Computer Environments for Proof Construction[*]

Richard Scheines

Wilfried Sieg

Department of Philosophy

Carnegie Mellon University

Pittsburgh, PA 15213

email: R.Scheines@andrew.cmu.edu

W.Sieg@andrew.cmu.edu

in:

Interactive Learning Environments

(Elliot Soloway, editor)

[*] We thank Ken Koedinger for many helpful discussions and some needed encouragement. We also thank Virginia Hoage, who collected the data, and Chris Walton, who performed the programming miracles that made this study possible.

Abstract

Does the presentation and use of the search space matter for complex problem solving tasks? We address these questions for the construction of proofs in sentential logic. Using a fully computerized logic course, we isolated crucial features of computer environments and assessed their relative pedagogical effectiveness. After being given a pretest for logical aptitude, students were divided into three matched groups, each of which used a distinct computerized environment to construct proofs. All students were presented with identical course material on sentential logic for approximately five weeks.

Students completed more than one hundred exercises during those five weeks and took a midterm at the end of the period. The group using the *most informative* and *most flexible* interface performed substantially better on the midterm - the difference was particularly striking for hard problems. In two follow-up experiments we added strategic problem solving help; student performance improved again (entirely on the difficult problems). Our interpretation is, first, flexibility in traversing the search space leads students to see the global structure of the search space and to develop effective heuristics; second, strategic help reinforces and extends these heuristics.

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 forty computer programs that help teach logic and surely more will follow [Croy 1986, Burkholder 1989]. Recently, we have developed the Carnegie Mellon Proof Tutor (CPT), a computer environment for proof construction that combines a sophisticated graphical interface with a system for strategic help; the latter is based on a logically complete and cognitively natural program for finding proofs in sentential logic [Scheines and Sieg, 1993; Sieg and Scheines, 1992; Pressler and Scheines, 1988].

The design of CPT was based on the belief that students learn more from exercises when their problem solving environment has three features. First, its interface must relieve the student of non-essential cognitive load. That is, it must take over routine calculations, display the problem state informatively, and identify errors instantly.[1] Second, it must allow students maximal flexibility in traversing the problem space. Third, it must provide locally appropriate *strategic guidance*. (This point, for sure, motivates most "intelligent" computer tutors, e.g. [Anderson 1985b].) Students quickly master the syntactical constraints on logic proofs, but it takes them significantly longer to internalize the strategic subtleties of proof construction.

In this study we report on a series of experiments that tested the pedagogical value of three components of CPT's problem solving environment, namely: 1) an informative and manipulable display of the problem state, 2) the flexibility of working forwards and backwards, and 3) minimal strategic help at any point in a problem. We gathered data

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

relevant to the first two components in the fall semester of 1989 and data relevant to the third in the spring of 1989 and the fall of 1990. The 1989 data support the utility of the first feature and even more strongly the utility of being able to work forwards and backwards. The 1990 data confirm the importance of strategic help.

These results undermine the view that working backwards is the preferable method for proof construction, but their significance goes beyond this parochial logical point. A computer-based logic course, used as a platform for our experiments, allowed us to isolate the impact of *problem space representation* and *space traversal possibility* on student performance. *Informativeness of the former and flexibility of the latter improved performance.* -- These features of the computer environment and the underlying logical system are described in section 2. At the end of that section we formulate more precisely the central issues we wanted to clarify. Section 3 is devoted to a discussion of our experimental methods, whereas section 4 contains the central results. The last section examines these results with reference to our original questions.

2. Constructing Proofs (with CPT). Most philosophy departments offer courses in formal logic, and most of these classes assign a large number of proof construction exercises. There is a great variety of logical systems within which formal proofs can be given, but most share one feature: a proof is a sequence of lines, each of which is either an assumption or the result of applying a logical "inference rule" to previous lines. One begins a proof construction problem with a gap between what is given (the premises) and what is sought (the conclusion). Two straightforward methods for closing this gap are forwards and backwards chaining. In forward chainging the gap is closed by inferring new lines from the premises, and in backward chaining the gap is closed by creating subgoals such that the line desired is proved as soon as the subgoals are proved. Although it is rarely taught as such, finding a proof in a formal framework is thus a classic search problem, as formulated for example in [Newell and Simon, 1972]. One can work forwards to a goal G in the forest of lines that follow via the inference rules from assumptions Γ (Fig. 1),

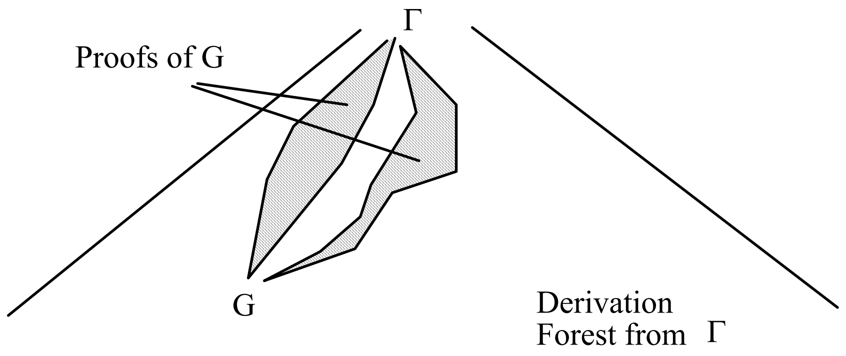


Fig. 1

work backwards to the premises Γ in the inverse forest of arguments that *lead to* G (Fig. 2),

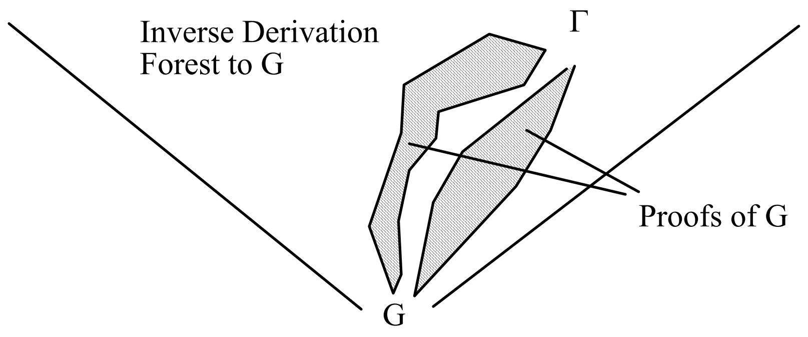


Fig. 2

or confine the search to the region where the forward and backward searches overlap (Fig. 3).

