeled as hierarchical inference steps for each type of postulate application, which corresponds to asserting a single proof statement into the proof table. Associated with the individual inference steps are three competence levels of dialogue scripts that are used both to initiate the tutor’s message and to provide feedback for a student’s input. The tutor dynamically changes the competence level that is also associated with each of the inference steps. Changing competence levels controls the amount of scaffolding, which in turn realizes fading as well as generating a hint sequence.

3.6. The AGT Learning Environment

AGT consists of several GUI components. Figure 3.2 shows a screen shot of the tutor. On the left side from top to bottom, the tutor provides the Problem Description window and the Proof Table window. On the right hand side, there are the Message window, the Postulate Browser window, and the Inference Steps window. The next section provides brief descriptions for each window, and Section 3.6.2 explains how learning proceeds in this learning environment.

Most of the AGT components are written in Common-LISP running on a PC. The graphic user interface (GUI) was written as a Java applet that can run on a variety of web browsers. The

LISP modules and GUI components had a socket communication channel to exchange various messages.

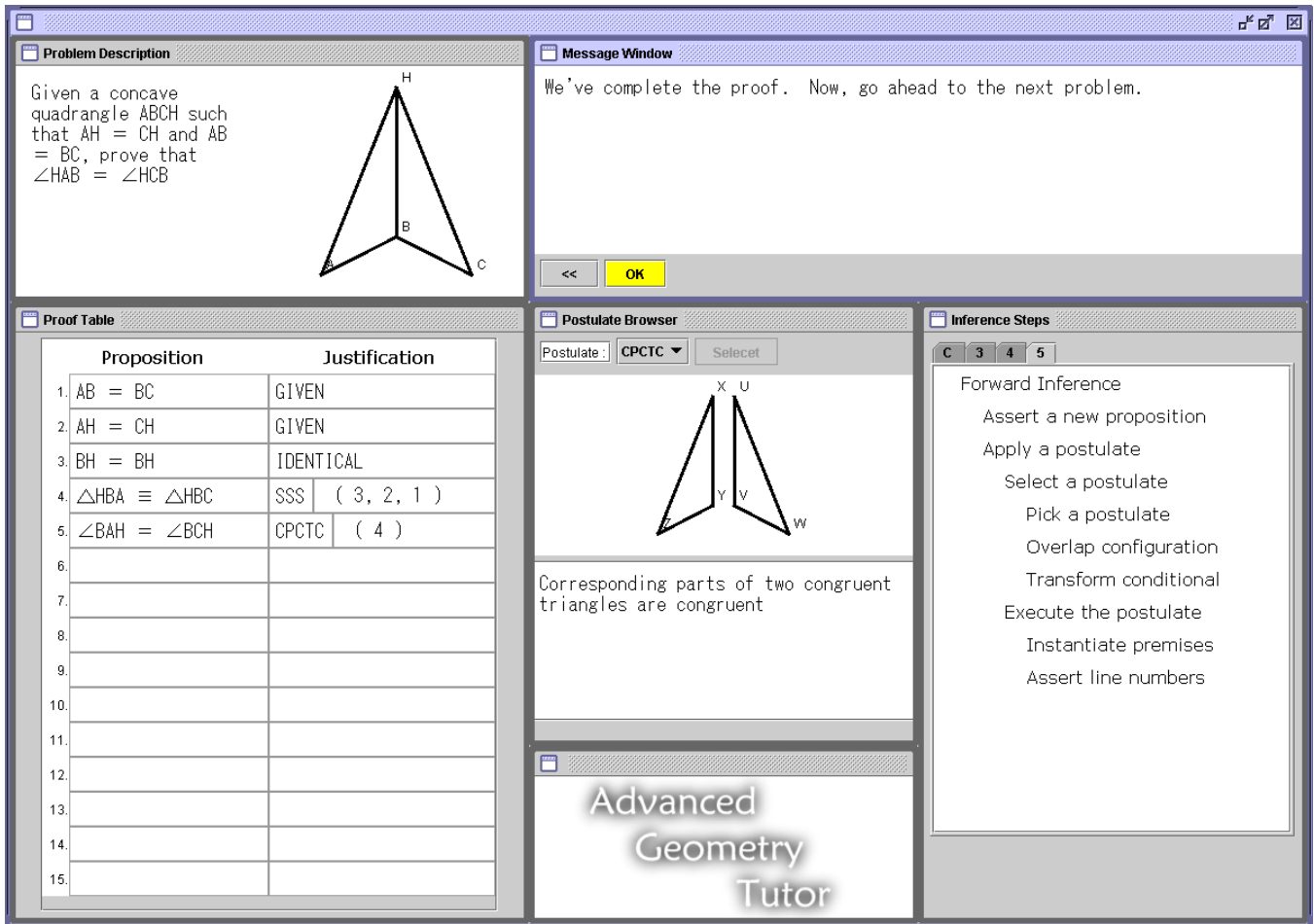


Figure 3.2: The Advanced Geometry Tutor

3.6.1. GUI components

Problem Description window: This window shows a problem statement and a problem figure. The problem figure displayed in this window is used for construction. That is, the student can draw lines on the problem figure when it is time to do so.

Proof window: A proof is realized as a two-column table where each row consists of a proposition and its justification. A justification consists of a name of a postulate and, for the proof statements with a non-obvious justification, a list of line numbers for the propositions that

match the premises of the postulate. The Proof window shown in Figure 3.2 shows a complete proof for the problem displayed in the Problem Description window.

Message window: All kinds of messages from the tutor appear in this window. When the tutor provides modeling, the instructions that the student must follow appear here. When a student makes an error, feedback from the tutor appears here. More importantly, this window is used for the students' turn in the dialogue, which sometimes consists of merely clicking the [OK] button. Dialogues are stored, and the student is free to browse back and forth by clicking a backward [\ll] and a forward [\gg] button.

This window is also used unobservable inference steps, which by definition do not have actions onto the proof window. An example of such unobservable response is for "Check Duplication" step shown in Figure 3.2. For this step, the student must answer if the premises for the postulate application are already in the proof table or not. The tutor may ask the student "Is $AB=CD$ already in the proof table?" and awaits students response. At that time, additional buttons appear in the Message window allowing the student select a [Yes] or [No] response.

Postulate Browser window: The student can browse the postulates that are available for use in a proof. When the student selects a postulate listed in the browser's pull down menu, the configuration of the postulate, its premises, and its consequence are displayed.

This window is also used by the tutor. As shown in Figure 3.2, when the tutor provides Show-tell or Tell level scaffolding on how to apply a particular postulate to a particular proposition, the configuration of the postulate changes its shape so that the student can see how the postulate's configuration should be overlapped with the problem figure.

Inference Step window: The Inference Step window reifies the relevant goal hierarchy of postulate application as indented texts where each line corresponds to a single inference step.

The tutor highlight the inference step that is about to perform. The Inference Step window in Figure 3.2 shows inference steps for forward chaining without construction.

3.6.2. Students' activities and tutor's behavior

The tutor first shows a problem in the Problem Description window. The tutor then starts to guide the student's problem solving by displaying messages in the message window. Depending on the student's competence on an individual inference step, the tutor provides messages at one of the three levels of scaffolding described in Section 3.5. The student is supposed to read these messages and press [Ok] button to proceed the tutoring session.

Also as described in Section 3.5, since there is no facility to seek a hint, the students must enter something wrong even if they do not know what to enter. The tutor then starts a hint sequence.

When the student needs to input an equation, (e.g., $\angle ABC = \angle DEF$), an inline equation builder appears at the place where the equation must be asserted (Figure 3.3). The student can select a template of the equation (e.g., $\angle \square = \angle \square$ for an angle congruence) then just enter point labels to complete the equation.

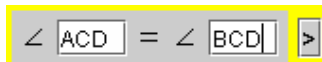


Figure 3.3: The equation builder

Since AGT only deals with construction that can be done with connecting two existing points, the student only needs to specify two points in the problem figure to make a new segment.

4. Evaluation of AGT

An evaluation study was conducted in the spring of 2004 to test the effectiveness of AGT and to compare the FC tutor to the BC tutor. This chapter describes an overview of the evaluation study followed by its results in Chapter 5 and general discussion in Chapter 6.

4.1. Subjects and Design

We recruited 52 students for monetary compensation from the University of Pittsburgh. There were 24 male and 28 female students at the average age of 23.3 (SD=5.4). The students were randomly assigned to conditions. The sessions were run individually.

4.2. Procedure

After completing a consent form for the study, students were asked to read a 9-page booklet describing basic concepts and skills of geometry theorem proving. Then they took a pre-test for 40 minutes, which was open-book. Immediately after the pre-test, each student used AGT and solved 11 problems. The tutor sessions were split in two or three days based on the students' preference. On the last day of the tutoring sessions, immediately after solving the last problem, the students were asked to take an open-book post-test for 40 minutes. For all students, the entire study sessions completed within 7 days.

4.3. Materials

Since the BC tutor and the FC tutor taught different strategies, the materials used in the study differed across conditions. The difference of materials was localized to the difference in the strategies, for example, the materials for the forward tutor condition only showed inference steps relevant to forward inference.

4.3.1. The booklet

The booklet contained (1) a review of geometry proofs that explains the structure of geometry proofs and the way they are written, (2) a technique for making a construction, and (3) explanations of all postulates used in the study. For each postulate, the explanation consists of a general (English) description of the postulate, the configuration of the postulate, a list of premises, and the consequence of the postulate. APPENDIX C and APPENDIX D show the geometry booklet for the BC tutor and the FC tutor respectively.

4.3.2. Pre- and post-test

The FC and BC tutoring conditions used equivalent but slightly different tests. They were equivalent in their solution structures on each test items; regardless of the strategy used, they both required the same knowledge to solve corresponding test items. The difference was the direction of writing proofs. For the FC tutoring condition, the students were asked to fill the table from top to bottom by starting with the givens. So, in a correct proof table, the givens were placed at the top of the table and the goal to prove was placed at the bottom. On the other hand, the students in the BC tutoring condition were asked to fill the table from top to bottom starting with the goal to be proved. Namely, they placed the to-be-proved goal at the top of the proof table and the givens at the bottom.

For both tutoring conditions, two tests, Test-A and Test-B, were used for the pre- and post-tests. Their use was counterbalanced so that the half of the students were assigned to use Test-A as the pre-test whereas the other half used Test-B as the pre-test. Test-A and Test-B were designed to be isomorphic in both the surface structure of test items and their solution structures and the item order on the test. That is, both tests were intended to require exactly the same geometry knowledge to be applied.

Regardless of the tutoring condition and the test version, a test consisted of 3 fill-in-blank items and 3 proof-writing items. The fill-in-blank items showed proofs with blanks that the students were supposed to fill in. There were two, one, and two blanks, respectively, on each of the three fill-in-blank items. The proof-writing items consists of one non-construction problem, one construction problem, and one far transfer problem that required a construction that is not just to connect two existing points but to extend segments. Overall, three problems in the test require construction; one fill-in-blank item and two proof-writing items.

The proof-writing test items for the FC tutoring condition showed the given propositions at the top of proof tables, hence the students in the FC condition did not have to assert given propositions into the proof table. On the other hand, the proof-writing test items for the BC tutoring condition showed a to-be-proven goal at the top of the proof tables hence the students in that condition did not have to assert a proposition to be proven.

APPENDIX E and APPENDIX F show Test-A and Test-B used for the backward tutor condition, and APPENDIX G and APPENDIX H show those for the forward tutor condition.

4.3.3. Proof problems used in the study

Besides the six problems used in the pre- and post-tests, 11 problems were used during the tutoring session. Among the 11 training problems, six required construction, which could be done by connecting existing two points. APPENDIX I shows the 11 problems used in this study.

