An Experimental Comparison of Alternative Proof Construction Environments

by

Richard Scheines and Wilfried Sieg

August 1993

Report CMU-PHIL-40

Philosophy
Methodology
Logic

Pittsburgh, Pennsylvania 15213-3890

An Experimental Comparison of Alternative Proof Construction Environments[1]

Richard Scheines
Wilfried Sieg

Department of Philosophy
Carngie Mellon University
Pittsburgh, PA 15213
email: RS2L@andrew.cmu.edu
WS15@andrew.cmu.edu

Department of Philosophy Technical Report CMU-Phil-40
August 1993

[1] We thank Virginia Hoage, who collected the data and miraculously put it into a form we could use.

Abstract

In this paper we compare computerized environments in which students complete proof construction exercises in formal logic. After being given a pretest for logical aptitude, three matched groups were presented identical course material on logic for approximately five weeks by a computer. During the treatment, all students were required to complete several hundred proof construction exercises. The three groups did the exercises and the midterm in different environments. The group with a more sophisticated interface performed better on the midterm. Nearly all the difference in performance showed up in the harder problems. In a follow up experiment in which flexible strategic problem solving help was added to the environment, performance improved slightly, but the data are inconclusive.

1. Introduction

Formal logic is our best theory of rigorous reasoning. Because formal logic and computation are so closely related, computer aids to logic teaching and theorem proving have proliferated. At present there are over 40 computer programs that help teach logic, and surely more will follow [Croy 86, Burkholder 89]. We have spent the last few years developing the Carnegie Mellon Proof Tutor (CPT), a computerized proof construction environment that combines a sophisticated graphical interface with a system for strategic help based on a complete and cognitively natural algorithm for finding natural deduction proofs in propositional logic [Sieg 91, Pressler 88].

CPT was built on the educational belief that students learn more from doing the problems they are assigned when their problem solving environment has three features. First, its interface must relieve the student of all non-essential cognitive load. That is, it must take over routine calculations, display the problem state informatively and reliably, and identify errors instantly.[2] Second, it must allow students to traverse the problem space in any way they wish. Third, it must provide intelligent and flexible *strategic guidance.*[3] Although students quickly master the syntactical constraints on logic proofs, it may take them months to internalize the strategic subtleties of proof construction.

In this study we report on an experiment that tested the pedagogical value of three features of CPT's problem solving environment. They are:

1) an informative and manipulable display of the problem state,
2) the flexibility to work forwards and backwards instead of either way alone, and
3) strategic help at any point in any problem.

We gathered data relevant to the first two features in the fall of 1989 and the third in the spring of 1990. The fall data seem to support the utility of the first feature and strongly support the utility of the second. The spring data suggest that strategic help is important, but for several reasons cannot be considered conclusive.

2. Proof Construction

In the preponderance of problems assigned in a class on formal logic, students are asked to produce a proof of a particular conclusion from given assumptions (premises). A proof is a series of lines, each of which is either an assumption or the result of applying a logical "inference rule" to previous lines. Like most problem solving tasks, one begins a proof construction problem with a gap between what one is given (the premises) and what one seeks (the conclusion).[4] The gap can be closed by inferring

[2] John Anderson [Anderson 85a] has shown that immediate error correction reduces the frequency of future errors. Jill Larkin [Larkin 87] has shown that an informative and easy to parse display of the problem state improves performance and reduces training time.

[3] This point motivates most "intelligent" computer tutors, e.g. [Anderson 85b].

[4] See [Newell 72].

new lines from the premises, or by creating subgoals such that, if proved, would allow one to infer a line desired (Figure 1).

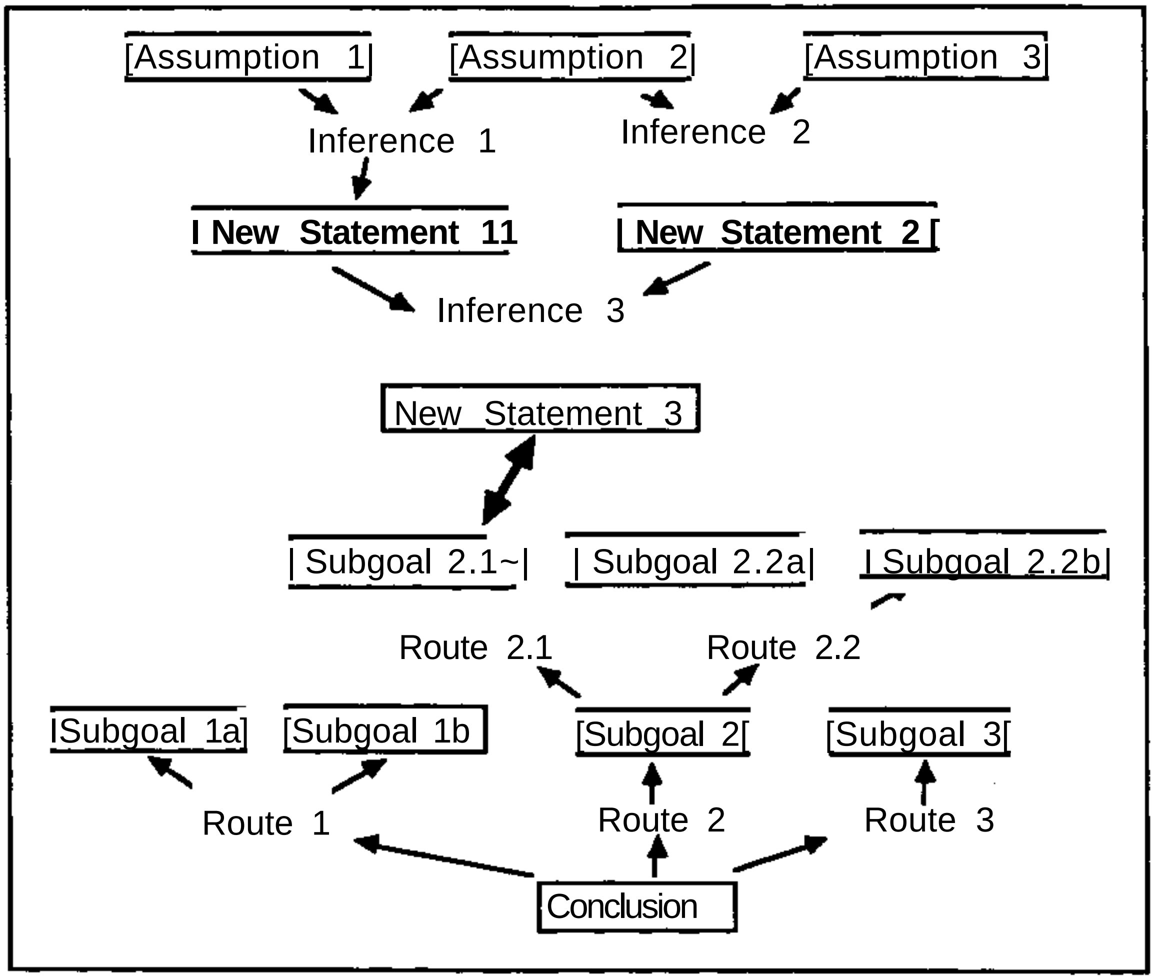


Figure 1

<u>Search</u>

Although it is almost never taught as such, finding a proof is a search problem. One can work forwards and search for a goal G in the forest of lines that emanate from a set of assumptions **r** (Figure 2),