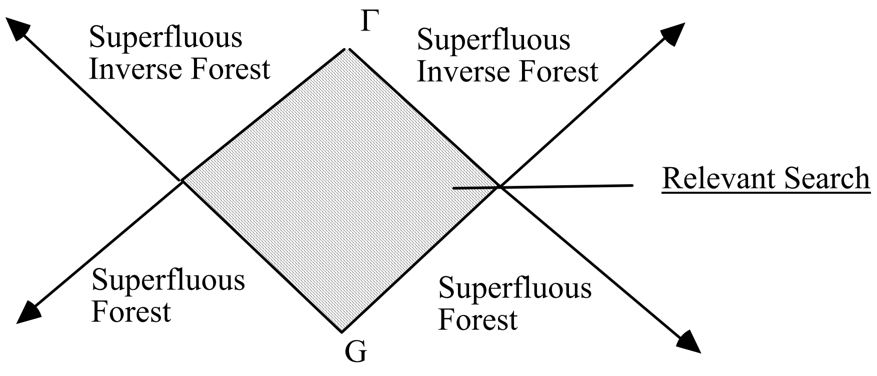
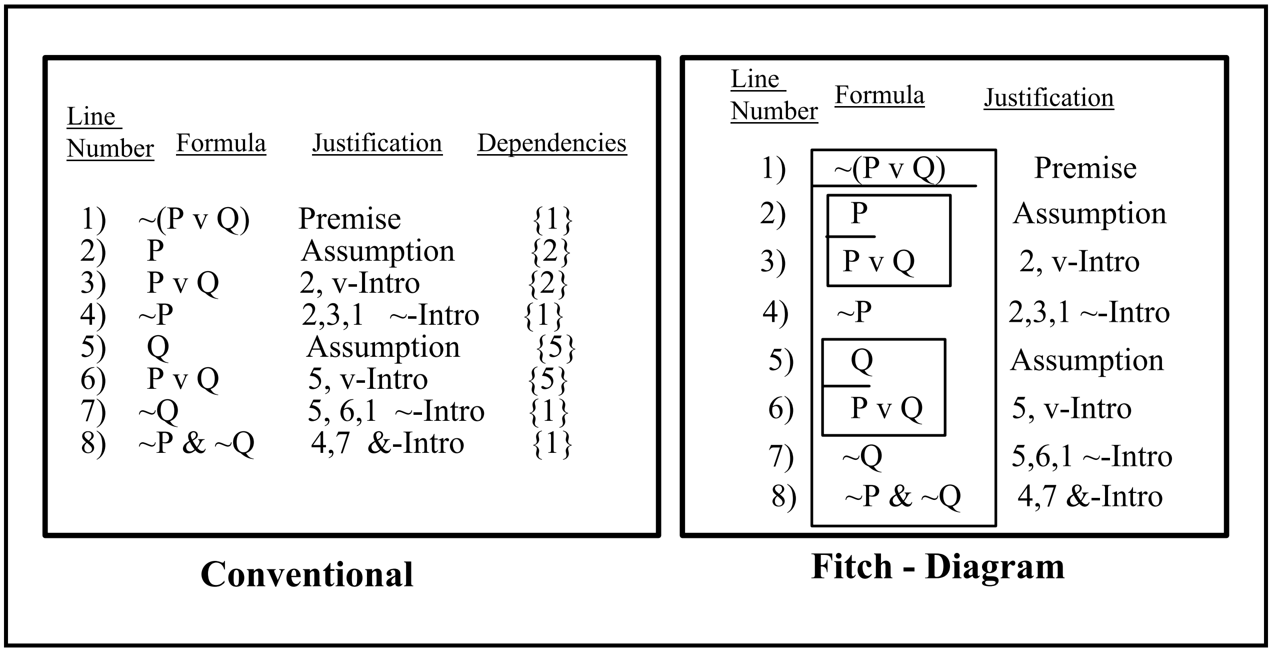


Fig. 3

All the standard problems associated with a Newell and Simon search problem arise in proof construction. Students wander down dead ends and cannot recover, travel in circles, and move laterally instead of vertically, i.e., they make moves which are legal, but which do not bring them closer to a solution.

As the logical frame, we chose "natural deduction systems" introduced by Gentzen in the thirties. (See [Gentzen, 1969].) They are indeed natural, as their inference rules correspond directly to steps in informal deductive arguments, in particlar mathematical proofs. All such systems allow the introduction and discharge of temporary (working) assumptions, both of which are essential to mathematical proof. In our experience, however, novice students find this feature of "natural deduction systems" particularly difficult.