There were 11 postulates taught in the tutoring sessions. All 17 problems (11 training problems and 6 test problems) could be solved with only those postulates. Table 4.1 shows the postulate applications necessary to solve each of the problems. In the table, an “o” shows that the corresponding postulate was used to solve the problem. An “X” shows that a postulate application required a construction. For the problems in the tests, a “?” indicates that the

corresponding postulate application was the subject of a blank to be filled. For the question #5 in the test, Test-B required additional applications of CPCTC and SSS, which appear in the parenthesis. Test-A did not require these two postulate applications. This imbalance was accidental and not a part of the experimental design.

Table 4.1: Proof problems used in the study

		CPCTC	Identity	SAS	SSS	VerAng	Z	Mtri	ASA	Trans	TriM	Coll-para	C-CP	C-EX	
Tutoring	1	o	o	o											
	2	X	o		o								X		
	3	o	o	o											
	4	X	o		o								X		
	5	o		o		o									
	6	X	o		o								X		
	7	X o		o	o	o							X		
	8	o				o	o	o	o						
	9	o				o	o	o	o						
	10							X		o	o		X		
	11							X		o	o	o	X		
Test A/B	1	?	?						o						
	2	o				?	?		?	o		o		X	
	3	o		?											
	4	o				o	o		o						
	5	X (o)	o	o	(o)		o						X		
	6	X				o	o	o	o				X	X	

5. Results

Random assignment appears to have balanced the incoming student competence across conditions. A post-evaluation analysis showed that there was no statistically significant difference in SAT math scores or in the pre-test scores between the two tutor conditions.

As shown in Table 4.1, the question #5 (a proof-writing problem) in Test-A and Test-B were not exactly identical. A post-evaluation analysis revealed that the students who took Test-B made more errors than those who took Test-A on the question #5, hence there was a main

effect for the test version on the pre-test: $t(50)=2.32$; $p=0.03$. When we excluded the question #5 from the analysis (both in pre- and post-tests) the main effect in the test version disappeared. Hence, the following analyses were done excluding question #5 from pre- and post-test unless otherwise stated.

5.1. Learning Time

During the tutoring sessions, students spent almost the same amount of time for each of the problems regardless of the condition. Figure 5.1 shows the average time spent on each problem comparing BC and FC tutor groups. A double asterisk (**) shows that the difference is statistically significant ($p<0.05$), whereas a single asterisk (*) shows a marginal difference ($p<0.1$). When there were differences, the FC students were faster than the BC students.

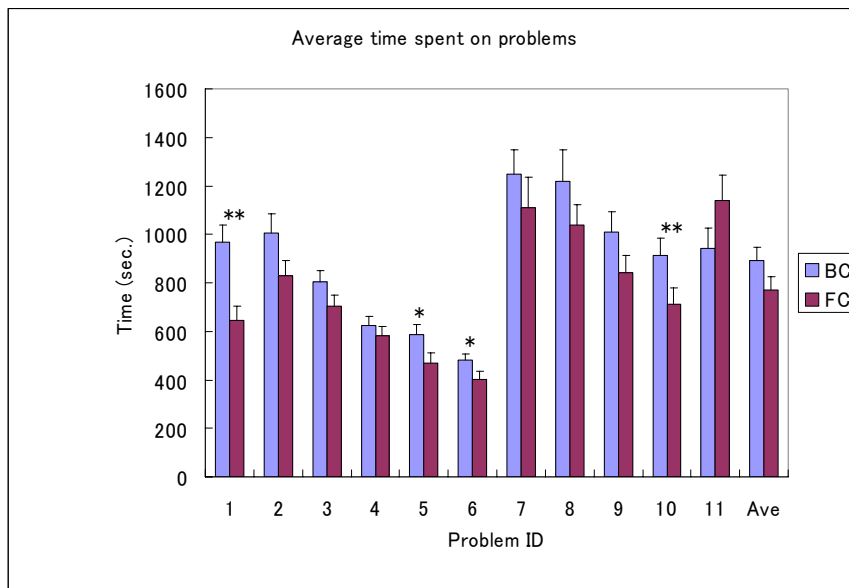


Figure 5.1: Average time spent on a problem

For the first two problems, the tutor, both BC and FC, fully provided students with modeling that showed every single inference steps, and the students merely watched the tutor’s performance and clicked [OK] button to proceed the steps. Since backward chaining requires

more steps to be performed than forward chaining, it took considerably longer for BC condition to go through first two problems.

5.2. Pre- and Post-Test Scores

Even though Test-A and Test-B are designed to be isomorphic, there was a slight difference in the length of proofs between those two tests. Also, the proofs written by backward chaining tended to be longer than those written by forward chaining, because only backward chaining must write givens as a part of the proof (those are written in the first few rows in the test for the FC condition). Hence, we used a ratio of correct proof statements to the length of a correct proof as a score of a proof-writing problem.⁴ For the fill-in-the-blank problems, correct answers were counted. The test score on fill-in-the-blank problems was then calculated as the proportion of problems answered correctly. The overall test score was the average of the proof-writing and fill-in-the-blank scores.

Figure 5.2 shows the test scores on both pre- and post-tests across the tutor and the test-version conditions. There was no main effect for the tutor on Pre-test scores. On the other hand, in the post-test, there was a main effect for the tutor: $F(1,48)=10.13$; $p<0.01$. The FC students scored higher. There was no main effect for test (Test-A vs. Test-B) nor was there an interaction with condition for either pre-test or post-test.

⁴ Precisely speaking, “correct” proof statements here means those that were coded as “on-path” or “off-path” defined in Section 5.5.1. Both on-path and off-path proof statements are true statements. The difference is that only the on-path statements are in a model proof, whereas the off-path statements are not. In other words, off-path proof statements are “reasonable,” but not a part of a correct proof.

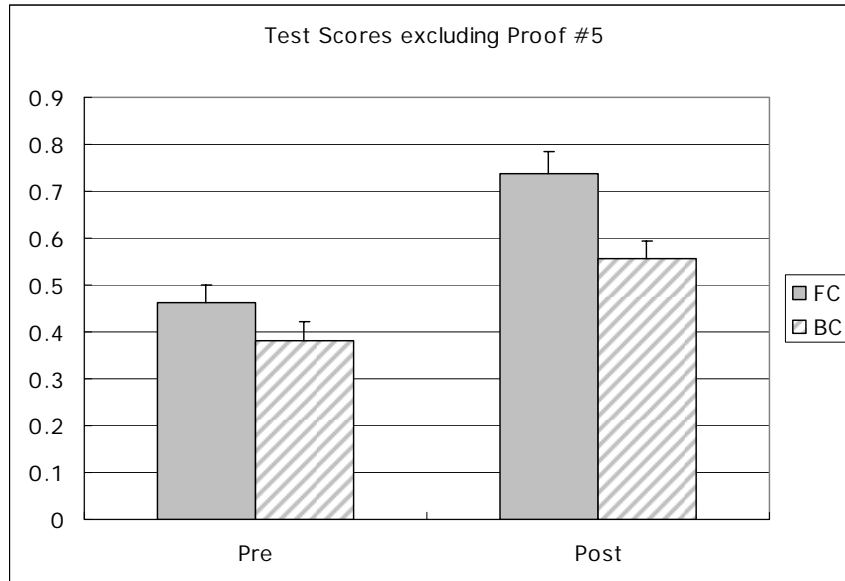


Figure 5.2: Pre- and post-test scores

A regression analysis revealed that the multiple regression equation of the post-test score upon the pre-test score and the tutor condition was:

$$\text{Post-test} = 0.50 + 0.52 * \text{Pre-test} - 0.14 \text{ (if BC)}$$

In an ANCOVA using pre-test score as the covariate, the mean adjusted post-test scores were 0.58 and 0.72 for BC and FC conditions respectively. The effect size in the difference of those scores was 0.72.⁵

In order to determine why FC students gained more than BC students, the two parts of tests (i.e., fill-in-blank and proof-writing) were examined separately. Figure 5.3 shows the average scores on the fill-in-blank test items. There was no significant difference in those scores between FC and BC on both pre- and post-test scores. There was a main effect in the test (pre- vs. post-) for both FC and BC conditions: *paired-t*(25)=-2.74; *p*=0.01 for FC, *paired-t*(25)=-3.43; *p*<0.01 for BC. That is, the students gained, but the FC and BC students gained the same amount.

⁵ $(M_{FC} - M_{BC}) / SD_{BC}$ where M_{FC} and M_{BC} are the mean adjusted post-test scores of FC and BC students, and SD_{BC} is the standard deviation of BC students.

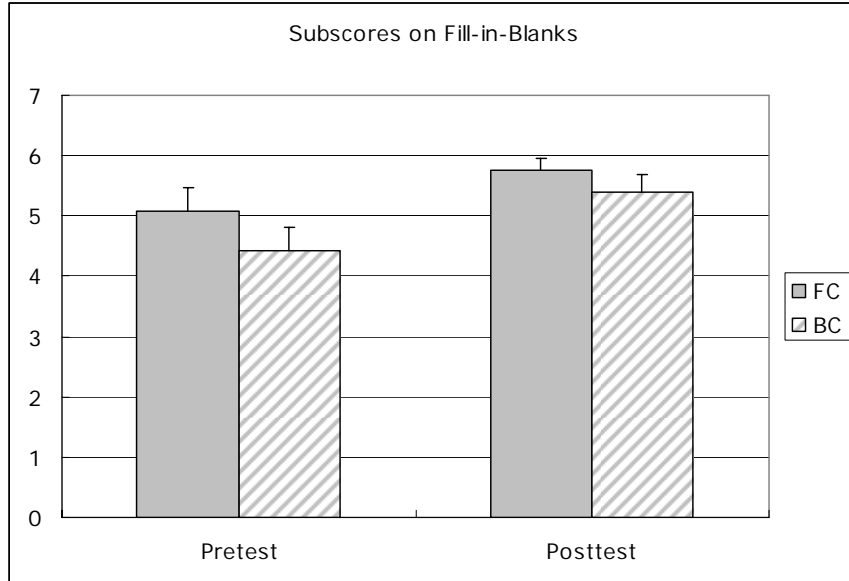


Figure 5.3: Average subscores on fill-in-blank questions

Figure 5.4 shows the average test scores on the proof-writing test items. There was no significant difference in the pre-test scores between FC and BC: $t(50)=-0.91$; $p=0.37$. On the other hand, there was a significant difference in the post-test scores between FC and BC: $t(50) = -2.53$; $p=0.02$. The effect size was 0.93.

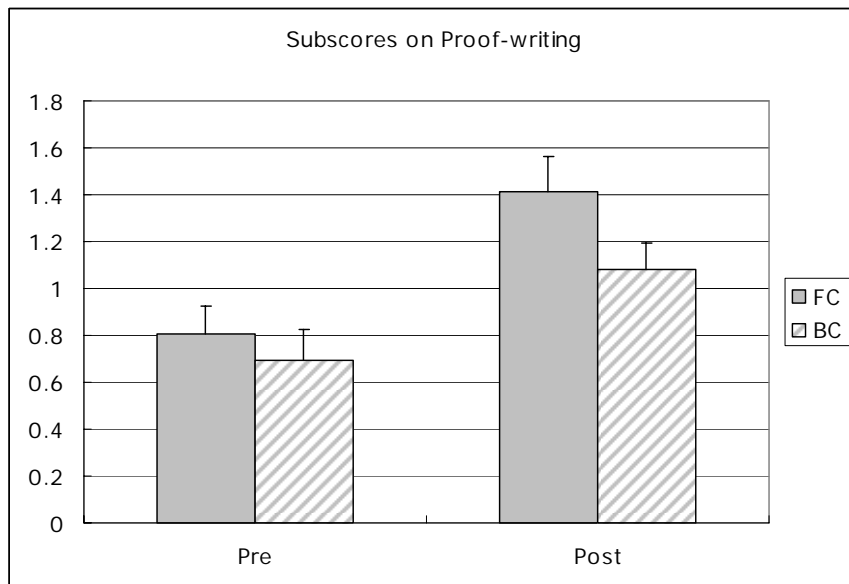


Figure 5.4: Average subscores on proof-writing questions (excluding question #5)

The difference in the overall post-test scores between BC and FC was thus mainly from the difference in their performance on proof-writing questions. The FC students wrote better proofs than the BC students. To understand why the FC students outperformed the BC students in proof writing, we further conducted two more detailed analyses on (1) an aptitude-treatment interaction, and (2) a comparison between proofs with and without construction.