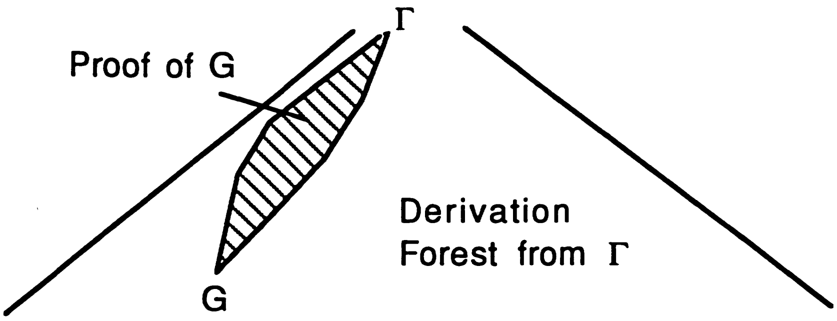


Figure 2

or work backwards and search for Γ in the forests that *lead to* G (Figure 3),

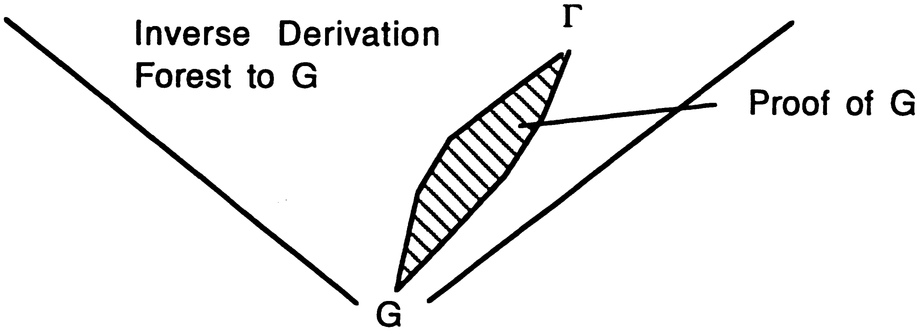


Figure 3

or one can confine the search to the region where the forward and backward searches overlap (Figure 4).

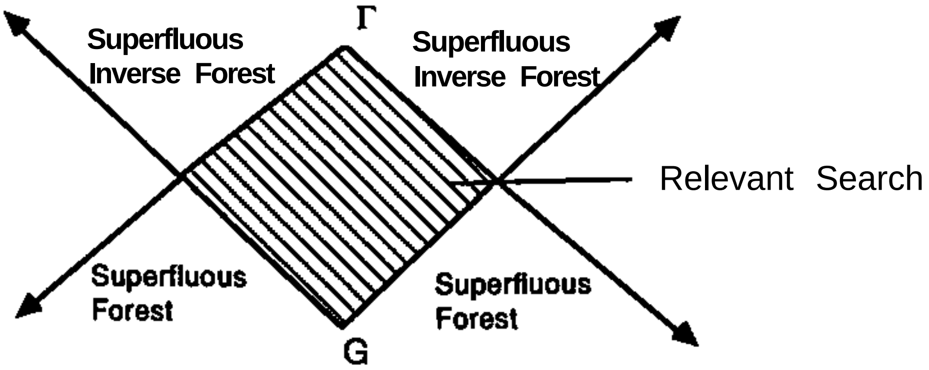


Figure 4

All the standard problems of search arise in proof construction. Students wander down dead ends and cannot recover, they travel in circles, and they move laterally instead of vertically, i.e., they make moves which are legal but which bring them no closer to a solution.

If they are explicitly taught to see proof construction as a search problem, and if they are given a problem solving environment in which the structure of their search is made clear, then students should learn proof construction more easily and see its connection to other problem solving contexts as well.

Temporary Assumptions

Proof construction is of course more than just search. It involves pattern matching, strategic thinking, and a skill that is somewhat special to logic: introducing and discharging temporary assumptions. Taught traditionally, this skill is particularly hard for many students to master.

In certain proof construction contexts one is allowed to introduce a temporary assumption which can then be used in a corresponding sub-proof. Once the sub-proof is finished, the assumption must be cancelled. For example, suppose we are told that a given number is either even, or divisible by five but greater than ten, and we are asked to prove it is not a prime. We do so by first assuming that it is even and under this assumption showing it cannot be prime, and then assuming it is divisible by five and greater than ten and under that assumption proving that it is not prime. In the subproof in which we assume the number is divisible by five and greater than ten, we cannot also assume that it is even. Assumptions can be nested in complicated ways, and students often get lost tracking them.[5]

3. Standard Proof Checkers

The great majority of computerized proof construction environments are simple "proof checkers" in which only mistake correction is standard [Pressler 88]. These programs present the premises in a column on the top of the screen and allow the student to add to this column by entering steps sequentially until the conclusion has been proved (Figure 5).

1	P -> (Q & R)	Premise
2	S -> (Q & T)	Premise
3	P	wp
4	Q & R	1,3 ->_Elim
5	Q	4, &_Elim_l
6	P -> Q	3,5 ->_Intro
7	S	wp
	▬	

Figure 5

Most also calculate the conclusion of an inference and automatically justify it appropriately, so routine calculational relief is also quite common. Some allow working backwards as well as forwards, but most do not.[6] Since working backwards is easy on pencil and paper, and a technique that is encouraged in almost all logic classes, those environments that do not allow it are in some sense an impediment to good problem solving.

Very few logic programs give the student an informative display of the problem state.[7] A column of lines can represent a finished proof adequately, but not the *search* for a proof. One is in essence collapsing a

[5] Before we used CPT in our Introductory Logic class at Carnegie Mellon, perhaps 50% of our students encountered serious trouble with logical dependency.

[6] "Symlog", by the Portararos is an example [Portoraro 87].

[7] This is beginning to change. For example, see [Weston 87].

forest into a line, and so relevant structure and information are necessarily lost. A column of lines also poorly represents the structure of temporary assumptions.

Finally, standard proof checking programs give no strategic help. This is like a chess tutor that checks each player's moves to make sure they are legal, but can not itself play or give advice on where to move.

4. The Carnegie Mellon Proof Tutor (CPT)

CPT provides all the standard proof checking operations, i.e. mistake correction and calculational relief, but it also provides an informative display, the ability to work backwards, and strategic help.