How can a computer environment aid students who are learning proof construction? CPT provides all the operations that are now standard for logic programs, i.e. mistake correction and calculational support, but it also provides an informative display, the possibility to work forwards or backwards, and strategic help. CPT's display (Fig. 5) is novel in that it combines, in a graphical way, information about the search with information about the structure of temporary assumptions. To address the last issue first: if a proof is represented by a sequence of lines, then there is associated with each line a set (possibly empty) of temporary assumptions on which that line depends. Thus, when a temporary assumption q is introduced, any line that depends on q includes q's line number in its dependency list. So-called Fitch diagrams, introduced by the American logician Fitch and used in many logic textbooks, [Kalish, Montague, and Mar, 1980] for example, offer a superior option for displaying logical dependency: a box is drawn for each additional temporary assumption. Every formula inside the box depends on this assumption. When the assumption is discharged, the box is closed. Thus determining logical dependency in a Fitch diagram is cognitively as natural and easy as deciding box containment. Consider the respective displays of a proof in Fig. 4.[2]



Conventional

Line Number	Formula	Justification	Dependencies
1)	~(P v Q)	Premise	{1}
2)	P	Assumption	{2}
3)	P v Q	2, v-Intro	{2}
4)	~P	2,3,1 ~-Intro	{1}
5)	Q	Assumption	{5}
6)	P v Q	5, v-Intro	{5}
7)	~Q	5, 6,1 ~-Intro	{1}
8)	~P & ~Q	4,7 &-Intro	{1}

Fitch - Diagram

Line Number	Formula	Justification
1)	~(P v Q)	Premise
2)	P	Assumption
3)	P v Q	2, v-Intro
4)	~P	2,3,1 ~-Intro
5)	Q	Assumption
6)	P v Q	5, v-Intro
7)	~Q	5,6,1 ~-Intro
8)	~P & ~Q	4,7 &-Intro

Fig. 4

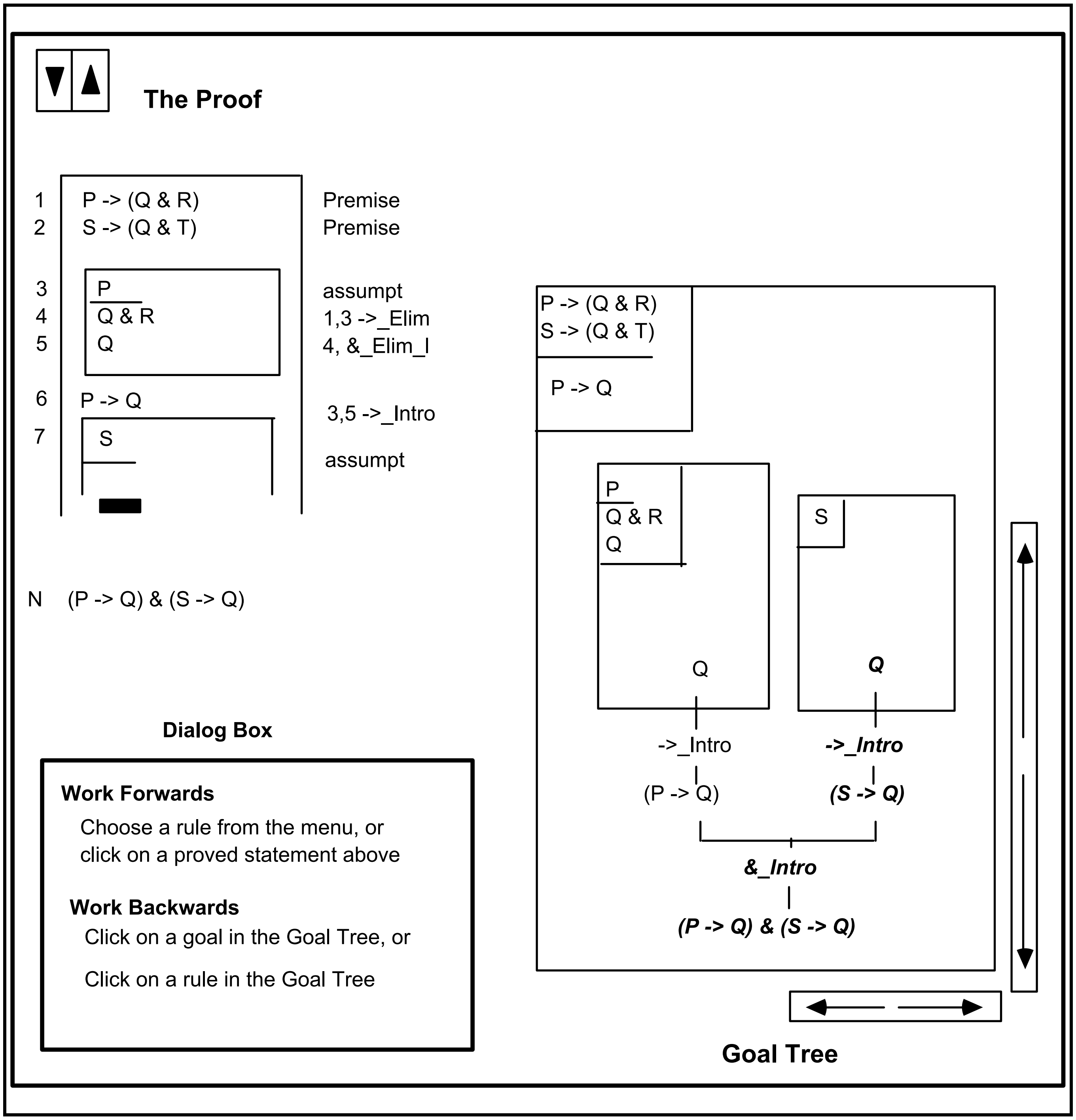
Glancing at the conventional diagram, it might appear strange that lines 3 and 6 contain the same formula. It might also appear that we have derived several contradictions, e.g. lines 5 and 7. Neither "oddity" appears in the Fitch diagram: one can

[2]We use the symbol "~" for negation.

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* (~-Intro) from this assumption.