To see if there was any difference in the proof-writing performance between high and low competent students, we have conducted an ANCOVA on the post-test scores on proof-writing questions (#4 and #6) with the pre-test scores on proof-writing questions as a covariate. We used the median of the pre-test scores on proof-writing (0.72) to split the students into HIGH and LOW competent groups. Figure 5.5 shows estimated marginal means on the proof-writing questions for HIGH and LOW competent students in both BC and FC tutor conditions. The interaction between the tutor condition (FC vs. BC) and the competence level (HIGH vs. LOW) was not significant: $F(1, 47)=1.18; p=0.28$.

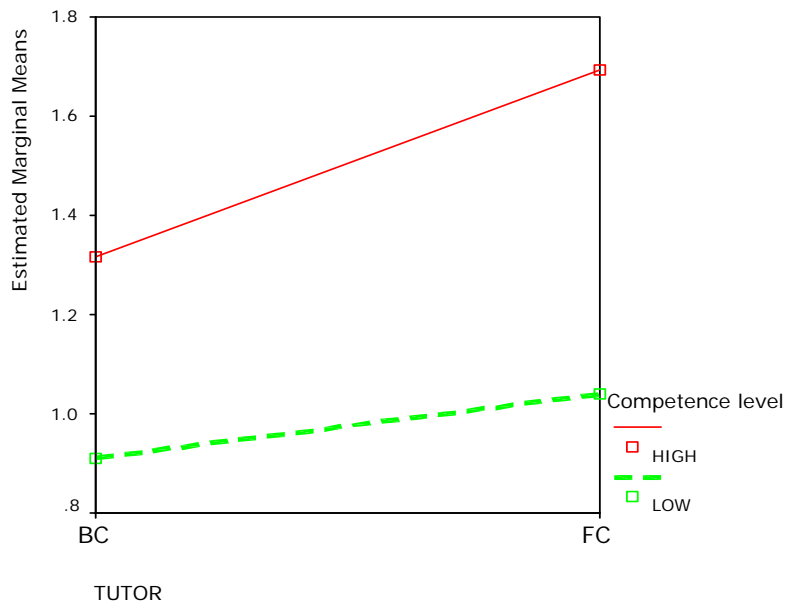


Figure 5.5: ATI analysis for proof-writing scores on the post-test

Since we exclude the question #5 for this overall analysis, there is only one proof-writing problem (the question #4) that does not involve construction and only one proof-writing problem (the question #6) that involves construction. Figure 5.6 shows mean scores on those questions. The difference in the non-construction problem between FC and BC students were not significant: $t(50)=0.66$; $p=0.51$, whereas the difference in the construction problem was significant: $t(50)=2.89$; $p<0.01$. Both FC and BC performed equally well on the non-construction problem, but the FC students outperformed the BC students on the construction problem. That we did not see difference on non-construction problems may be due to a ceiling effect; the non-construction problem was very easy for the students in both conditions. The difference in the overall proof-writing scores shown in Figure 5.4 originated in the difference in their performance on the construction problem.



Figure 5.6: Mean scores on proof-writing for non-construction and construction problems

Remaining sections provide exploratory analysis for the superior performance of FC students. First, we analyze how the students learned the postulates, and then we analyze how they wrote proofs.

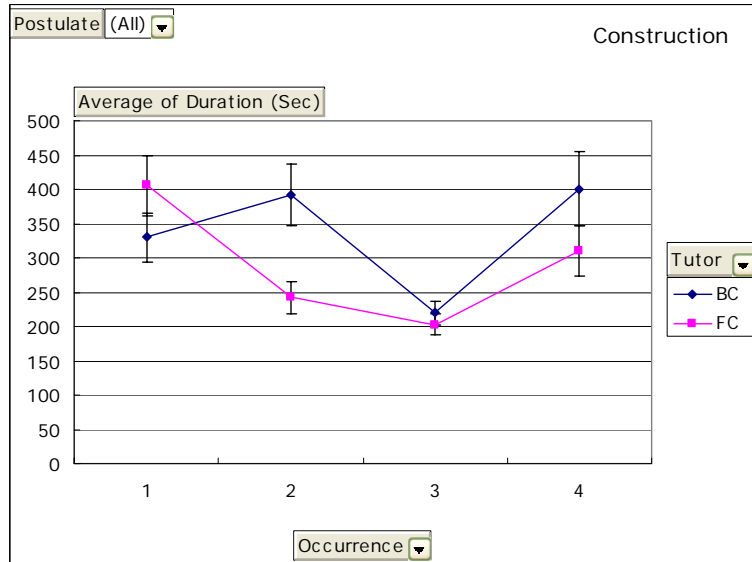
5.3. Learning the Postulates

This section summarizes students' learning on the geometric postulates. There were 11 postulates used in the study. We first show how students improve their performance on postulate applications during the tutoring sessions. We then show the difference in their skills on postulate applications between pre- and post-tests.

5.3.1. Improvement of students' performance in postulate applications

How could the students improve their skills in applying postulate? AGT recorded various sorts of students' activities during the tutoring sessions. Here we show the time and the accuracy of postulate applications. Figure 5.7 shows the average time, in seconds, to apply a postulate. Figure 5.8 shows the average number of errors made within a single postulate application. Both figures show two separate graphs each for postulate applications with construction (a) and without construction (b). The figures show an aggregated average across all students and all postulates in each category. The X-axis of the graphs is the occurrence of postulate applications; the left most data point in Figure 5.7, for example, shows the average time to apply a postulate at the first opportunity measured across all students and all postulates. A bar at each data point in the graphs shows a two-standard error interval.

a) With construction



b) Without construction

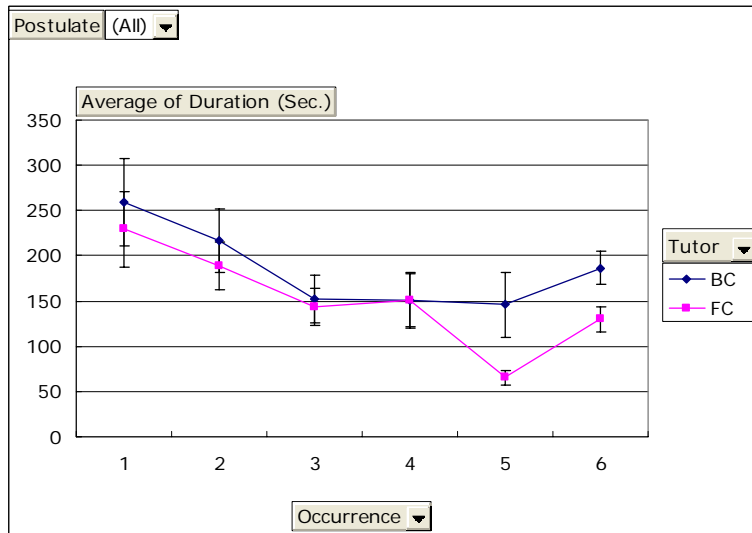
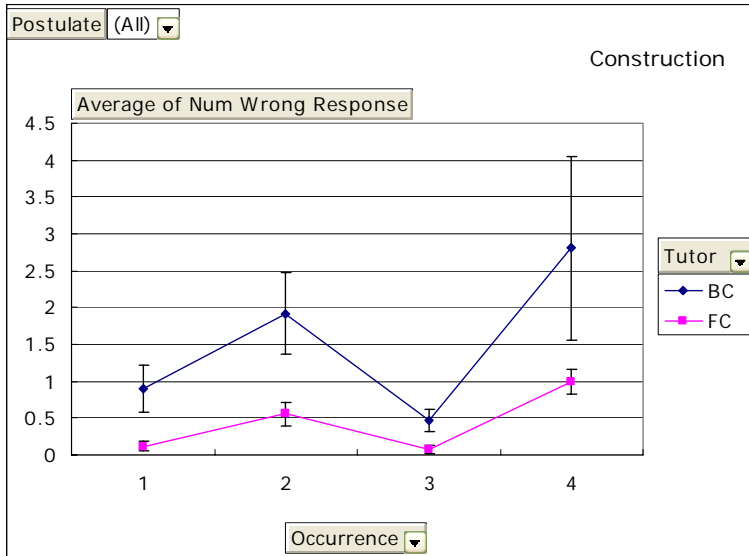
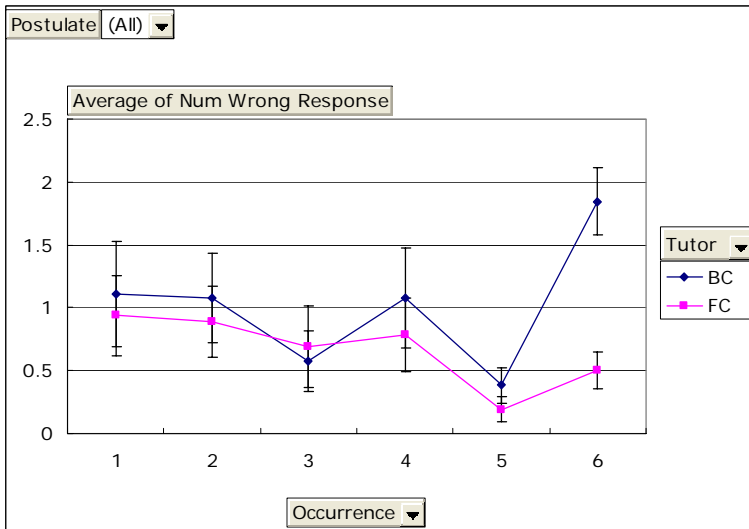


Figure 5.7: Average duration for postulate applications



a) With construction



b) Without construction

Figure 5.8: Average number of errors made during single postulate application

As shown in the figures, both tutor conditions showed a similar pattern in learning postulate applications, with a few differences. Overall, one can see gradual drop in all four graphs with a few exceptional jumps.

The irregular jumps were caused by a particular postulate. To have better understanding on learning individual postulates, APPENDIX J shows learning curves for individual postulates with and without constructions. For example, a jump at the fourth occurrence in postulate

application *with* construction was due to the fourth occurrence in CPCTC applications, which took place on the 7th training problem (see Table 4.1). We have not conducted any detailed analysis to identify what exactly cause such a jump, but the 7th training problem required two segments to be constructed while all other problems only required one segment (APPENDIX I, #7). There was also a jump at the 4th occurrence in the average number of wrong response for CPCTC *without* construction. Again, we have not inspected this particular phenomenon, but this CPCTC application also took place in the 7th training problem, which required two CPCTC applications, one with construction and the other one without construction. Apparently, the students had a difficulty to apply CPCTC to the 7th training problem.

5.3.2. Pre- and post-test difference

How well did the BC and FC students apply postulates? How did they improve their skills on postulate applications during the tutoring sessions? To see changes in their performance on postulate applications, the *mastery level* of individual postulate applications was calculated where the mastery level of postulate P is defined, for each test, as the ratio of the number of correct application of P made by a student to the total number of application of P in the correct proofs for the proof-writing problems.

First, we compared the difference between the tutor conditions. Figure 5.9 shows the comparison between BC and FC conditions on pre- and post-test in the mastery level of each postulate (again, excluding question #5). A double asterisk shows a statistically significant difference, and a single asterisk shows a statistically marginal difference. On the Pre-test, the FC students applied postulates better than the BC students: $t(50)=-2.05$; $p=0.05$. On the post-test, the tendency was weakened, but the FC students were still better at applying postulates: $t(50)=-1.95$; $p=0.06$.