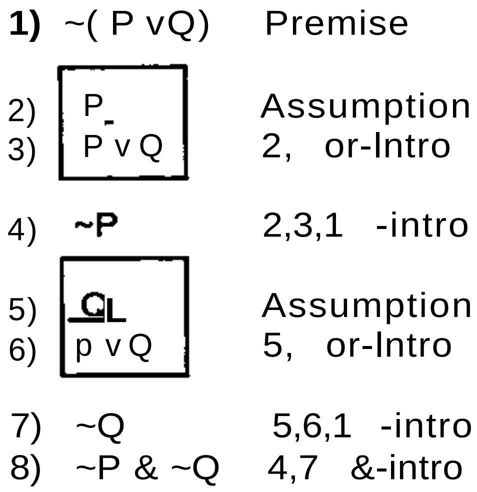
CPT's display is novel in that it combines information about the search along with information about the structure of temporary assumptions. In the single column of lines used by standard proof checkers, associated with each line ψ is a set of lines on which ψ depends. Thus when a temporary assumption θ is introduced, any line that depends on θ includes θ's line number in its dependency list. The Fitch diagram presents a superior and graphical option for displaying logical dependency.[8] In a Fitch diagram a box is drawn for each new temporary assumption added. Everything inside this box depends on this assumption. When the assumption is dishcarged, the box is closed. Thus calculating logical dependency from a Fitch diagram is as cognitively natural and easy as deciding box containment. Consider the alternative displays of the same proof in Figure 6.

[8] Many standard logic texts use a Fitch style representation. See [Kalish 80], for example.

Conventional

1)	~(P v Q)	{1}	Premise
2)	P	{2}	Assumption
3)	PvQ	{2}	2, or-Intro
4)	~P	{1}	2,3,1 -intro
5)	Q	{5}	Assumption
6)	PvQ	{5}	5, or-Intro
7)	~Q	{1}	5,6,1 -intro
8)	-P&-Q	{1}	4,7 &-intro

Fitch - Diagram

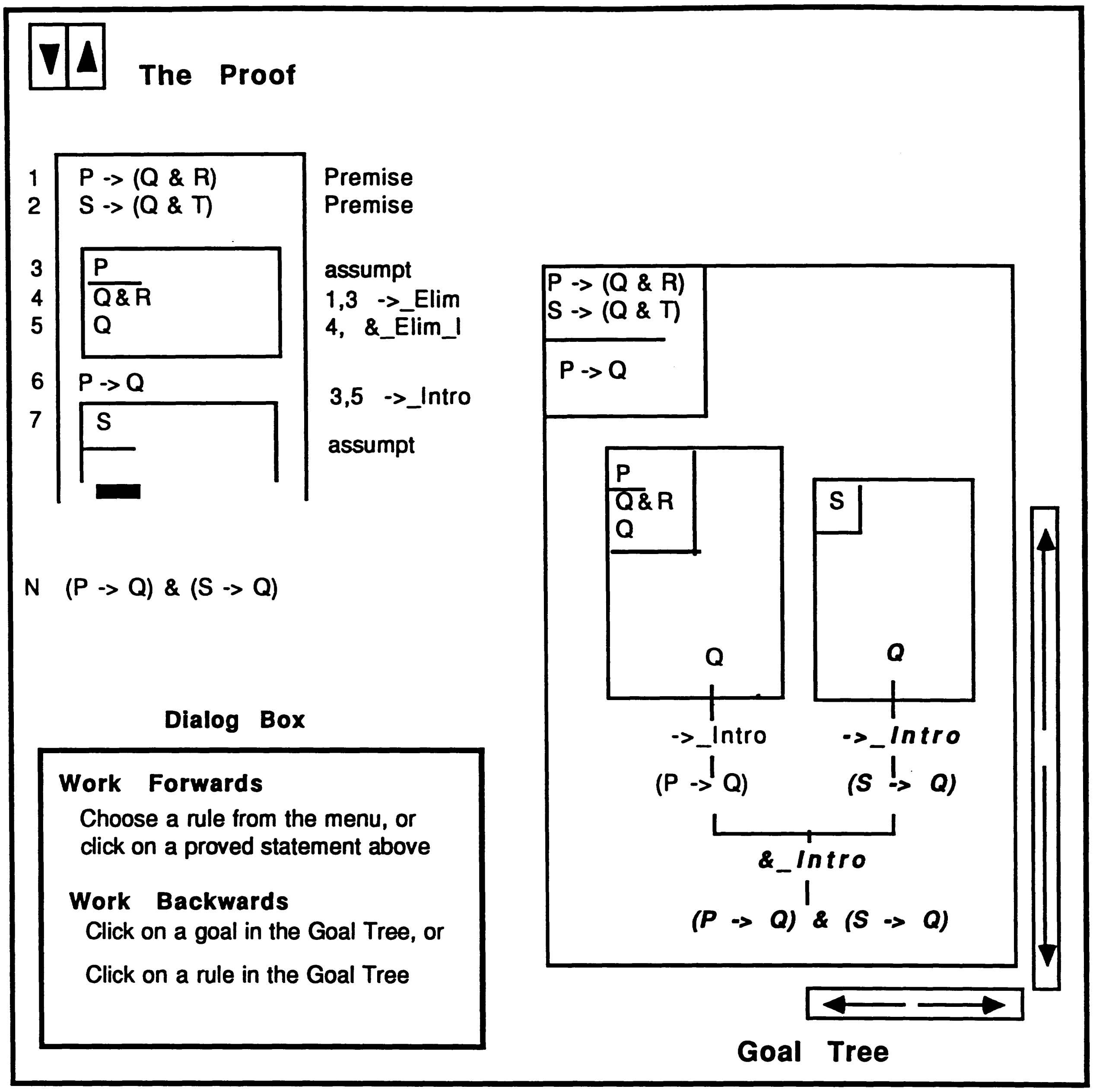


1)	~(P vQ)	Premise
2)	P	Assumption
3)	P v Q	2, or-Intro
4)	~P	2,3,1 -intro
5)	QL	Assumption
6)	p vQ	5, or-Intro
7)	~Q	5,6,1 -intro
8)	~P & ~Q	4,7 &-intro

Figure 6

Glancing at the conventional diagram, it might appear strange that lines 3 and 6 contain the same formula. It might also appear that we have derived several contradictions, e.g. lines 5 and 7. Neither oddity seems problematic in the Fitch diagram. One can see immediately that lines 3 and 6 occur within different boxes and thus depend on different assumptions. One can also see that line 5 is an assumption and line 7 is the result of an argument by *reductio* (-introduction) from this assumption.

A Fitch diagram does not represent the search for a proof any better than a column of lines, however. In CPT we have combined a tree-like diagram of subgoals with a Fitch style diagram to give the student a "Goal Tree. " Since the Fitch diagram is standard and used in many texts, we provide it in parallel to the Goal Tree as a representation of "The proof" (Figure 7).



The Proof

```
1   P -> (Q & R)        Premise
2   S -> (Q & T)        Premise

3   P                   assumpt
4   Q&R                 1,3  ->_Elim
5   Q                   4,  &_Elim_l

6   P->Q                3,5  ->_Intro
7   S                   assumpt
```

N (P -> Q) & (S -> Q)

Dialog Box

Work Forwards
Choose a rule from the menu, or click on a proved statement above

Work Backwards
Click on a goal in the Goal Tree, or

Click on a rule in the Goal Tree

Figure 7

Students can work forwards or backwards in CPT, and they can in a glance determine the state of their search and the logical structure of their temporary assumptions.