Fitch diagrams do not represent, however, the search for a proof any better than a column of lines, and we decided to complement the Fitch diagram by a tree-like diagram of subgoals to give students a concrete representation of their backwards search effort. The "Goal Tree" is thus provided in parallel to the standard Fitch diagram (Fig. 5).[3]

[3]Jonathan Pressler and Richard Scheines designed the Goal Tree display in 1988.



```
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

Fig. 5

Students can work forwards or backwards in CPT, and they can determine at a glance the state of their search and the logical structure of their temporary assumptions. To work forwards, students must select an inference rule and lines in the proof to which this rule will be applied. Lines are selected by clicking on them either in the Fitch diagram or in the Goal Tree; inference rules are selected from pull down menus at the top of the screen (not shown in Fig. 5). To work backwards, a goal must be selected by clicking on a formula in the Goal Tree, where unproved goals are shown in italic and bold font, but those that have been established are in regular font. Once a goal has been identified, an inference rule for reaching this goal must be selected from the pull down menus. If the goal and rule fully determine the subgoals (premises for the rule), then CPT fills in these subgoals automatically. If not, the user is prompted at the appropriate location on the screen to type in the subgoals, or click on formulas somewhere else in the display which are then entered (possibly after some editing) as the subgoals. Since it is possible to enter inappropriate subgoals, the program checks whether the selected rule, when applied to the subgoals, will in fact produce the goal originally identified.

Our experience has been that students master the (underlying logical) operations very quickly and are then free to turn their full attention to the real difficulty in proof construction as a problem solving task: *strategy*. Once proof construction problems exceed a basic level of difficulty (perhaps 5-7 step proofs), the search space becomes large and one must follow heuristics in order to find a solution. After years of teaching proof construction in a variety of settings, it became clear that our advice to students was usually one of a small number of relatively simple strategic principles. We thought it would be better if students could receive instantaneous strategic guidance instead of having to wait and see a professor or TA. So we developed an algorithm for proof construction that employed the strategies and heuristics we had been teaching, and Sieg, in the summer of 1987, proved that algorithm to be complete; cf. [Sieg and Scheines, 1992].

Because of the algorithm's completeness, students can request strategic help at any point in a problem. The student identifies a goal in the Goal Tree, and CPT finds a proof of that goal (if indeed it is provable). It then offers strategic help in one of three ways. It will either 1) display the solution and part of the search it conducted to find that solution (by completing the Goal Tree that emanates from the goal the student selected), or 2) allow the student to step through its search for the solution (by adding one backwards rule-application to the Goal Tree above the selected goal each time the student hits the space bar), or 3) enter into a simple dialogue with the student concerning the "next" backwards step toward a solution. Having designed such a system and tested it informally on dozens of students, our goal was to test it experimentally in a controlled way. The crucial question was: *Do the goal-tree display, the possibility to work backwards and forwards, and the strategic help objectively improve performance?*

3. Methods

3.1 The Experiments. Every semester, 30-70 students take *Introduction to Logic* at Carnegie Mellon from a computer program called VALID that was developed at the Institute for Mathematical Studies in the Social Sciences (IMSSS) at Stanford University in the 1970s [Suppes, 1981]. VALID thoughtfully covers an extensive introductory curriculum and is now used at perhaps half a dozen universities around the country.