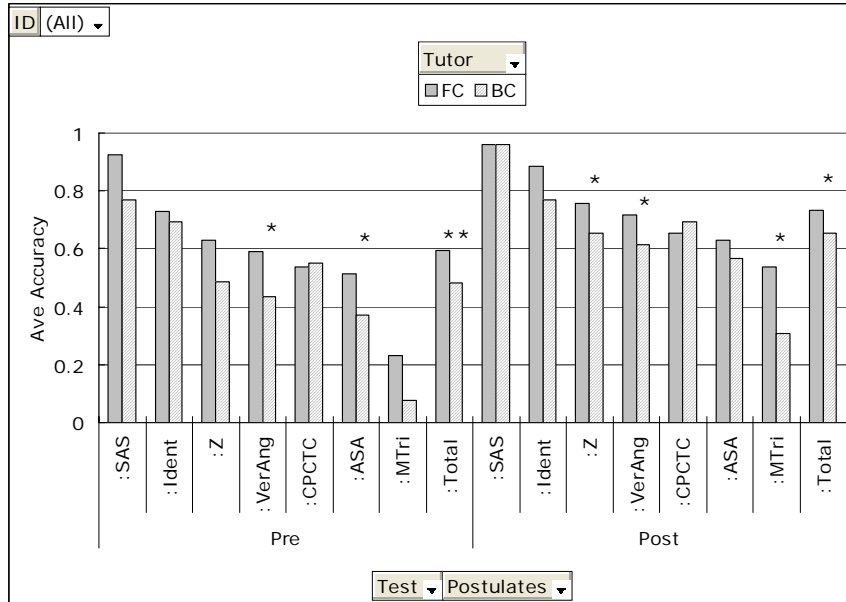


Figure 5.9: Difference between tutor conditions in accuracy of postulate applications

Second, we compared gains in the mastery level of postulate applications from the pre-test to the post-test. Figure 5.10 compares the mastery level of postulate applications between pre- and post-tests for each tutor condition. A double asterisk shows that the difference was statistically significant and a single asterisk shows a marginal difference. As can be seen in the figure, both tutor conditions showed significant improvements for most of the postulates.

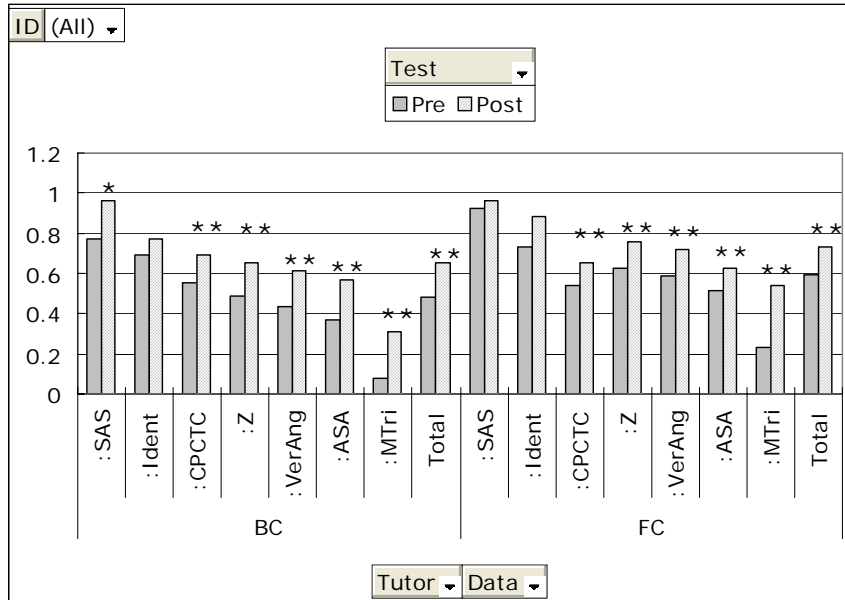


Figure 5.10: Difference in accuracy of postulate applications between tests

Averaging over all 11 postulates, the BC students improved the accuracy of their postulate applications from 0.48 on the pre-test to 0.65 on the post-test. The difference was significant: $t(50)=-4.35$; $p<0.01$. On the other hand, the FC students improved from 0.59 on the pre-test to 0.73 on the post-test. The difference was also significant: $t(50)=-3.91$; $p<0.01$. In an ANOVA, there was no main effect for the difference in the amount of gain between BC and FC: $F(1, 48)=0.50$; $p=0.48$. Thus, BC and FC students equally improved the mastery level of postulate application.

It must be noticed that the mastery level used in this section does not necessarily reflect students' competence on each postulate's application. This is because a failure in applying one postulate may hinder other postulate applications that are "descendants" of the failed postulate application. For example, consider a simple chain of reasoning $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$. If one failed to carry out the first derivation, then it is likely that he/she would also fail to apply the

second and the third derivation. The mastery level defined above suffers from this problem. Nonetheless, we can still read trends in the improvement made by the students.

5.4. Strategy Used to write Proofs in the Post-test

Some students who had been assigned to the BC tutoring condition used a forward chaining strategy for some proof-writing questions on the post-test, and vice versa. The direction of a proof was considered to be inconsistent with the assigned tutoring condition when one of the following conditions was met:

(1) The order of propositions in the proof table: Sometimes the order of propositions was reversed. For example, in the FC tutoring condition, correct proofs have givens on the top and the goal on the bottom. If a proof table showed the goal at the top of the proof table, then that proof was considered to be written with backward chaining.

(2) A gap between the top lines and the “body” of the proof: Since the given or the to-be-proven propositions were placed at the top of a proof table according to the assigned problem solving strategy, when the students took an inconsistent strategy, they tended to start writing at the bottom. As a consequence, there was often a big gap between the givens, in the case of forward chaining, or the to-be-proven goal, in the case of backward chaining, and the first line of the proof written by the student. APPENDIX K shows examples of inconsistent proofs.

Figure 5.11 shows the number of proofs written in a discrepant strategy to the assigned tutoring condition (e.g., BC students used forward chaining). As shown in the figure, 17 out of 52 (33%) of the pre-test proofs in the BC tutor condition were written in forward chaining by ten (out of 26) different students. Three of those ten students plus one new student in BC condition used the FC strategy on the post-test. On the other hand, there was only one student in FC tutor condition who used the BC strategy on the pre-test (i.e., the two proofs shown in figure were

written by this student). It was the same student who applied a discrepant strategy on the post-test problems in the FC tutor condition.

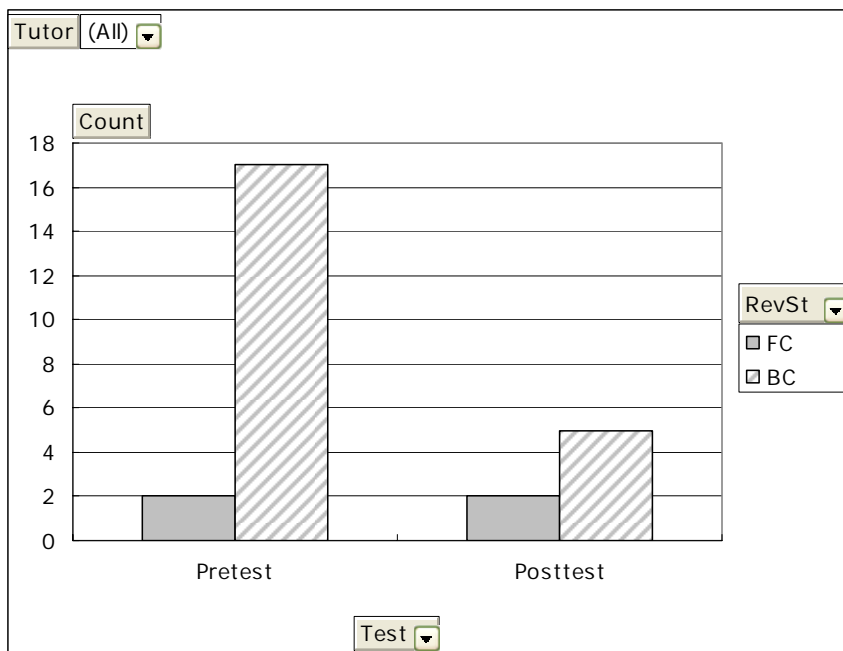


Figure 5.11: Number of proof written in opposite strategy (Max=52)

The above observation suggests that the students preferred forward chaining to backward chaining. This could be either because that they found forward chaining easier than backward chaining, or they were simply more familiar with forward chaining. An unofficial interview for some students indicated that they were taught forward chaining when they learned geometry theorem proving in a high school.

5.5. Students' Performance in Proof Writing

This section provides a detailed analysis on performance on proof writing on the post-test. Since the difference in the test versions (Test-A and Test-B) is not an issue for the analysis here, we included the question #5, which lacks the isomorphism in Test-A and Test-B. However, since we want to concentrate only on those who used the proof strategy taught during the

tutoring sessions (not something that they happened to have learned in the past), we only deal with proofs that were written with the proof strategies with the assigned tutor. Namely, the proofs shown in Figure 5.11 are excluded from the analyses in this section. We call the proof strategies with the assigned tutor the *TStrategy* hereafter. In sum, on the post-test, the BC students wrote 69 proofs with their TStrategy (BC) and the FC students wrote 75 proofs with their TStrategy. Of those, the BC students wrote 27 (39%) correct proofs and the FC students wrote 46 (61%) correct proofs.

What does a “correct” proof mean? We first provide the basic definitions of terms used to codify students’ proofs. We then provide detailed analysis of those proofs.

5.5.1. Basic definitions for coding schema

A *proof* consists of *proof statements*, and those are the unit of analysis used in the following subsections to analyze students’ performance on proof writing.

Proof statements, each of which consist of a proposition, a justification, and premises (see 2.1), were coded as follows:

On-path: The proof statement is a part of a correct proof.

Off-path: The proof statement is not a part of correct proof, and the following conditions hold: the proposition may or may not be true, but the postulate used as a justification has a consequence that unifies with the proposition, and the premises support the proposition. The individual premises may or may not be true.

Wrong: The proof statement is neither on-path nor off-path.

Missing: We also took into account the proof statements that are supposed to be in the proof, but were not mentioned at all. Those statements were coded as *missing*.

The on-path and off-path proof statements are often called *reasonable* hereafter, because they are equally plausible at the time they were entered due to the nondeterministic characteristic of the task.

Based on those coding, which are italicized in the description below, the proofs were coded as follows:

Correct: The proof contained a chain of *on-path* proof statements from given propositions to the goal. The proof may include extra *off-path* and *wrong* statements that have no connection to the chain of on-path proof statements.

Wrong: The proof contained a chain of proof statements from given propositions to the goal, but the chain involves at least one not *on-path* proof statement.

Stuck: The proof did not contain a chain of proof statements. The proof may contain any type of proof statements.

Incomplete: All proof statements in the proof were *reasonable*, but the proof lacks a chain from given propositions to the goal due to a lack of one and only one proof statement. An example incomplete proof involves a proof statement that applies the SSS axiom for triangle congruent and concluded segment congruence instead of triangle congruence, and the triangle congruence is not in the proof table.

Blank: No attempt was made at all.

The incomplete proofs could be coded as “stuck,” but we treated them separately because (1) they only contained reasonable statements hence (2) the student could have just made a slip, which means that the student probably did not get stuck.

5.5.2. Analysis of proofs

What types of incorrect proofs did they write? Figure 5.12 shows the number of occurrence of each type of the proofs in each of the tutor conditions. “OD” shows the number of proofs in each tutor conditions that were not written in TStrategy. Hence the remaining part of the figure actually shows the number of proofs written in TStrategy. The figure clearly shows that the FC students wrote more correct proofs than the BC students. It also shows that both BC students and FC students were equally likely to commit “Wrong” proofs. Aggregating “Stuck” and “Blank” proofs, however, BC students were more likely to get stuck.