Students can also request strategic help at any point in any problem. CPT solves the problem from the point the student has left it, and then gives the student one of three kinds of advice. It will either 1) display its solution and the search it conducted for that solution, or 2) allow the student to step through its search for the solution, or 3) conduct a dialogue with the student on how to proceed towards a solution.

Having designed such a system and tested it informally on dozens of students, our goal was to test it experimentally. In particular, we wanted to know if the goal-tree display, the ability to work backwards and forwards, and the strategic help objectively improved performance.

4. Methods

CPT is currently just a proof construction environment, not a logic teacher. Before students can use it they need to learn some logic. Were any person to teach different groups this logic, they would inevitably bias the treatment. Fortunately, there is an entire course on logic taught *entirely* by computer. The VALID program, which was developed at the Institute for Mathematical Studies in the Social Sciences (IMSSS) at Stanford University in the late 1960s and early 1970s [Suppes 81], thoughtfully covers an extensive introductory curriculum, including a system of propositional logic which is an extended system of natural deduction. VALID is now used at perhaps half a dozen universities around the country, including Carnegie Mellon University (CMU).

VALID alternates between introducing new logical ideas and assigning exercises to build the skills that correspond to these ideas. After a new idea is introduced via several screens of text, students are presented with a menu of proof construction exercises. Upon selecting an exercise, they are shunted into VALID's interactive proof construction environment to complete the exercise (Figure 8). After finishing an exercise they are returned to the menu, or, if the menu is completed, to the main curriculum. The course is entirely self-paced, i.e., there is no time limit within which a problem must be completed. There is also no penalty for mistakes: students simply keep trying until they solve the required number of problems and then move on to the next concept. The great bulk of a student's time is spent actively working on exercises, not passively reading text.

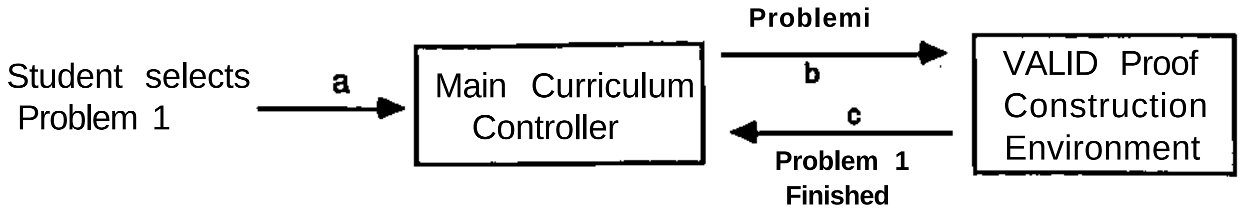

Student selects Problem 1
a
Main Curriculum Controller
Problemi
b
c
Problem 1 Finished
VALID Proof Construction Environment

Figure 8

VALID's proof construction environment is a standard proof checker, and part of its curriculum introduces the environment and trains students how to use it.

Every semester, 30-60 students take Introduction to Logic at Carnegie Mellon from VALID. They use VALID's proof construction environment for one brief lesson, and then use CPT instead of Valid's proof construction environment to complete the remainder of the exercises prior to the midterm, and to take the midterm as well.[9] They are exposed to CPT for approximately five weeks. In the fall semester of 1989, we conducted an experiment to compare three different versions of CPT.

On the first day of classes we gave all students a pre-test for logical aptitude designed by the Educational Testing Service and given to us by Stanford. We used the results of this test to split the class into three groups of 11 students, equal in both the means and variances of their pretest scores. Comparing two features, we would have liked to have four groups, one for each boolean combination of features. But since our sample size was small, we settled for three. Each group proceeded through the VALID curriculum, but used a different proof construction environment (pee) to do its exercises and then the midterm. The groups were trained to use a standard proof construction environment by VALID, and then trained for an extra half hour by a TA to learn their pee.

The first pee (the Forwards group) was a simulation of a standard proof construction environment (Figure 5). Students using it could only work forwards and were given the standard column representation of proofs, i.e., they had neither a Fitch diagram nor a Goal Tree. The Backwards group used a degraded version of CPT in which they could *only*

[9] **They receive no strategic help while completing the midterm, of course.**

work backwards from the conclusion toward the premises by subgoaling in the Goal Tree. The third pee provided students with the full CPT interface, which includes a Fitch diagram, a Goal Tree, and the ability to work forwards or backwards (Figure 7). *No group in the fall had any strategic help.*

The midterm exam consisted of eight proof construction problems similar to those the students faced prior to the midterm. Two problems were easy, three medium, and three hard (Figure 9).

Midterm Problems

Easy

From:
Derive: (P -> S) -> ((S -> R) -> (P -> R))

From:
Derive: ((P -> R) & (Q -> S)) -> ((P & Q) -> (R & S))

Medium

From:
Derive: **((P** & Q) -> **(R** v S)) -> ((P -> R) v (Q -> S))

From:
Derive: ((P -> Q) -> P) -> P

From: (R <-> (P & Q)) & P
~S -> ((Q <-> P) & ~R)
Derive: S

Hard

From:
Derive: ((P -> S) & ((~P & Q) -> S)) <-> ((P v Q) -> S)

From: ~(P & Q) & (P v Q)
(~P & Q) -> S
(P & ~Q) -> S
Derive: S

From: (P v ~Q) -> (R v ~S)
(PvT)
(T v ~P) -> (~S v M)
~(R v M)
Derive: ~S

Figure 9

Students were given three hours to complete the test. Besides recording whether they succeeded or failed to complete a given problem, we measured how much time it took to finish a problem, how many steps were in the finished proof, and how many errors of logical dependency students made. To be fair to the students, we adjusted their grades by an amount equal to the difference between their group's mean and the best group's mean.

We collected data on the next semester's class as well. In the spring of 1990 had 35 students and we did not split them into groups. Each student had the full CPT interface *plus our first implementation of strategic help*. They used an identical curriculum, but were given a substantially harder midterm. The first eight problems on their midterm were identical to the fall's (up to notational variations), but it contained two extra problems that were harder than any given to the fall class.[10] Instead of three hours to take the test, we were forced to give this group only two and a half. We analyzed the results for all four groups, plus two amalgams. The first amalgam, 1-way, consists of the first two groups in the fall, i.e., the Forwards group and the Backwards group. These groups could only work in one direction. The second amalgam, 2-way, consists of the CPT group in the fall plus the entire spring class.

Figure 10 shows the pretest means, in percents. The Backwards group is slightly high because 3 of the poorer students dropped out after we had matched the groups. 1 student dropped out of the Forwards group. Thus the sample sizes for the groups are:

Group	Sample Size
Forwards	10
Backwards	8
CPT	11
Spring	35
1-Way	18
2-Way	46

[10] We added extra problems so we could distinguish between good and really gifted students. We will compare the spring classes performance on these problems against next fall's students.