VALID alternates between introducing logical material and assigning exercises to build corresponding skills. After the material has been introduced via several screens of text interspersed with multiple choice questions, students are presented with a menu of proof construction exercises. Upon selecting an exercise, they shift to VALID's interactive proof construction environment and work on the exercise. Having finished or partially finished an exercise, they return to the menu, or, if enough exercises on the menu are successfully 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 new material. The bulk of a student's time is spent working on exercises, not on reading text.

VALID's proof construction environment uses a line presentation of proofs and a standard proof checker. We modified the course by replacing VALID's proof construction environment for sentential logic with that of CPT. Students who are taking *Introduction to Logic* use CPT for approximately five weeks. They complete exercises and take a midterm in that environment. (Of course, they receive no strategic help while completing the midterm.) In the fall semester of 1989, we conducted an experiment to compare the pedagogical effectiveness of three different versions of the proof construction environment (pce): in the first version students could only work forwards (and did not have the Goal tree display); in the second version they had the full CPT display, but could work only backwards; the third version was the full CPT environment (without strategic help).

To minimize the differences between treatments, each pce was implemented inside the CPT shell. Thus, all operations except for those actually involving proof construction were identical for the three groups. The first pce (for the "Forwards group") was a standard proof construction environment. Students using this pce could *only work forwards* and were given the standard column representation of proofs, i.e., their display resembled the left side of Fig. 4 and did not include a Fitch diagram or a Goal Tree. The second pce (for the "Backwards group") was a version of CPT in which the problem was displayed as we show in Fig. 5, with both a Fitch diagram and a Goal Tree. However, in this pce the students could *only work backwards* from the conclusion toward the premises by subgoaling in the Goal Tree. The third pce (for the full "CPT group") was the same as the second, but students were allowed to work backwards in the Goal Tree and forward in either the Goal Tree or Fitch diagram. None of the pces provided strategic help in the fall of 1989.

On the first day of classes we gave all students a pre-test for logical aptitude designed and validated at Stanford. We used the results of this test to split the class into three matched groups of 11 students each. We maximized the range of ability within each group, and made sure that each group showed a similar distribution of abilities. Each group proceeded through the VALID curriculum, but used "its" pce for the exercises and the midterm. The groups were initially trained for half an hour by a TA in the use of their particular pce.

The midterm exam consisted of eight proof construction problems similar to those the students had faced before the midterm. From prior tests, using student performance on those tests as a guide, we chose two problems that were easy, three that were of medium difficulty, and three that were hard. Students were given three hours to complete the test. Besides recording whether they did or did not succeed in completing a given problem, we measured how much time it took them to finish a problem, how many steps were in the completed proof, and how many errors of logical dependency were 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 for classes during the next two semesters as well. In the spring of 1990 we had 35 students, and in the fall of 1990 we had 42; but neither of these classes were split into groups. We added strategic help to CPT in the spring of 1990, but the help system was quite buggy; in fact it was hardly used, once the word got out that asking for help frequently caused the system to crash. Only 6 students asked for help more than 5 times during that semester. We therefore consider the spring 1990 data more relevant to replicating or failing to replicate the performance of the full "CPT group" from the fall of 1989. The help system was fixed for the fall semester of 1990. During that semester students were encouraged to use it often and, indeed, the 42 students asked for strategic help more than 1,600 times. To the 1990 classes we gave a substantially harder midterm than to the 1989 class. The first eight problems were identical (up to notational variations) on all midterms, but for the 1990 classes we added two new and very difficult problems. We also increased the time pressure: instead of three hours for 8 problems, we gave the 1990 groups only two and a half hours for 10 problems.