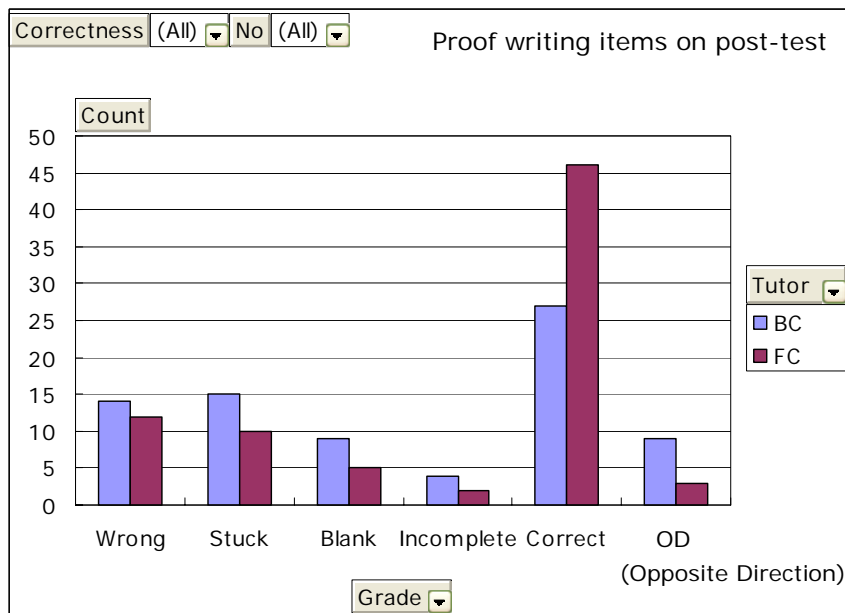


Figure 5.12: Classification of incorrect proofs

To see if the performance of the BC students writing correct proofs was inferior to the FC students, we ran a *t*-test on the individual student’s number of proofs that were not correct, which varied from 0 to 3. Figure 5.13 shows the average number of incorrect proofs written by the students in each tutor condition. The difference is statistically significant: $t(50)=2.01$; $p=0.02$.

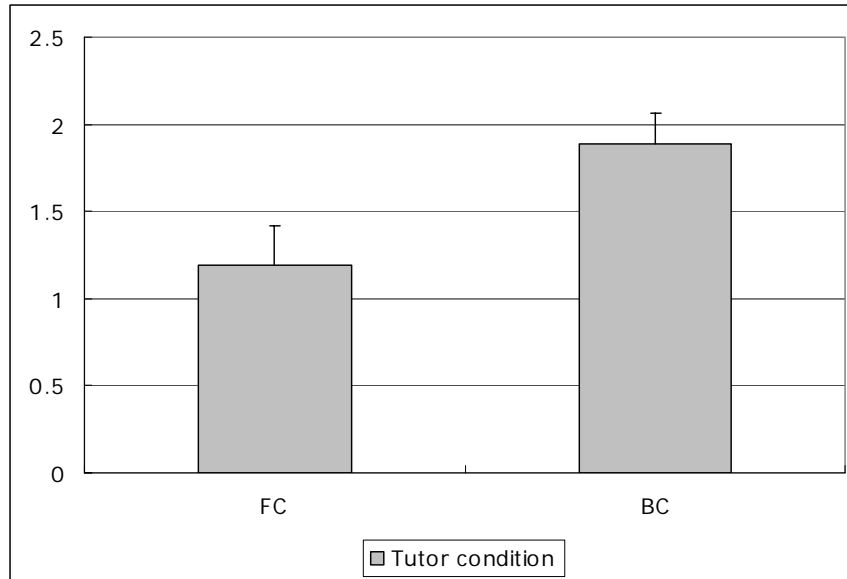


Figure 5.13: Portion of incorrect proofs on the post-test

Again, we ran a comparative analysis on students' performance in proof-writing between construction and non-construction problems. Table 5.1 shows three 2x2 Contingency Tables comparing correctness of proofs and the strategy used (TStrategy) for each of the proof-writing problems (NO). For correctness, the variable Correct is coded as 1 for the correct proofs and 0 for all other type of proofs. We ran a Chi-square test for each of the three Contingency Tables separately. The Chi-square tests did not revealed statistically significant relationship between the correctness and the strategy used for the problem #4 (non-construction problem: $\chi^2=0.94$, $p=0.33$) and #5 (construction problem: $\chi^2=2.62$, $p=0.11$). On the other hand, for the problem #6 (construction problem) there was a significant relationship between the correctness of the proofs and the strategy used ($\chi^2=7.13$, $p<0.01$). Both BC and FC students performed equally well on the non-construction problem (#4). Their performance on a construction problem (#5) was not significantly different, but the trend was that the FC students wrote more correct proofs than the BC students. On the other hand, the FC students outperformed BC students on another construction problem (#6). The question #6 involved a construction technique (to extend a

segment). It also required applying postulates (Z, MTri, and ASA) that were introduced late in the training sessions hence had fewer opportunities to practice. Hence, the question #6 might be more “difficult” for students to solve, and that difficulty might be a potential source of difference in students’ performance.

Table 5.1: 2x2 Contingency Table on the correctness of the proofs

STRATEGY * Correct * NO Crosstabulation

NO	STRATEGY			Correct		Total
				0	1	
4	STRATEGY BC	Count	8	17	25	
		Expected Count	6.5	18.5	25.0	
	FC	Count	5	20	25	
		Expected Count	6.5	18.5	25.0	
	Total	Count	13	37	50	
		Expected Count	13.0	37.0	50.0	
5	STRATEGY BC	Count	14	8	22	
		Expected Count	11.2	10.8	22.0	
	FC	Count	10	15	25	
		Expected Count	12.8	12.2	25.0	
	Total	Count	24	23	47	
		Expected Count	24.0	23.0	47.0	
6	STRATEGY BC	Count	20	2	22	
		Expected Count	15.9	6.1	22.0	
	FC	Count	14	11	25	
		Expected Count	18.1	6.9	25.0	
	Total	Count	34	13	47	
		Expected Count	34.0	13.0	47.0	

Problem NO = 4 is a non-construction problem. Problems NO = 5 & 6 are construction problems.

We also ran same kind of Contingency Table analyses on the other type of proofs (e.g., Wrong, Stuck, Blank, and Incomplete), but there were no statistically significant relationship between the tutor conditions and the type of proofs other than the one shown in Table 5.1

In sum, both FC and BC students performed equally well on the non-construction problem, but the FC students did better than BC students on the construction problems. The next sections discuss what part of backward chaining makes it more likely fail to find a correct proof.

5.5.3. Analysis of proof statements

There were 479 proof statements (215 and 264 in the BC and FC conditions respectively) appearing on the post test. Of those 479 proof statements, 400 were reasonable and 79 were wrong statements. 180 statements were missing (92 and 88 in the BC and FC conditions). Figure 5.14 compares the number of proof statements of each type made by FC and BC students.

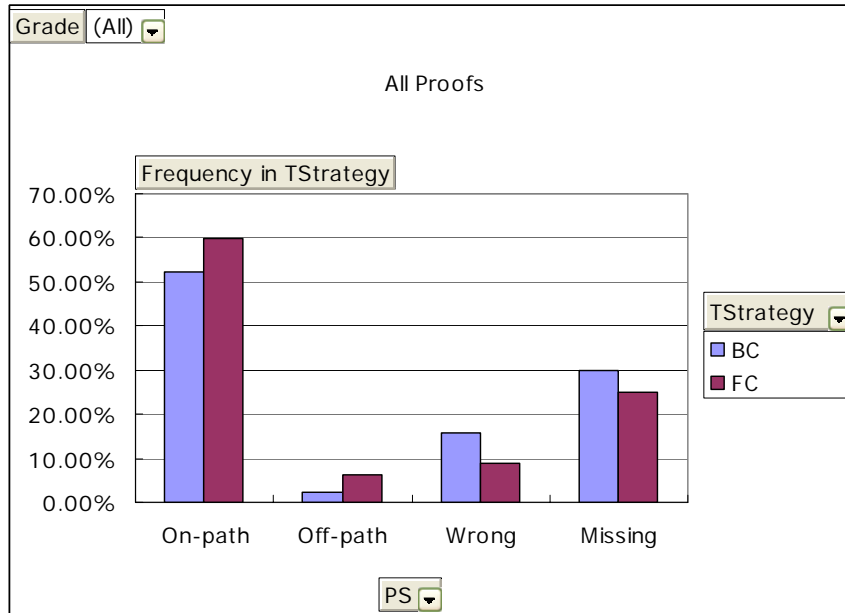


Figure 5.14: Classification of proof statements

Surprisingly, the students seldom made off-path statements. Since AGT did not teach search control or heuristics, even when they ended up with a correct proof, the students could make off-path proof statements, which by definition are reasonable proof statements but are not a part of correct proofs. Still the number of off-path statements shown in Figure 5.14 is fairly small compared to on-path statements. When the students eventually wrote correct proofs, did they struggle at all? To answer this question, we conducted a Contingency table analysis (Table 5.2) to compare the difference in off- and on-path statements across the TStrategies.

Table 5.2: 2 x 2 Contingency tables on on-path and off-path proof statements

TStrategy * PS Crosstabulation

			PS		Total
			Off-path	On-path	
TStrategy	BC	Count	7	160	167
		Expected Count	12.1	154.9	167.0
	FC	Count	22	211	233
		Expected Count	16.9	216.1	233.0
Total	Count		29	371	400
	Expected Count		29.0	371.0	400.0

a) All proofs

TStrategy * PS Crosstabulation

			PS		Total
			Off-path	On-path	
TStrategy	BC	Count	0	111	111
		Expected Count	1.1	109.9	111.0
	FC	Count	3	178	181
		Expected Count	1.9	179.1	181.0
Total	Count		3	289	292
	Expected Count		3.0	289.0	292.0

b) Correct proofs

TStrategy * PS Crosstabulation

			PS		Total
			Off-path	On-path	
TStrategy	BC	Count	7	49	56
		Expected Count	13.5	42.5	56.0
	FC	Count	19	33	52
		Expected Count	12.5	39.5	52.0
Total	Count		26	82	108
	Expected Count		26.0	82.0	108.0

c) Incorrect proofs

There was a significant difference in the appearance of on-path and off-path statements when all proofs were aggregated (Table 5.2 a); $\chi^2 = 3.99$, $df = 1$, $p = 0.046$. The difference was not significant for correct proofs (Table 5.2 b); p on Fisher's Exact Test = 0.29, but remains significant for incorrect proofs (Table 5.2 c); $\chi^2 = 8.52$, $df = 1$, $p < 0.01$.

In the 46 correct proofs written by the FC students, there were only 3 off-path statements. In the 27 correct proofs written by the BC students, there were no off-path statements. Surprisingly, when the students wrote correct proofs, they seldom wrote off-path proof statements. When they wrote incorrect proofs, the students in both conditions made off-path statements, but FC students made more off-path statements than BC students. This observation

agrees with our initial prediction that forward chaining would produce more off-path statements. Yet the number of off-path statements made by FC students was quite small (19).

Did the proof statements for construction differ from the ones for non-construction? More precisely, when the students apply postulates, did they make more off-path statements for the postulate applications that involve constructions than those that do not? Do FC students made more off-path statements than BC students on the postulate applications with constructions? To answer those questions, we broke down the 2 x 2 Contingency tables shown in Table 5.2 to compare the appearance of on-path and off-path proof statements that do and do not involve constructions.