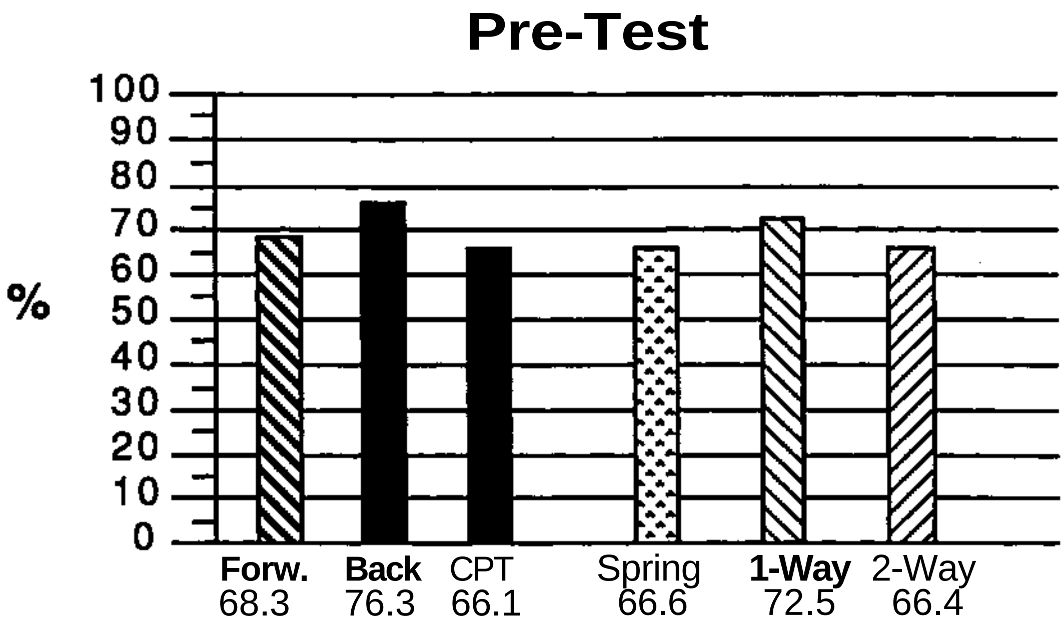


Figure 10

Students in the CPT group, who were allowed to work forwards and/or backwards, chose to work backwards over 46% of the time. They exploited the extra parts of their environment that alternative groups could not. Since each group completed over a hundred proof construction problems over the course of five weeks with their own pee before they took the midterm, no training effects were still operative. By the time students took the midterm, they were solving problems in their pee as well as they were going to.

5. Results

The results for the midterm are striking (Figure 11).

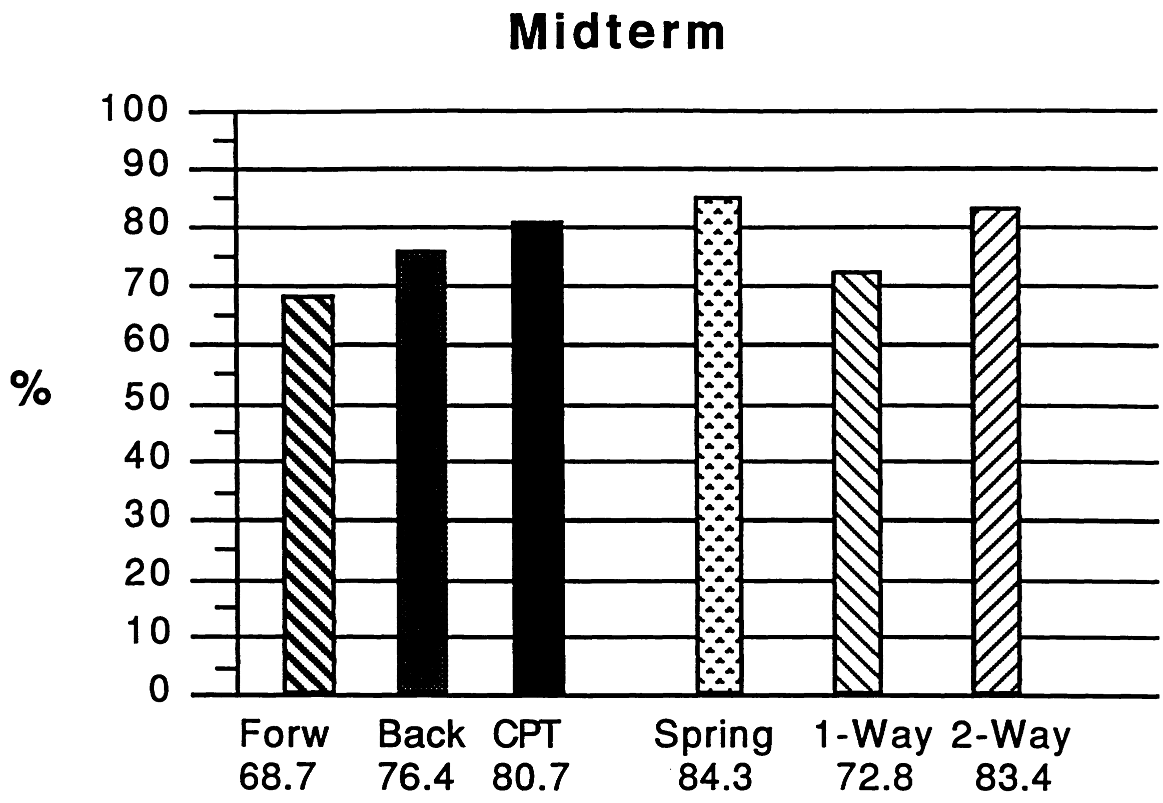


Figure 11

Not surprisingly, most of the difference in the student's performance was in the harder problems (Figure 12).[11]

[11] We say this is not surprising because in our experience the way a subject is taught makes little difference to either the very smart students or to the average student's performance on easy problems. One expects to see the payoff from better educational treatments in harder problems done by average students.