Fig. 6 shows the pre-test means in percents; the overall distribution of pre-test scores for the spring and fall classes was approximately Gaussian. The Backwards group is slightly high, because three of the poorer students dropped out after we had matched the groups. One student dropped out of the Forwards group.

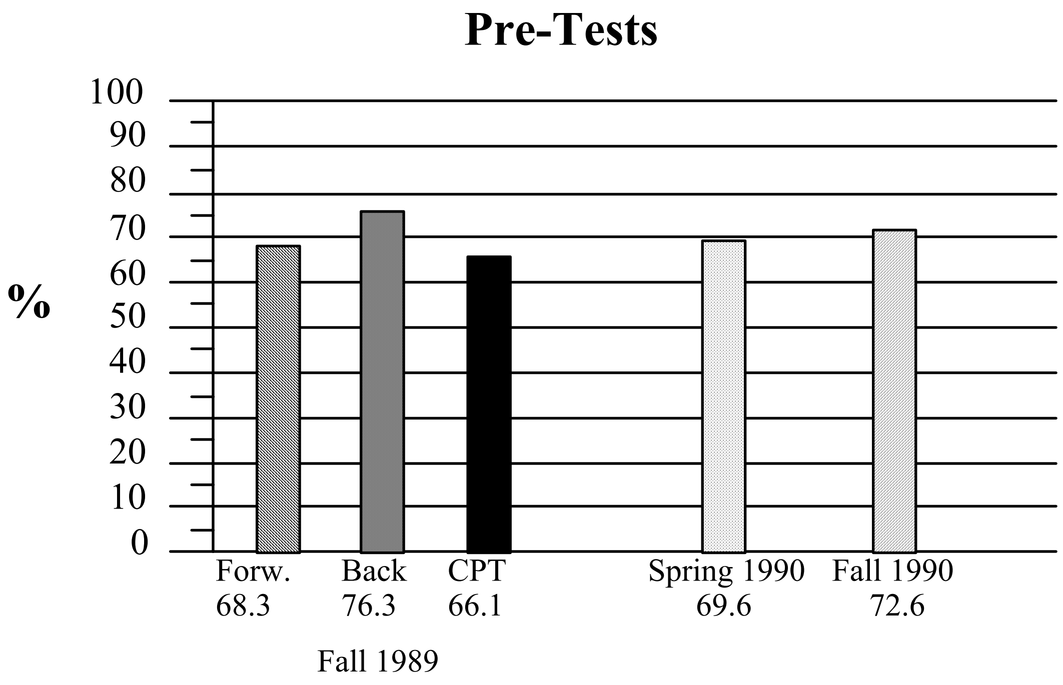


Fig. 6: Pre-test Scores

Students in the CPT group, who were allowed to work forwards as well as backwards, chose to work backwards 46% of the time and thus used those parts of their environment that the alternative groups could not. Since each group completed more than one hundred proof construction problems over the course of five weeks with their own pce before taking the midterm, training effects were negligible.

3.2 Statistical Methods. Although these experiments involve a total of 107 students in three different classes, no statistical comparison involves the whole sample. The largest comparison involves two aggregated groups with a total sample of 65, and this comparison shows a difference that is "significant" at .05; cf. Section 4.2. All comparisons in the fall of 1989 involve groups of approximately 10 students. At this sample size, it is unrealistic to expect statistically significant differences in measured group performance. For example, supposing that two groups of 10 students -- identical on all covariates -- each show normally distributed post-test scores with a standard deviation of 20 percentage points. Then if the mean for the control group is 65% (a D grade) and the mean for the treatment group 83% (a B grade), this difference is not "significant" at .05. Such an observed difference in the post test could be due to chance, but to infer that it consequently "doesn't mean a thing" is, in our view, to take classical hypothesis testing much too seriously. Precisely these sorts of cases, where observed differences are large but "statistically insignificant" because of small sample size, have motivated modern Bayesian statistics.

Rather than do a full Bayesian analysis of our data, which would involve a complicated and fairly arbitrary specification of a prior distribution, we choose to regard our results as strongly suggestive, not as conclusive: the differences in group performance are stable and show a very clear pattern. Taken individually, only one aggregated group comparison is statistically significant on a two-tailed t-test at .05. That comparison exhibits the *same differences* in mean as other, similar comparisons; the other comparisons are distinguished from that one *only* by involving a smaller sample. Thus, taken together, the comparisons below provide much stronger evidence than individual hypothesis tests would suggest.

4. Results

4.1 Overall Performance. As most histograms were reasonably symmetric, we compare groups by comparing their means. Fig. 7 shows the mean percentage of proofs correctly completed for those problems that were presented to all groups. To compare group scores as they relate to the group's aptitude, we display each group's midterm score next to its pre-test score. Comparing the midterm means to the pre-test means is in itself meaningless, but the relative difference between pre-test and midterm score gives a basis for comparing the various treatments. While the forwards and backwards groups scored (by coincidence) nearly the same percentage as in their pre-test, every other group improved on their pre-test score by 12-18 percentage points.

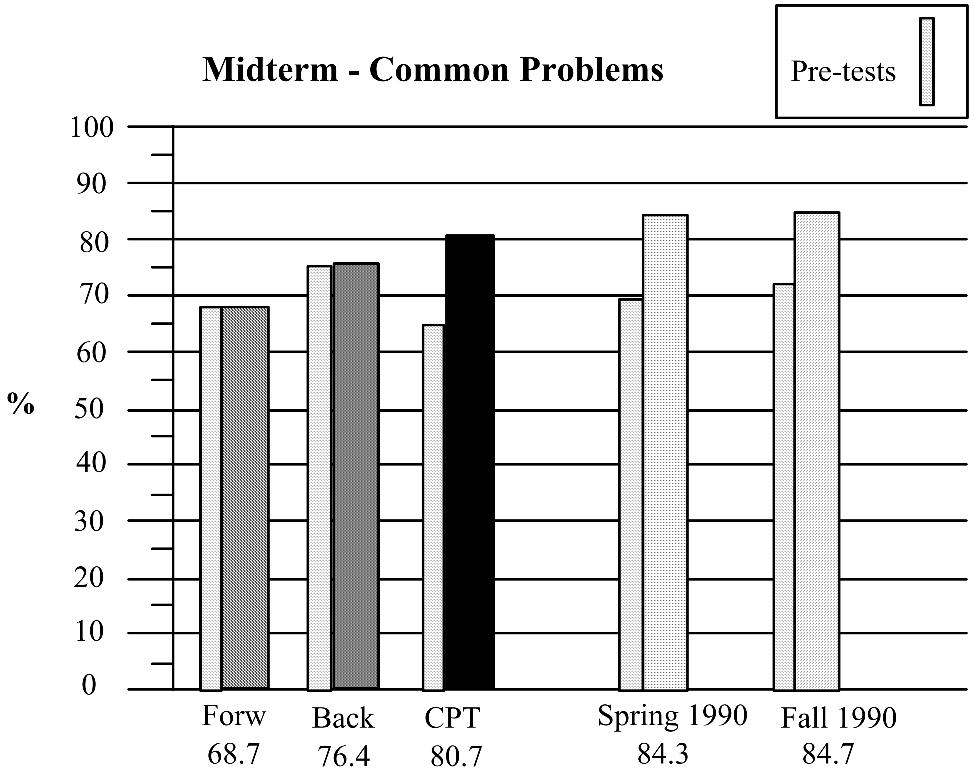


Fig. 7

The groups were nearly indistinguishable on the easy and medium problems, but the difference in performance on the hard problems common to all groups was striking (Fig. 8).

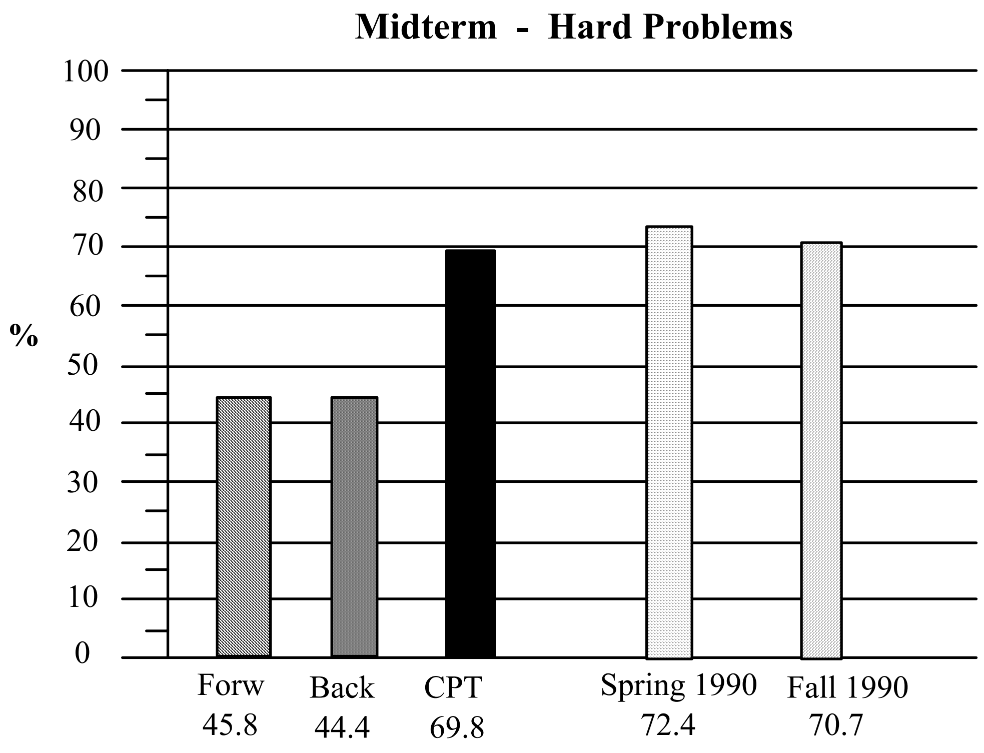


Fig. 8

4.2 Interface Effects. Prior to performing this experiment, we were convinced that working backwards was a superior method for finding proofs in logic. We thus expected to see a marked difference in performance between the group that could only work forwards and the group that could only work backwards, with the latter having the advantage. As is obvious from the data this difference did not materialize, but a related one did: groups that were restricted to work in one direction did dramatically worse than those allowed to work in both directions. To make this difference vivid, we aggregated the forwards and backwards groups into a "1-way" group, and compared it to a "2-way" group that is an aggregation of the full CPT in 1989 and Spring 1990 groups.