Table 5.3: 2 x 2 Contingency table on on-path and off-path statements comparing construction vs. non-construction problems

TStrategy * PS * CONST Crosstabulation

CONST				PS		Total
				Off-path	On-path	
No	TStrategy	BC	Count	7	140	147
			Expected Count	12.2	134.8	147.0
		FC	Count	22	180	202
			Expected Count	16.8	185.2	202.0
	Total		Count	29	320	349
			Expected Count	29.0	320.0	349.0
Yes	TStrategy	BC	Count		20	20
			Expected Count		20.0	20.0
		FC	Count		31	31
			Expected Count		31.0	31.0
	Total		Count		51	51
			Expected Count		51.0	51.0

Table 5.3 shows number of on-path and off-path statements appeared in the post test (both correct and incorrect proofs), comparing the ones that involve construction (CONST = Yes) and that do not (CONST = No). As shown in the table, there were 349 postulate applications that did not involve constructions (they may or may not be a part of a construction problem), whereas 51 postulate applications required constructions to apply. For the proof statements that do not involve construction, the difference in appearance of on-path and off-path statements was

significant: $\chi^2 = 4.20$, $p=0.04$. Surprisingly, when the students apply a postulate with construction, they did not make any off-path attempt, that is, when they made constructions, the students always apply postulates in the way a model proof does.

Table 5.4: 2 x 2 Contingency tables on on-path and off-path statements comparing construction vs. non-construction problems in correct and incorrect proofs

TStrategy * PS * CONST Crosstabulation

CONST				PS		Total
				Off-path	On-path	
No	TStrategy	BC	Count	0	101	101
			Expected Count	1.2	99.8	101.0
		FC	Count	3	152	155
			Expected Count	1.8	153.2	155.0
	Total		Count	3	253	256
			Expected Count	3.0	253.0	256.0
Yes	TStrategy	BC	Count		10	10
			Expected Count		10.0	10.0
		FC	Count		26	26
			Expected Count		26.0	26.0
	Total		Count		36	36
			Expected Count		36.0	36.0

a) Correct proofs

TStrategy * PS * CONST Crosstabulation

CONST				PS		Total
				Off-path	On-path	
No	TStrategy	BC	Count	7	39	46
			Expected Count	12.9	33.1	46.0
		FC	Count	19	28	47
			Expected Count	13.1	33.9	47.0
	Total		Count	26	67	93
			Expected Count	26.0	67.0	93.0
Yes	TStrategy	BC	Count		10	10
			Expected Count		10.0	10.0
		FC	Count		5	5
			Expected Count		5.0	5.0
	Total		Count		15	15
			Expected Count		15.0	15.0

b) Incorrect proofs

Table 5.4 shows the similar Contingency tables with comparison between construction and non-construction problems. The relationship between the appearance of on-path/off-path statements and the tutor condition (TStrategy) was not significant for the correct proofs (Table a: p for Fisher's Exact Test = 0.28), but it was significant for incorrect proofs (Table b: $\chi^2 = 7.34$; p

< 0.01). When the postulate application did not involve construction, the FC students wrote more off-path statements for incorrect proofs.

5.5.4. False subgoaling in backward chaining

Unlike forward chaining, backward chaining could apply a postulate with the premises that are false (i.e., the propositions that would never be justified) while its consequence matched with a proposition being justified. The proof statements in this type were coded as “off-path” in our coding schema shown in section 5.5.1.

It must be emphasized that this type of off-path statement does not appear in forward chaining. This is because forward chaining always apply a postulate with true premises, hence never enters false proposition into the proof table. Backward chaining, on the other hand, enters unjustified propositions that could be false. Especially, students could enter false propositions that *look like* they are true in the problem figure.

One could hypothesize that this *false subgoaling* might be the source of the difficulty in backward chaining. In other words, the students could enter many false propositions as the premises of backward postulate applications, and that could make a proof intractable.

Surprisingly, there were no false subgoaling occurred in the BC students’ proofs at all. All the off-path statements made by BC students involved only true propositions. One possible account for not observing false subgoaling is that since we had provided a problem figure in which all the given propositions are quantitatively satisfied (see section 2.2.2), there were coincidentally few false propositions that looked true hence the propositions that the students guessed to be true were indeed true.

5.6. Analysis of Postulate Applications

One of the interesting findings is that the BC students wrote less reasonable proof statements than the FC students (Figure 5.14). The BC students tended to write wrong proof statements more often than the FC students, but most of the time, the BC students apparently just gave up a proof (Figure 5.12).

In order to understand why the BC students made fewer reasonable proof statements, we coded each of the 79 wrong statements identified in the preceding section as a triplet of independent codes of (1) the proposition, (2) the justification, and (3) the premises, which are three constituent of a proof statement. For each proof statement, we coded each instance of a proposition, a justification, and the premises independently as follows:

On-path: A proposition, justification, and premise were coded independently as “on-path” when they appeared in the correct proof.

Off-path: A proposition was coded as “off-path” when it was true in the given problem configuration, but not a part of the correct proof. A justification was coded as “off-path” when it was relevant to proving a proposition to be justified (i.e., its consequence unifies with the proposition), but does not indeed appear in the correct proof. “Off-path” premise(s) correctly support a justification, but are not a part of proof (e.g., a different combination of side-angle-side for triangle congruence). In backward chaining, off-path premises may or may not be true propositions.

Wrong: A proposition was coded as “wrong” when it does not hold in the problem configuration. “Wrong” justifications do not have a consequence that matches

with a proposition being justified. “Wrong” premises do not support a justification.

Blank: A proposition and justification were coded independently as “blank” if they were not written (left blank). Premises, which were supposed to be the line numbers of the corresponding proof statements, were coded as “blank” only when the appropriate statements were not in a proof table. This means that even when a proof statement did not have the line numbers that indicate premises, if the premise statements were all in the proof table, then the premises were coded as “On-path.” This conservative coding schema was adopted to prevent that were carelessly forgotten from being coded as “Blank.”

The unit of analysis here is an instance of a proposition, a justification, or premises in the proof statements. Hence a proof statement can have a triplet of any combination of the codes of these instances. For example, a reasonable proof statement may have both a proposition and a justification coded as “on-path,” but the premises coded as “off-path.” The rest of the current section describes the difference in appearance of proposition, justification, and premises separately.

Figure 5.15 shows the frequency of each type of proposition. To test the difference in the usage of propositions, we built a 2 x 3 contingency table (Table 5.5 a); two rows of Forward and Backward chaining, and three columns of On-path, Off-path, and Wrong usage of the proposition (there was no proof statement with no proposition). A Chi-Square test on the 2 x 3 contingency table showed no significant difference in the type of propositions (p of Fisher’s Exact Test=0.48). The lack of difference remained when the proof statements were further broken

down into the ones that did and did not involve constructions (Table 5.3 b): p of Fisher's Exact Text = 0.38.

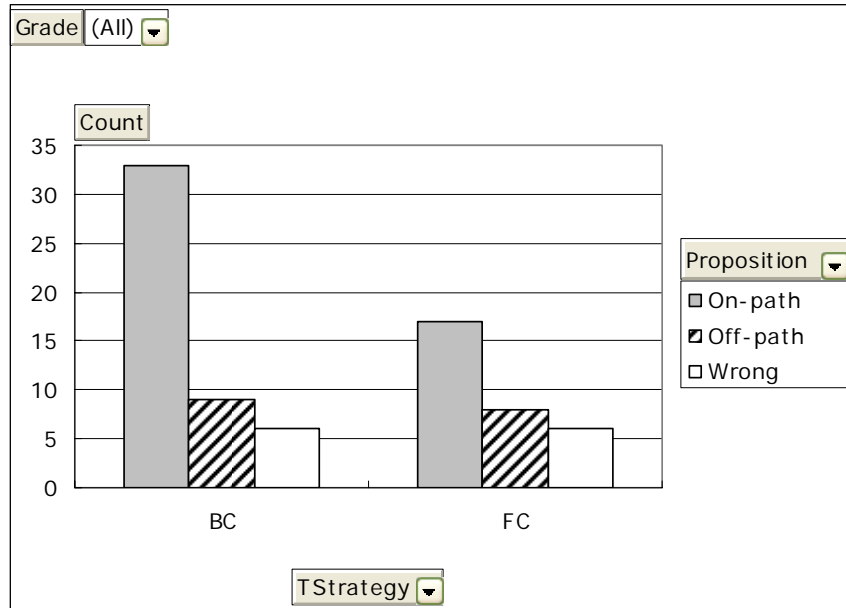


Figure 5.15: Type of propositions

Table 5.5: A 2 x 3 Contingency table on the type of propositions

a) All proof statements

TStrategy * Proposition Crosstabulation

		Proposition			Total	
		Off-path	On-path	Wrong		
TStrategy	BC	Count	9	33	6	48
		Expected Count	10.3	30.4	7.3	48.0
	FC	Count	8	17	6	31
		Expected Count	6.7	19.6	4.7	31.0
Total		Count	17	50	12	79
		Expected Count	17.0	50.0	12.0	79.0

b) Comparison between statements that do and do not involve constructions.

The statements may or may not be a part of a construction problem.

TStrategy * Proposition * Construction Crosstabulation

Construction		Proposition			Total		
		Off-path	On-path	Wrong			
No	TStrategy	BC	Count	9	27	6	42
			Expected Count	10.5	24.1	7.4	42.0
		FC	Count	8	12	6	26
			Expected Count	6.5	14.9	4.6	26.0
	Total		Count	17	39	12	68
			Expected Count	17.0	39.0	12.0	68.0
Yes	TStrategy	BC	Count		6		6
			Expected Count		6.0		6.0
		FC	Count		5		5
			Expected Count		5.0		5.0
	Total		Count		11		11
			Expected Count		11.0		11.0

Figure 5.16 shows the frequency of types of justifications. A Fisher's Exact Test on a 2 x 4 contingency table (Table 5.6 a) did not show a significant difference in the usage of justification between forward and backward chaining ($p=0.18$). The trends remained when the proof statements were further broke down into the ones that did and did not involve constructions (Table 5.6 b): p for Fisher's Exact Test = 0.16 for non-construction proof-statements, and 1.00 for construction proof-statements.

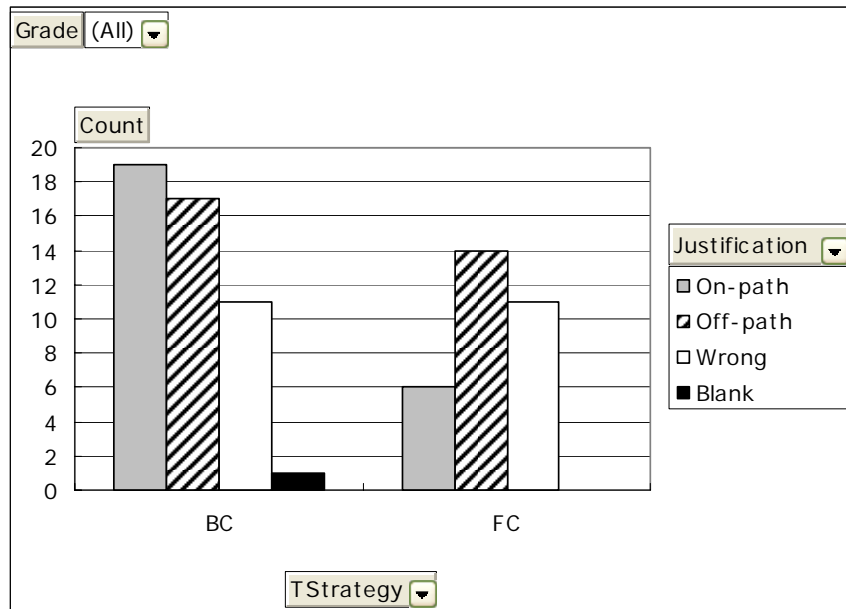


Figure 5.16: Type of justification

Table 5.6: A 2 x 4 Contingency table on the type of justification

a) All proof statements

TStrategy * Justification Crosstabulation

			Justification				Total
			Blank	Off-path	On-path	Wrong	
TStrategy	BC	Count	1	17	19	11	48
		Expected Count	.6	18.8	15.2	13.4	48.0
	FC	Count	0	14	6	11	31
		Expected Count	.4	12.2	9.8	8.6	31.0
Total	Count		1	31	25	22	79
	Expected Count		1.0	31.0	25.0	22.0	79.0

b) Comparison between statements that do and do not involve constructions.