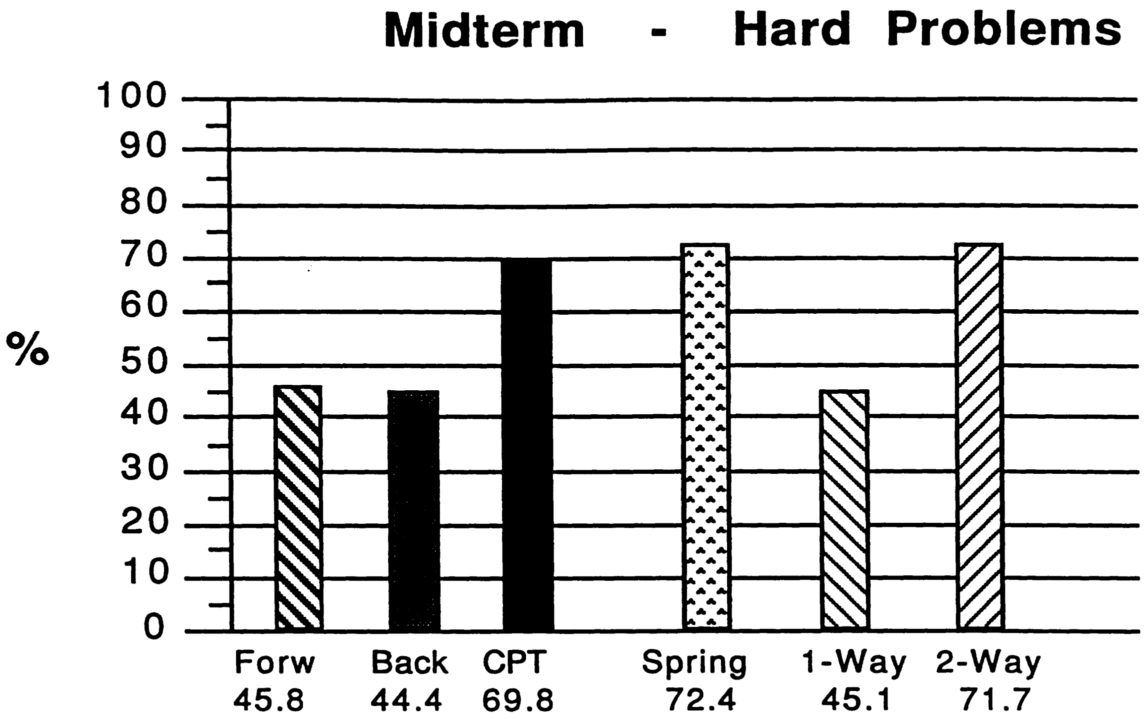


Figure 12

Although at these sample sizes one needs a large difference in means to achieve statistical significance on a T-test, the difference between the 1-way and 2-way performance on the hard problems is significant at the .026 level. That is, if we assume that the two groups were equal in their ability to solve the hard problems,[12] there is a 2.6% chance that we would observe as large a difference in their means as we did or larger. From this we can reject the hypothesis that the groups were equal when they took the midterm. Since the only difference in these groups was the pce each used, we assume the differences in the pces caused the difference in their final proof construction abilities.

The results on the easy (Figure 13) and medium problems are much less dramatic (Figure 14).

[12] Actually it appears that the 1-way group had slightly more ability. Their mean pretest score was 72.5 compared with 66.4 for the 2-way group. This makes the difference in their performance on hard problems even more dramatic.

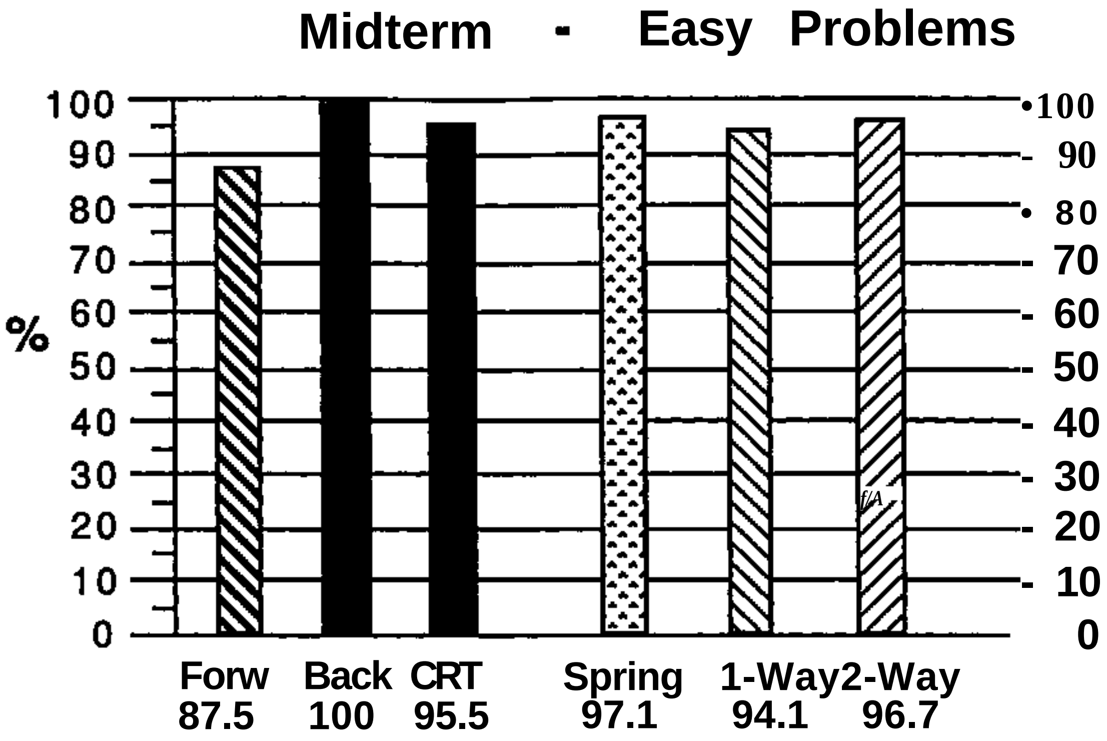


Figure 13

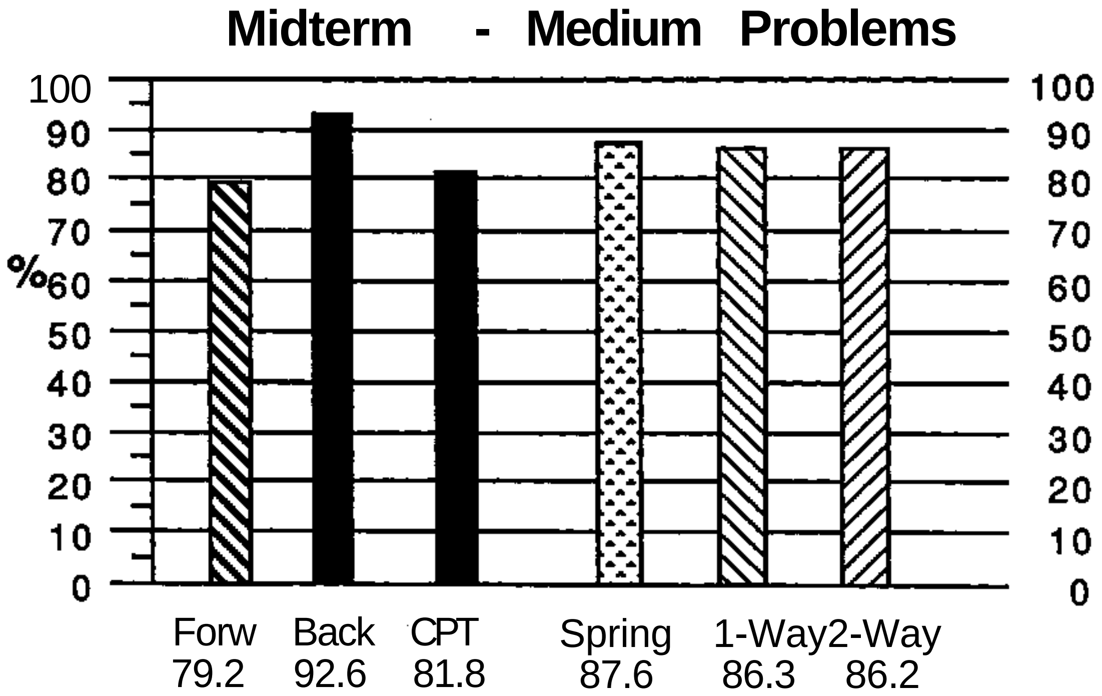


Figure 14

The students who had the better interface constructed more elegant proofs (Figure 15).