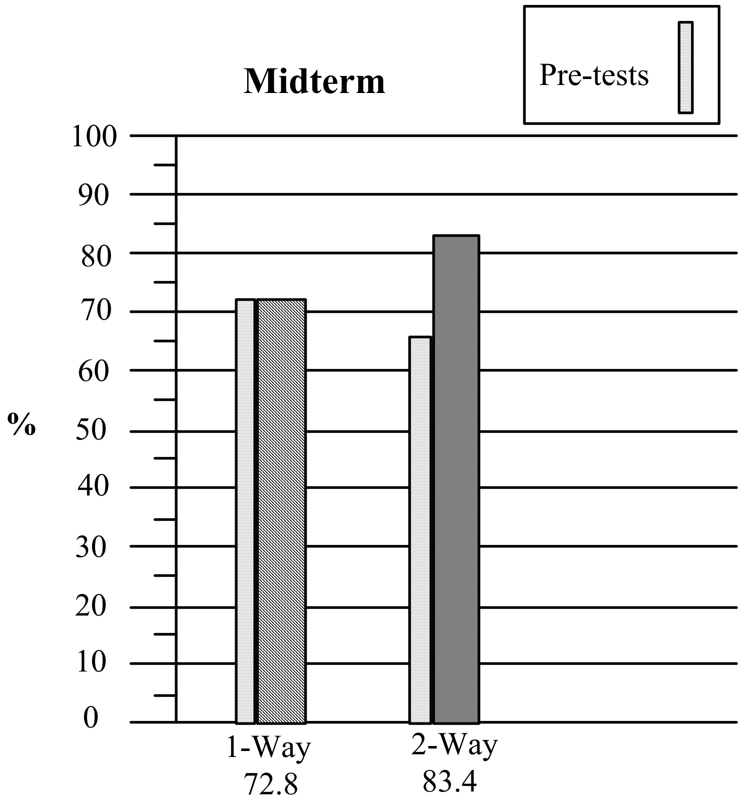


Fig. 9

The 2-way group outperformed the 1-way group on the overall midterm (Fig. 9), and the difference between their scores on the hard problems is dramatic (Fig. 10), enough so to be statistically significant at .026 on a T-test. That is, if we assume that the two groups went into the midterm with equal ability to solve the hard problems, there is a 2.6% chance that we would observe as large a difference in their means as we did or a larger one. (Actually, the 1-way group apparently had slightly more ability: its mean pre-test score was 72.5 compared with 68.7 for the 2-way group.) Thus, we reject the hypothesis that the groups were equal when they took the midterm, and since the only apparent difference was the pce each group used, we assume that differences in pces caused differences in proof construction performance on hard problems.

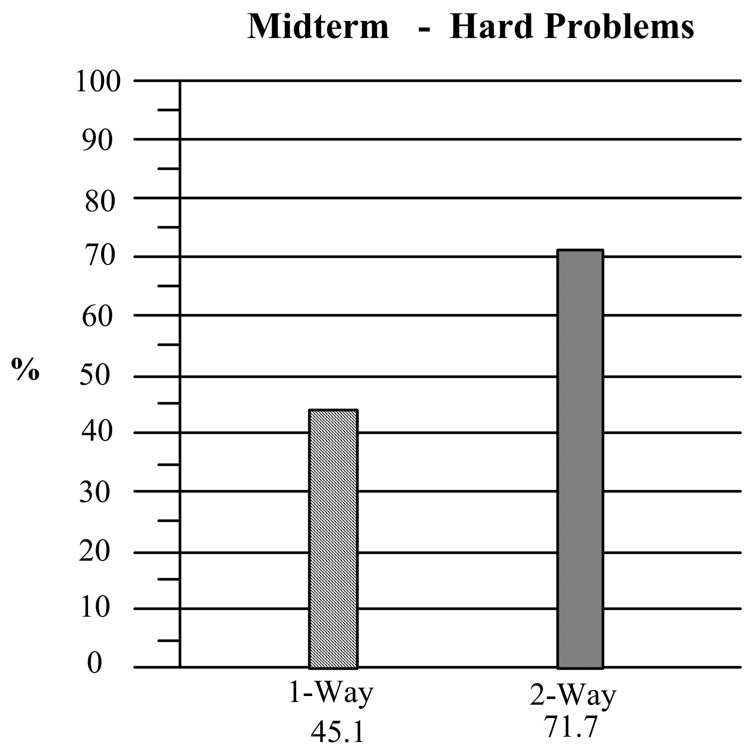


Fig. 10

Even though our preconceptions about working backwards were not borne out in overall performance, other measures suggest that they are at least partially correct. Students who worked backwards seemed to find more elegant proofs than those who worked only forwards, as the average number of steps in completed proofs is smaller by almost 15% (Table 1). They also took slightly less time to find proofs.

Group	Average number of steps in completed proofs	Average number of seconds to complete proofs
Forwards Only	16.7	872
Backwards Only	14.3	835
Full CPT	13.8	835

Table 1

The Fitch diagram and the corresponding boxes in the Goal Tree are intended to help students master the relatively difficult topic of logical dependency. To test the effectiveness of these representational aids, we kept track of student attempts to make inferences that were illegal because of inappropriate logical dependencies. In fact, one cannot make such an error while working backwards, so for this hypothesis we compare the forwards only group to the groups using the full interface. Including the Spring group, the 43 students in the full CPT groups made 3 errors of logical dependency on the midterm, while 11 students in the Forwards group made 11 such errors.

4.3 Strategic Help. The interface effects were most striking on the hard problems, and so far the effect of strategic help seems to be confined to the extremely hard problems (Fig. 11). As to our interpretation, see section 5.

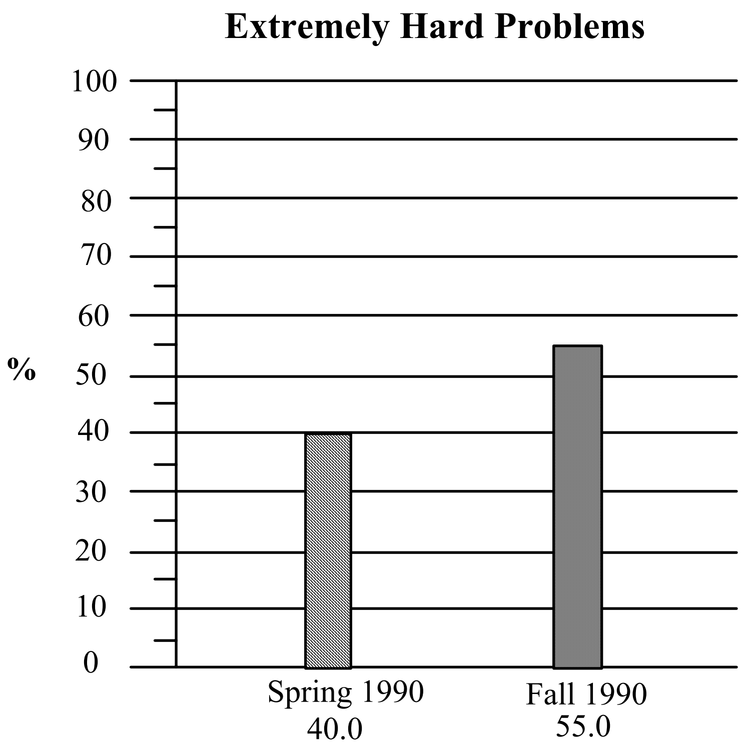


Fig. 11

5. Analysis. We set out to test three features of CPT's problem solving environment: 1) the informative and manipulable display of the problem state, 2) the flexibility to work forwards and backwards, and 3) the strategic help at any point in any problem. Although the data seem to indicate that all three features benefit the student, they give the strongest support for the flexibility to work both ways on a problem. Students forced to work exclusively in either direction were at a disadvantage, as is evidenced by the large differences between the 1-way and 2-way groups (Figs. 9 & 10). The CPT group differed from the Backwards group only in the ability to work forwards, so being able to work forwards clearly matters. 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. As a result, the utility of working backwards is more difficult to assess in this case. It seems unlikely, however, that the difference between these two groups was entirely due to the display. After all, the Forwards group and Backwards group differed also in their displays, but did not exhibit nearly the difference the CPT and Forwards group did. It is our interpretation that only the 2-way groups learned to separate those proof contexts in which working backwards was advantageous from those in which working forwards was better. From these results we hypothesize that *flexibility in traversing the search space leads students to see the global structure of the search space and to develop appropriate heuristics*.