TStrategy * Justification * Construction Crosstabulation

Construction				Justification				Total
				Blank	Off-path	On-path	Wrong	
No	TStrategy	BC	Count	1	15	16	10	42
			Expected Count	.6	17.3	11.7	12.4	42.0
		FC	Count	0	13	3	10	26
			Expected Count	.4	10.7	7.3	7.6	26.0
	Total	Count		1	28	19	20	68
		Expected Count		1.0	28.0	19.0	20.0	68.0
Yes	TStrategy	BC	Count		2	3	1	6
			Expected Count		1.6	3.3	1.1	6.0
		FC	Count		1	3	1	5
			Expected Count		1.4	2.7	.9	5.0
	Total	Count			3	6	2	11
		Expected Count			3.0	6.0	2.0	11.0

Figure 5.17 shows the frequency of types of premises. A Fisher's Exact test on a 2 x 4 contingency table (Table 5.7 a) revealed a significant difference: Fisher's Exact Test = 7.25; $p=0.04$. The BC students left more blank premises than the FC students. That is, BC students were more likely to fail to provide premises. This trend is especially prominent when the postulate application involves construction (Table 5.7 b) whose p -value for the Fisher's Exact Test is 0.06 whereas the one for postulate applications that do not involve construction is 0.16.

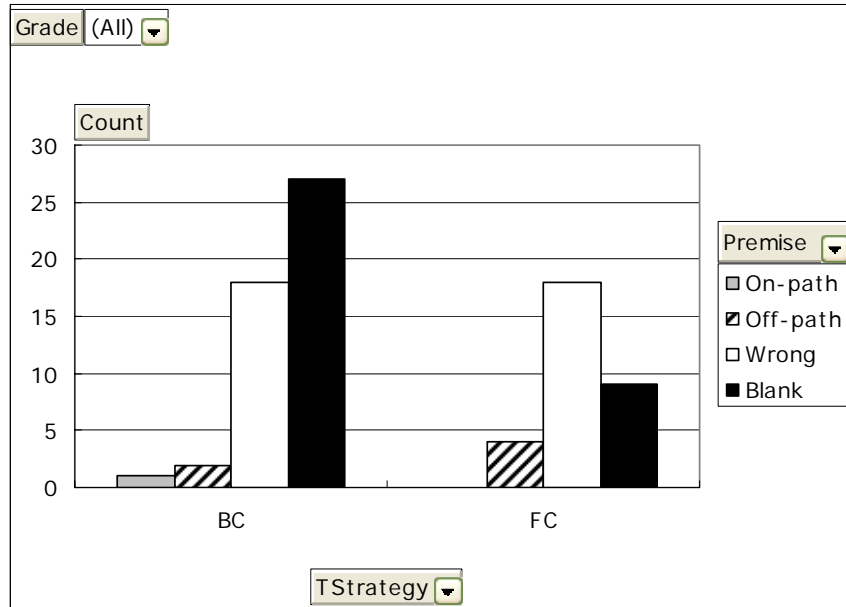


Figure 5.17: Usage of premise

Table 5.7: A 2 x 4 Contingency table on the use of premises

a) All proof statements

TStrategy * Premises Crosstabulation

			Premises				Total
			Blank	Off-path	On-path	Wrong	
TStrategy	BC	Count	27	2	1	18	48
		Expected Count	21.9	3.6	.6	21.9	48.0
	FC	Count	9	4	0	18	31
		Expected Count	14.1	2.4	.4	14.1	31.0
Total	Count		36	6	1	36	79
	Expected Count		36.0	6.0	1.0	36.0	79.0

b) Comparison between statements that do and do not involve constructions.

TStrategy * Premises * Construction Crosstabulation

				Premises				Total
				Blank	Off-path	On-path	Wrong	
No	TStrategy	BC	Count	23	2	1	16	42
			Expected Count	19.8	3.7	.6	17.9	42.0
		FC	Count	9	4	0	13	26
			Expected Count	12.2	2.3	.4	11.1	26.0
	Total	Count		32	6	1	29	68
		Expected Count		32.0	6.0	1.0	29.0	68.0
Yes	TStrategy	BC	Count	4			2	6
			Expected Count	2.2			3.8	6.0
		FC	Count	0			5	5
			Expected Count	1.8			3.2	5.0
	Total	Count		4			7	11
		Expected Count		4.0			7.0	11.0

To see the use of premises in details, we conducted a Chi-square test on a 2 x 2 contingency table whose columns consists of the blank premises and the aggregation of remaining types. The analysis revealed that BC students tended to leave the premises blank more often than FC students ($\chi^2 = 5.63$; $p=0.02$ for all proof statements, $\chi^2 = 2.61$; $p=0.11$ for proof-statements without construction, and p for Fisher's Exact Test is 0.05 for the proof-statements with construction). A similar 2 x 2 Chi-square test showed that the BC students also tended to use wrong premises more often than FC (Fisher's exact test = 3.21, $p=0.06$). Other 2 x 2 analyses showed no significant difference in the use of neither off-path or on-path premises between the FC and BC students.

6. Discussion

After proving 11 geometry theorems with intensive aid from AGT, the students in the FC and the BC conditions showed different performance on proof writing in the post-test. This chapter discusses those differences in both theoretical and pedagogical aspects. More specifically, the following sections discuss the relationship between conceptual and strategic knowledge, the source of difficulties in backward chaining, a relation between the cognitive load theory and the current study, and the complexity of the search.

6.1. Learning Domain Concept does not secure Proof-writing Skills

There were no significant between-condition differences in the accuracy of postulate applications on the post-test (Figure 5.9). FC and BC students improved their accuracy of postulate applications equally well (Figure 5.10). Both conditions showed a similar pattern in learning progress (i.e., the learning curve; Figure 5.7 and Figure 5.8). Still, the FC students outperformed the BC students in writing proofs.

These findings support a claim that it is not sufficient to understand domain principles, namely, the concept of each geometric postulate. Knowing about the knowledge to apply is one thing; knowing how to apply it is another. Learning strategic skills (how to write a proof, how to apply postulates, etc) apparently requires more elaborate practice. And this is indeed the place where the difference between forward and backward chaining took place. The following sections discuss potential source of difficulties in backward chaining.

6.2. Difficulty in Thinking Backwards: Subgoaling

Like other studies mentioned in Section 2.4, the students in this study showed better performance on FC than BC. However, the current study narrowed down the factors of students' difficulty on working backwards. The most striking finding is that a potential source of the difficulty in backward chaining lies in the difference between the ways forward and backward chaining assert proof statements.

The BC students tended to get stuck at proving premises even when they picked a correct proposition and a postulate. In other words, it seems to be difficult for BC students to specify *subgoals* as the to-be-justified propositions to support a postulate application. As mentioned in the definition of coding schema for premises (Section 5.6), the students could just forget to write line numbers even when they indeed asserted corresponding propositions in the proof table. However, premises were coded as "On-path" if all corresponding propositions appeared in the proof table, which happened only once in the BC condition. Although we can not totally disregard the possibility that the students had simply forgotten to enter and specify premises, it is also unlikely that only the BC students had suffered from such carelessness. Hence, the BC students apparently have indeed had difficulty in subgoaling.

Subgoalings requires the students to write into the table one or more propositions (i.e., premises) that have yet to be proved. At the time they are entered into the proof table, those propositions are not “true” assertions, but just hypothesis to be proved. This uncertainty may increase the chance of failure in backward chaining. Furthermore, those propositions are new in the proof table unless the same proposition has been used in different subgoalings in the past. In other words, the students must enter new propositions whose truth value is not known. Forward chaining, on the other hand, always enter propositions that are derived from known facts (i.e., justified propositions) in the proof table. Backward chaining differs mostly from forward chaining in this guess-and-try fashion in entering proof statements.

The booklet for the BC students (see APPENDIX C) only showed the entire structure of a proof table. It did not tell students any procedures (i.e., the inference steps) to complete a proof table. The BC tutor taught details on how to assert premises, but it did not emphasize that the premises are unjustified assumptions. Explicating guess-and-try fashion in backward chaining might be a key issue for designing an effective tutor.

6.3. Implication of the Cognitive Load Theory

As mentioned in Section 2.4, backward chaining is apparently more challenging for students to learn than forward chaining. One possible theoretical account for this issue is the cognitive load theory advocated by Sweller (1988), which basically predicts that the more working memory load the students have, the more cognitive capacities required hence the learning would be hindered due to a deficit of the cognitive capacity. Compared to forward chaining, backward chaining may require storing more information in working memory, which raises cognitive load, resulting in less learning.

Since the students followed the exact sequence of inference steps described in Section 3.3, we can estimate the working memory load as a number of the piece of information that must be stored in students' mind during problem solving. When we examined the cognitive model shown in APPENDIX L, it turned out that the cognitive load is rather small for both forward chaining and backward chaining. This is because AGT provides a proof table and the way it is filled is straightforward (i.e., a line at a time from top to bottom). Indeed, the only things that the FC students must store in their working memory is (1) the mapping for a postulate configuration being overlapped onto the problem figure, and (2) the conditional description of a postulate, i.e., the consequence and the premises of a postulate with labels used in the Postulate Browser window. The BC students also had to store these two things in their working memory; the mapping and the conditional description. In addition, the BC students needed to store in working memory a set of premises of the postulate, because the premises must be tested for duplication before they were asserted to the proof table. However, since the test for duplication took place immediately after the premises were stored in working memory, cognitive load during the training for BC students should be fairly small.

Based on the above observation, cognitive load theory predicts that FC and BC students should have the same learning gains hence should have shown the same performance on the post-test. Still our data showed that the FC students outperformed BC students. Thus, it might be either that (1) cognitive load theory does not explain why the FC students learned more than the BC students, or that (2) the BC students suffered from other kind of cognitive load that was not captured in our cognitive model of proof writing.

6.4. Complexity of the Search

That both FC and BC students seldom assert off-path proof statements was a surprising finding.

BC students could have created more unsuccessful subgoaling (both off-path and wrong), resulting in having irrelevant propositions in a proof table. If this had taken place, then the students would have had more difficulties in keeping their proofs straight, because now they needed to keep track of which postulates they must cross out when they hit an impasse and back up. Indeed an average branching factor in backward chaining measured by running GRAMY backwards on 11 training problems is 5.47 (SD=2.07). Although this could account for the difficulty in backward chaining, unsuccessful (both false and reasonable) subgoaling statements were not observed in this study.

Similarly, FC students could have asserted many off-path proof statements, because they are equally plausibly the constituents of the proof at the time they were asserted. We calculated average number of possible assertions for forward chaining by running GRAMY on each of the 11 problems used in the training sessions. For each proof statement in each of the proofs, we counted the number of correct assertions that could have made. The average of those counts is 20.84 (SD=10.11). The chance for the FC students to make off-path statements was indeed very high. And yet they made few such statements.

That both FC and BC students did not make large number of unsuccessful constructions weakens a claim that the students must learn backward chaining because it reduces the search for appropriate constructions. This is a particularly important contribution of the current study. Indeed, the FC students wrote more on-path statements for construction than BC students (Table 5.3). Furthermore, there was no instance of off-path constructions, a postulate application with

construction that is “reasonable,” but not a part of correct proof. This is a clear contradiction to our prediction that forward chaining could produce enormous amount of off-path constructions.