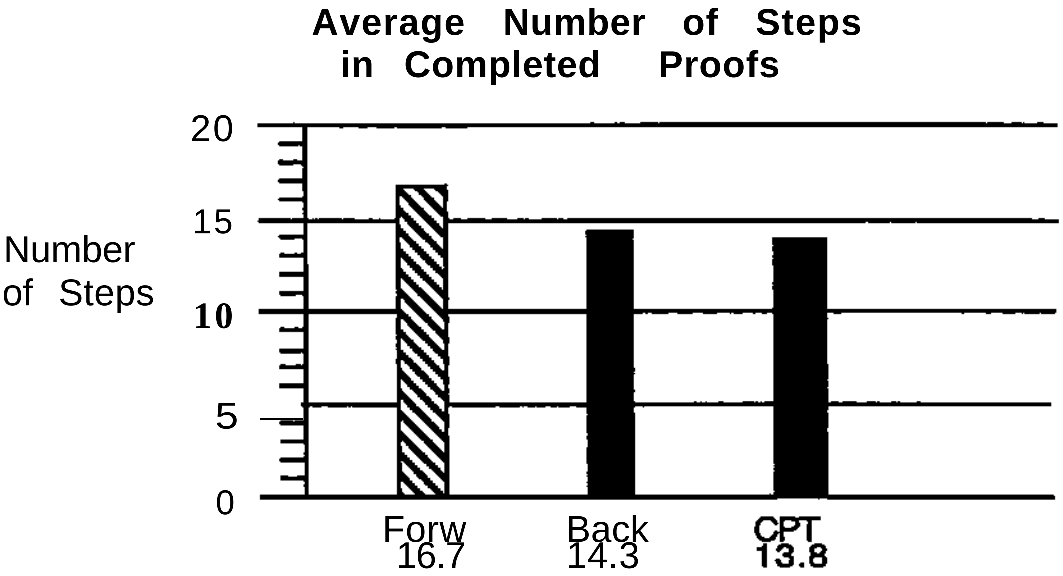


Figure 15

They also took less time to find them (Figure 16).

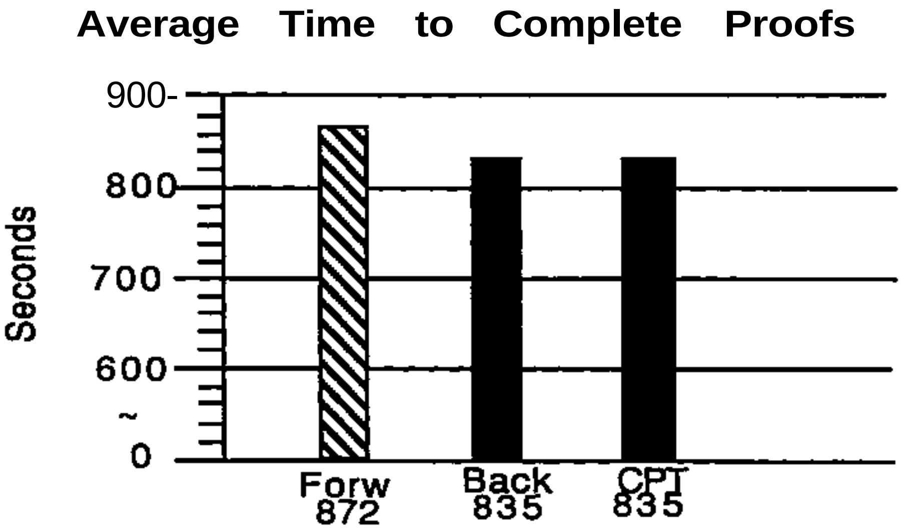


Figure 16

The students in either the Backwards, CPT or Spring group made almost no errors (3) of logical dependency. The Forwards group made 11 such errors. The Backwards group's lack of errors is not surprising, since one cannot make them while working backwards in the Goal Tree.

6. Analysis

Recall that we set out to test three features of CPT's problem solving environment. They are:

1) an informative and manipulate display of the problem state,
2) the flexibility to work forwards and backwards instead of either way alone, and
3) strategic help at any point in any problem.

The data strongly support the utility of the second feature. Although working backwards is clearly helpful in certain contexts of proof construction, it is unnatural in others. The same holds for working forwards. Students forced to work exclusively in either direction were at a severe disadvantage, as is evidenced by the large differences between the 1-way and 2-way groups, as well as between the CPT group and either the Forwards or Backwards group. The CPT group differed from the Backwards group only in the ability to work forwards, so clearly this direction of search mattered. The CPT group differed in two ways from the Forwards group, first in its ability to work backwards and second in the use of an informative display, so the utility of working backwards is more difficult to assess. It seems unlikely that the difference between the CPT group and the Forwards group was entirely due to the display, however. The Forwards group and Backwards group differed in their display, but did not exhibit nearly the difference that the CPT and Forwards group showed. Our interpretation is that only the students in the CPT and Spring groups learned to separate those proof contexts in which working backwards was advantageous from those in which working forwards was better. Thus the ability to take alternative routes through the search space helps the student learn the structure of the space better.

The Goal Tree and Fitch diagram embody the informative display. Every group had such a display except the Forwards group. That it mattered is evidenced first by the difference between the number of errors in logical dependency made by the Forwards and CPT groups. Errors in logical dependency could be made just as well in either group's pee, but only the Forwards group actually committed them. Second, although the difference in midterm success rate between the Forwards and Backwards groups matches their pretest difference almost exactly and is insignificant, the Backwards group outperformed the Forwards group in time on search and in the size of the proof found. This is so even though students spend most of their educational life training to solve problems in a forwards fashion. Neither the differences in times of search or proof lengths was statistically significant, but at sample sizes of 8 and 10 they would have to be alarmingly large to be so. Our interpretation is that the Backwards group searched more efficiently and with fewer errors because of the Goal Tree diagram.

Assessing the effect of strategic help is more difficult and in this paper has to be considered preliminary and inconclusive. First, students in the spring were given a harder midterm, both substantively and in time pressure. We do not know how many percentage points higher they would have scored had their midterm been the same. Second, one of our programmers introduced a bug in the data collection facility, so we have no data on length of proof, search time, or errors of logical dependency. The same.bug prevented us from keeping accurate tabs on which students actually used the help facility. Since many of them rarely used it, we are missing a rather important piece of information. Third, students in this semester used the first implementation of the help facility, which was rough and had some disappointing bugs. Nevertheless, many students reported a great appreciation for the facility, and the differences between the Spring group and the CPT group from the fall are interesting.

In the shared sections of the midterm, the Spring group was approximately 3.5 percentage points better than the CPT group. The same difference holds for the hard problems. These differences are of course statistically insignificant, yet in the context of a much harder test suggest that some students actually benefitted from the strategic help.

In the fall of 1990 we plan to redo this comparison, giving one group CPT without help and one group CPT with help.