The Goal Tree and Fitch diagram embody the informative display that was used by every group except the Forwards group. That it mattered is evidenced, first, by the difference between the number of errors in logical dependency that were made by the Forwards and CPT groups. Such errors can be made in either group's pce, but the Forwards group actually committed them far more often. Second, although the difference in midterm success rate between the Forwards and Backwards groups matches their pre-test difference almost exactly, the Backwards group outperformed the Forwards group in time of search and in the size of the proof found. Our preliminary conclusion is that the Backwards group searched more efficiently and with fewer errors because of the Goal Tree diagram. This is supported by subjective surveys administered to each of the groups we discussed and to several groups who took *Introduction to Logic* after 1990. Students were asked to rate features of the tutor on a Likert scale ranging from one (strongly disagree), to three (neutral), to five (strongly agree). The mean ratings for "The Goal Tree display is helpful" and "Being able to work backwards is helpful" have been well above four (agree) in each survey; indeed, they approached five (strongly agree) on more than one occasion. No other feature of the tutor is as popular as either of these two.

Assessing the effect of strategic help is more difficult and in this paper has to be considered preliminary. We hypothesize that *the help dialogue students enter when facing difficulties makes strategies locally explicit and reinforces the heuristics already developed for when to use particular strategies.* Because it would have required changing the VALID curriculum, we did not emphasize the strategic approach to proof construction that underlies CPT's proof search. In a future study, we will teach one group with a new text emphasizing CPT's method of proof construction. Then we will compare its performance (1) to that of a group using the standard VALID curriculum and CPT without help and (2) to that of a group using the standard VALID curriculum and CPT with help. We expect that the explicit teaching and practicing of strategic considerations will improve performance.

Even if it is somewhat difficult to separate the effects of the three features on performance, it is not hard to see how well they work in combination. The 1990 groups used the full CPT environment, while the Forwards group used a standard proof checker. The 1990 groups took a substantially harder midterm than the group that used a standard proof checker, but were given only 83% of the time. Even so, on the common part of the midterm their average score was over 21% higher than the group using a standard proof construction environment with a relatively uninformative display, the ability only to work forwards, and no strategic help.

References

Anderson, J.R.. (1987). Production systems, learning, and tutoring. In: D. Klahr, P. Langley, & R. Neches (Eds.), *Production System Models of Learning and Development*. Cambridge, MA: MIT Press, pp. 437-458.

Anderson, J.R. (1980). *Cognitive psychology and its implications*. San Francisco: W. H. Freeman.

Anderson, J.R., Jeffries, R. (1985a). Novice LISP errors: Undetected losses of information from working memory. *Human-Computer Interaction*, 1, 107-131.

Anderson, J. R., Boyle, C.F. (1985b). The geometry tutor. *Proceedings of IJCAI-85*, Los Angeles, CA.

Burkholder, L. (1989). Cumulative new software. *The Computers and Philosophy Newsletter*. Center for Design of Educational Computing, Carnegie Mellon University, 4.1, 48-89.

Croy, M. J. (1986). The current state of computer-assisted instruction for logic. *Teaching Philosophy* , 9, 333-350.

Gentzen, G. (1969). Investigations into logical deduction. In: M.E. Szabo (Ed.), *The collected papers of Gerhard Gentzen.* Amsterdam: North Holland, 68-131.

Kalish, D., Montague, R., and Mar, G. (1980). *Logic: techniques of formal reasoning.* New York: Harcourt Brace Jovanovich, Inc.

Larkin, J. (1987). Display-based problem solving. In *The 21st Carnegie Symposium*, Hillsdale, N.J.: Erlbaum.

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

Pressler, J. (1989). Computers in logic courses. In: Ian Lancashire (Ed.), *Methodologies in humanities computing*. Philadelphia: University of Pennsylvania Press.

Pressler, J., and Scheines, R. (1988). An intelligent natural deduction proof tutor. *Computerised Logic Teaching Bulletin.* 1.

Scheines, R., and Sieg, W. (1993). The Carnegie Mellon Proof Tutor. In Judith V. Boettcher (Ed.), *101 Success Stories of Information Technology in Higher Education: The Joe Wyatt Challenge*. McGraw-Hill.

Sieg, W., and Scheines, R. (1992). Searching for proofs (in sentential logic). In: Leslie Burkholder (Ed.), *Philosophy and the Computer.* Boulder: Westview Press.

Suppes, P. (Ed.), (1981).*University-level Computer-assisted Instruction at Stanford: 1968-1980*, Institute for Mathematical Studies in the Social Sciences, Stanford University.