The students used a pencil on the post-test. Thus they could erase off-path statements, which could have lowered the count of off-path statements. Unfortunately, we could not test this hypothesis. However, we were able to count the number of proof statements that were apparently erased in correct proofs. More precisely, we counted proof statements that once had all three constituents (i.e., a proposition, a justification, and premises), but were erased. There are five erased proof statements written by the BC students, and 15 by the FC students. Because it is hard to read the contents of the erased statements, we can not really tell much about these erased proof statements, but that there were only 20 out of 479 proof statements erased is a surprising observation.

Since the major focus of this study is on figuring out what makes backward chaining difficult to learn, we have not fully analyzed successful proofs. However, the reason of having few off-path search can be accounted by the theory of analogical search control advocated by VanLehn (1998). The theory predicts that once the students acquired a search control from problem-solving experience in the past, they can effectively retrieve such search control to solve new problems. Hence if the questions in the post-test were similar to the training problems, which is true in the current study, the students might have utilized episodic memories to select only on-path postulates.

Another account for not observing off-path statement is that the students in both conditions might have applied a bidirectional search; they applied some mental BC or some mental FC in order to find a connection between the givens and the goal. Unfortunately, we do not have any

factors in our data to test this hypothesis. We will discuss a possible extension of the experiment in Section 7.2.

6.5. Concluding Remarks

As mentioned in Section 2.4, most of the previous studies comparing FC and BC failed to show the main effect for the tutor condition. The current study showed that the FC students outperformed to the BC students in proof-writing, especially proofs with construction. We have not yet to know why our study has met such a difference while others did not, but we consider two potential explanations.

The effect of modeling: AGT not only have a detailed cognitive model of proof-writing, but also provides modeling prior to students' own problem solving. This modeling might not be as effective for BC as FC. Especially, the last steps of the BC model, i.e., asserting premises and line numbers, could have not been confusing hence not comprehensible for students.

A potential preference on forward chaining: As discussed in Section 5.1, the students apparently knew forward chaining prior to the experiment. The familiarity with the strategy may be the reason that the FC students made fewer errors on the post-test. Recruiting students who have no experience in geometry theorem proving for the exact same experiment conducted in the current study could test this hypothesis.

The difficulty in subgoaling and the effect of modeling are not specific only in geometry theorem proving. Thus, we conjecture that a similar result (i.e., forward chaining leads to better learning) would be obtained in other domains where one can describe domain principle as a set of production rules and the students learn how to apply them either forwards and/or backwards. Since many studies have indeed showed that the novices prefer forward chaining, the more carefully designed studies should be conducted to examine to what extent the each of the

problem solving strategies facilitates students' learning. The subskills that are part of the target cognitive skills (e.g., construction in the current study) must also be carefully designed and examined.

7. Future Work

Based on the lessons learned through this study, this section discusses implications to enhance effectiveness and efficiency of the tutor, and some open questions that would be of interest for future studies.

7.1. Implications for a Tutor Design

A potential way to improve the BC tutor's efficacy is to intensify modeling and scaffolding on subgoaling for backward chaining. Although asserting unjustified propositions into a proof step was explicitly stated in the cognitive model of backward chaining utilized in AGT, the model was not effective in supporting the BC students in learning subgoaling. The Postulate browser allowed the students to read a description of a postulate that contains its premises and a consequence. The cognitive model of backward chaining also captured an inference step to convert those descriptions into a conditional form (i.e., an IF-THEN form) that explicitly holds the premises in the condition part (the IF part). Unfortunately, those aids were not effective enough to enhance students' ability to deal with subgoaling.

The inadequacy of the BC tutor may also be due to a lack of instruction on *backtracking*. Backward chaining is essentially nondeterministic. For some goals, there are multiple equally plausible postulates whose consequences unify with the goal. Therefore, one must *choose* one of the postulates, try it, and if it does not work well, *back-up* to the choice point and choose another postulate. AGT acted as a more restricted tutor. Instead of allow students to choose a postulate

and possibly backup to this choice later, the tutor only allows them to choose an on-path postulate, so they never had to back up during training. This design principle is supported by Anderson *et al.* (1995)'s observation that the more the students flounder, the less opportunity they have for each cognitive skill to be exposed hence they achieve less learning. For subgoaling, however, it might be necessary for students to understand that they are asserting hypotheses that could be wrong. Moreover, when applying backward chaining during the post-test, students may have to choose among equally plausible postulates. This could cause confusion and consternation. Thus, it might be necessary to let students backtrack during training.

A related issue is to teach students to recover when they get stuck. Since the backward chaining strategy may lead them to an impasse, they should be taught what to do when they get stuck. AGT did not do this. Perhaps that is why the BC students often got stuck during the post-tests (see Figure 5.12). AGT should train an ability to analyze the situation to identify an impasse, to diagnose the cause of the impasse, and to figure out an alternative way to avoid it by selecting a different path.

7.2. Research Questions for Future Studies

Probably the most interesting inquiry to investigate is about the students' ability to plan for a proof. That the students seldom asserted off-path proof statements in correct proofs may be evidence that they had a plan in their mind and rejected those off-path statements during the planning phase. Koedinger and Anderson (1990) advocated a theory of abstract planning that predicts that the experts plan in the so-called *abstract space* hence needed less search effort than planning in the *execution space* (as the novices do) where the unit of inference corresponds to proof statements and algebraic operations. The students in the current study may have applied

abstract planning or other type of planning strategy in their mind during the post-test in our study. In any case, probing students' reasoning during problem solving via verbal protocols or other methods may reveal why they wrote so few off-path proof statements.

Since subgoaling is the key issue for learning backward inference for geometry theorem proving, we could design a tutoring system that emphasizes the nature of guess-and-try when applying a postulate. In other words, backing up must be explicitly taught. An investigation is needed on whether explicitly teaching backup techniques facilitates learning backward chaining.

Is backing up really a challenging issue for students learning backward problem solving? To answer this question, one can observe a correlation between the average branching factor of proof and measures of performance. For instance, if backing up actually matters, then the bigger the branching factor, the more off-path statements the students should make, or the more likely the students get stuck, or the longer they should take to assert the proof statement. If the post-test had more problems with a large variety of branching factors but similar complexity otherwise, then we could test this hypothesis.

REFERENCE

- Algarabel, S., & Dasi, C. (1996). Heuristics and memory strategies used by mathematicians. *Perceptual and Motor Skills*, 83(1), 41-42.
- Anderson, J. R., Bellezza, F. S., & Boyle, C. F. (1993). The Geometry Tutor and Skill Acquisition. In J. R. Anderson (Ed.), *Rules of the mind* (pp. 165-181). Hillsdale, NJ: Erlbaum.
- Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42(1), 7-49.
- Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2), 167-207.
- Chaiyasang, S. (1989). An Investigation into Level of Geometric Thinking and Ability to Construct Proof of Students in Thailand. *Dissertation Abstracts International*, 49A(8), 2137.
- Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13(2), 145-182.
- Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5, 121-152.
- Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Erlbaum.
- Dreyfus, T. (1999). Why Johnny Can't Prove (with Apologies to Morris Kline). *Educational Studies in Mathematics*, 38(1), 85-109.
- Dreyfus, T., & Hadas, N. (1987). Euclid May Stay -- and Even Be Taught. In M. M. Lindquist & A. P. Shulte (Eds.), *Learning and Teaching Geometry, K-12. 1987 Yearbook* (pp. 47-58). Reston, VA: NCTM.
- Elio, R., & Scharf, P. B. (1990). Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science*, 14(4), 579-639.
- 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* (pp. 83-111). Hillsdale, NJ: Erlbaum.
- Kalyuga, S., Chandler, P., Tuovinen, J., & Sweller, J. (2001). When problem solving is superior to studying worked examples. *Journal of Educational Psychology*, 93(3), 579-588.

- Koedinger, K. R. (1990). *Theoretical and Empirical Motivation for the design of ANGLE: A new geometry learning environment*. Pittsburgh: Carnegie Mellon University.
- Koedinger, K. R. (1991). *Tutoring concepts, percepts, and rules in geometry problem-solving*. Carnegie Mellon University, Pittsburgh, PA.
- Koedinger, K. R., & Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, *14*(4), 511-550.
- Koedinger, K. R., & Anderson, J. R. (1993). Reifying implicit planning in geometry: Guidelines for model-based intelligent tutoring system design. In S. P. Lajoie & S. J. Derry (Eds.), *Computers as cognitive tools* (pp. 15-45). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Laborde, C. (1990). Language and mathematics. In P. Nesher & J. Kilpatrick (Eds.), *Mathematics and cognition: A research synthesis by the International Group for the Psychology of Mathematics Education* (pp. 53-69). New York, NY: Cambridge University Press.
- Landa, L. N. (1975). Some problems in algorithmization and heuristics in instruction. *Instructional Science*, *4*(2), 99-112.
- Larkin, J., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and Novice Performance in Solving Physics Problems. *Science*, *208*(4450), 1335-1342.
- Lovell, K. (1971). The Development of the Concept of Mathematical Proof in Able Pupils. In M. F. Roskopf & L. P. Stefee (Eds.), *PIAGETIAN: Cognitive-Development Research and Mathematical Education*: NCTM.
- Matsuda, N., & VanLehn, K. (2004). GRAMY: A Geometry Theorem Prover Capable of Construction. *Journal of Automated Reasoning*, *32*(1), 3-33.
- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.
- Polya, G. (1957). *How to Solve It* (2nd ed.). Princeton, NJ: Princeton University Press.
- Priest, A. G., & Lindsay, R. O. (1992). New light on novice-expert differences in physics problem solving. *British Journal of Psychology*, *83*(3), 389-405.
- Reiser, B. J., Copen, W. A., Ranney, M., Hamid, A., & Kimberg, D. Y. (1994). *Cognitive and motivational consequences of tutoring and discovery learning* (No. Technical Report #54). Evanston, IL: The Institute for the Learning Sciences, Northwestern University.
- Renkl, A., Atkinson, R. K., & Grose, C. S. (2004). How Fading Worked Solution Steps Works ? A Cognitive Load Perspective. *Instructional Science*, *32*(1-2), 59-82.

- Renkl, A., Atkinson, R. K., Maier, U. H., & Staley, R. (2002). From example study to problem solving: Smooth transitions help learning. *Journal of Experimental Education*, 70(4), 293-315.
- Renkl, A., Stark, R., Gruber, H., & Mandl, H. (1998). Learning from Worked-Out Examples: The effects of example variability and elicited self-explanations. *Contemporary Educational Psychology*, 23(1), 90-108.
- Renner, J. W., & Stafford, D. G. (1976). The Operational Levels of Secondary School Students. In J. W. Renner (Ed.), *Research, Teaching, and Learning with the Piaget Model* (pp. 91-109). Norman, OK: University of Oklahoma Press.
- Scheines, R., & Sieg, W. (1994). Computer Environments for Proof Construction. *Interactive Learning Environments*, 4(2), 159-169.
- Schoenfeld, A. H. (1985). *Mathematical Problem Solving*. Orlando, FL: Academic Press.
- Schoenfeld, A. H. (1988). When good teaching leads to bad results: The disasters of "well-taught" mathematics courses. *Educational Psychologist*, 23(2), 145-166.
- Senk, S. L. (1985). How well do students write geometry proofs? *Mathematics Teacher*, 78(6), 448-456.
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257-285.
- Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1), 59-89.
- Trafton, J. G., & Reiser, B. J. (1991). Providing natural representations to facilitate novices' understanding in a new domain: Forward and backward reasoning in programming. *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, 923-927.
- Tubridy, A. F., Jr. (1992). *An Instructional Strategy to Enhance Proof-Writing Ability in Secondary School Geometry*. University of Texas at Arlington.
- VanLehn, K. (1998). Analogy events: How examples are used during problem solving. *Cognitive Science*, 22(3), 347-388.
- VanLehn, K., Jones, R. M., & Chi, M. T. (1992). A model of the self-explanation effect. *Journal of the Learning Sciences*, 2(1), 1-59.
- Zhu, X., & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction*, 4(3), 137-166.