If it is difficult to separate the effects of the three features we tested, it is not hard to see how well they work in combination. The Spring group used the full CPT environment while the Forwards group used a standard proof checker. The Spring group averaged almost two points lower on the pre-test than did the Forward group, and they took a harder midterm in less time than did the Forwards group. Yet their average score on the part of the midterm the two groups shared was 15.6% higher. This difference is significant at .119, which is quite acceptable given the small sample sizes.

7. Conclusions

Students who use CPT's proof construction environment seem to learn proof construction better than those who use standard proof checkers. We draw the following conclusions from our study. Problem solving environments that allow and informatively represent a variety of search strategies encourage better problem solving activity. Larkin's belief that the problem display is pedagogically important [Larkin 87] seems right.

Our results suggest but do not prove that intelligent strategic help is an important feature for problem solving environments. Much more work needs to be done on this issue. First, it is not yet clear that strategic tutoring helps performance, and second, it is not clear how to optimally deliver such help.

Whatever the case, more experimental work needs to be done. The computer is ideal as a tool for educational research as well as educational intervention. We need a more objective foundation from which to proceed. Too much of the advice for those studying the literature on educational computing is anecdotal and vague. What students say they like does not always correspond to what helps them learn. Carefully controlled experiments do.

References

Anderson, J.R., "Production Systems, Learning, and Tutoring", in D. Klahr, P. Langley, & R. Neches (eds.), *Production System Models of Learning and Development*, MIT Press, Cambridge, MA, pp. 437-458, 1987.

Anderson, J.R., *Cognitive Psychology and Its Implications*, W. H. Freeman, San Francisco, 1980.

Anderson, J.R., Jeffries, R., "Novice LISP Errors: Undetected Losses of Information from Working Memory", *Human-Computer Interaction*, Vol. 1, 1985: 107-131.

Anderson, J. R., Boyle, C.F., "The Geometry Tutor", *Proceedings of IJCAI-85*, Los Angeles, CA, 1985.

Bledsoe, W. W., and Loveland, D.W., (eds.), "Automated Theorem Proving: After 25 Years," *Contemporary Mathematics*, Vol. 29, Providence, RI, 1983.

Braine, M. D., "On the Relation Between the Natural Logic of Reasoning and Standard Logic," *Psychological Review*, Vol. 85, pp 1-21, 1978

Burkholder, L., "Cumulative New Software", *The Computers and Philosophy Newsletter*, pp. 48-89,Issue 4:1+4:2, July, 1989

Croy, M. J. , "The Current State of Computer-Assisted Instruction for Logic", *Teaching Philosophy* 9, pp. 333-350, 1986.

Garson, J., "Getting Problem-Solving Advice from a Computer", *BYTE*, May 1981: 186-196.

Gentzen, G., "Investigations Into Logical Deduction", in M.E. Szabo (ed), *The Collected Papers of Gerhard Gentzen*, North Holland, Amsterdam, pp. 68-131,1969.

Howson, A.G., Kahane, J.-P., and Pollak, H., "The Popularization of Mathematics", a report by The International Commission on Mathematical Instruction, in *Notices of the American Mathematical Society*, Providence, RI, Vol. 36, No. 1, January 1989.

Jacoby, L. L. , "On Interpreting the Effects of Repetition: Solving a Problem versus Remembering a Solution", *Journal of Verbal Learning and Verbal Behavior,* Vol 17, 649-667, 1978.

Johnson-Laird, P. N., "Models of Deduction", in R. J. Falmagne (ed.), *Reasoning: Representation and Process in Children and Adults,* Erlbaum, Hillsdale, N. J., 1975, pp.7-54.

Kalish, D., Montague, R., and Mar, G., *Logic: Techniques of Formal Reasoning*, Harcourt Brace Jovanovich, Inc., New York, 1980.

Larkin, J., "Display-Based Problem Solving", in *The 21st Carnegie Symposium*, Erlbaum, Hillsdale, N.J., 1987.

Loveland, D.W., *Automated Theorem Proving: A Logical Basis,* North-Holland, Amsterdam, 1978.

Moor, J. and Nelson, J., "Computer-Assisted Instruction in Logic: BERTIE", *Teaching Philosophy*, Vol. 8, No. 4, 1985: 319-323.

Moor, J. and Nelson, J., "BERTIE-II: Personal Computers and Logic", *Teaching Philosophy,* Vol.8, No. 4, 1985: 319-323.

Newell, A., & Simon, H., *Human Problem Solving,* Prentice-Hall, Englewood Cliffs, N.J., 1972.

Osherson, D.N., *Logical Abilities in Children*, (Vols. 2-4), Erlbaum, Hillsdale, N.J. 1974-1976.

Pelletier, J. F., "Seventy-Five Problems for Testing Automatic Theorem Provers", *Journal of Automated Reasoning*, Vol. 2, 1986: 191-216.

Pelletier, J. F., *Completely Non-Clausal, Completely Heuristically Driven Automatic Theorem Proving*, Technical Report TR82-7 Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, August 1982.

Polya, G., *Mathematics and Plausible Reasoning*, Princeton, 1954

Portoraro, A., and Portoraro, F. D., "SYMLOG: Computer Assisted Instruction in Symbolic Logic," *The Computers and Philosophy Newsletter*, Vol. 2, No. 2, pp 31-47, June 1987

Pressler, J., "Computers in Logic Courses", in Ian Lancashire (ed.), *Methodologies in Humanities Computing*, Philadelphia: University of Pennsylvania Press, 1989.

Pressler, J., and Scheines, R., "An Intelligent Natural Deduction Proof Tutor", *Computerised Logic Teaching Bulletin,* Vol. 1, No. 1, March 1988.

RIPS, L.J., "Deduction," in*The Psychology of Human Thought*, R.J.Sternberg and E.E. Smith (eds.), Cambridge University Press, 1989.

Rips, L. J., "Cognitive Processing in Propositional Reasoning", *Psychological Review,* Vol. 90, 1983: 38-71.

Robinson, J. A., " A Machine-Oriented Logic Based on the Resolution Principle", *Journal of the Association for Computing Machinery*, Vol. 12, No. 1, January 1965: 23-41.

Robinson, J. A., *Logic: Form and Function; The Mechanization of Deductive Reasoning*, North-Holland, New York, 1979.

Sieg, W., "Structure and Design: Getting at the Core of Mathematics and Science in Liberal/Professional Education," to appear in:*The Role of Design in Liberal/Professional Education*, 1989.

Sieg, W., and Scheines, R., "Searching for Proofs (In Sentential Logic)," forthcoming in the *Proceedings of the Fourth International Conference of Computers and Philosophy*, 1991.

Steen, LA., "Statement Before the National Science Board", in *Notices of the American Mathematical Society,* Vol. 33, pp. 241-246, 1986

Suppes, P. (ed.), *University-level Computer-assisted Instruction at Stanford: 1968-1980,* IMSSS, Stanford, 1981.

Weston, T., "Tree Diagrams in Computer-Aided Proof Construction", *The Computers and Philosophy Newsletter,* Vol. 2, No. 2, June 1987: 48-55.