Menu Selection vs. Typing: Effects on Learning in an Intelligent Programming Tutor

Albert T. Corbett, John R. Anderson, and Jon M. Fincham
Department of Psychology
Carnegie Mellon University, Pittsburgh, PA USA 15213

Abstract: A study is reported comparing a menu-driven and type-in interface for an intelligent tutoring system that assists students with Lisp programming exercises. In the menu-driven condition, students selected code symbols from a two-level hierarchical menu system. In the type-in condition students typed each symbol in their programs. In either case, the tutor traced the students' solutions symbol-by-symbol by means of a production system student model and kept the students on a correct solution path. Contrary to our expectations, menu-driven code entry impeded learning. Students in the menu condition made more mistakes in completing the tutor exercises and performed substantially worse on post tests. Students in the menu condition generally had a more positive opinion of the tutor, although not if they switched to it from a type-in interface.

This study compares a menu-driven and a type-in interface in a practice environment for programming exercises. Students in this study worked through an introduction to the Lisp programming language. They read six chapters of a Lisp programming text and completed exercises with the assistance of the CMU General Programming Languages Tutor (Anderson, Corbett, Fincham, Hoffman and Pelletier, in press). In each exercise, this tutor presents a description of a short program to write and as the student enters a solution, the tutor monitors the student's performance symbol-by-symbol, providing feedback on errors and providing a correct action if the student appears to be floundering.

Figure 1 presents a snapshot of the screen as the student begins an exercise with the tutor. The problem description appears in the top window and the student's solution appears in the code window immediately below. Messages from the tutor appear in the third window. Figure 1 depicts the menu-driven version of the tutor. As can be seen the menu appears immediately to the right of the code window. In this snapshot, the student has just selected **defun** from this menu and the tutor has expanded a template for this operator in the code window. The template includes a set of parentheses and three goal nodes in angle brackets. These nodes are not themselves Lisp code, but serve as goal reminders which the student replaces with Lisp code. Note that the tutor employs a hierarchical menu system. With the exception of **defun**, no code symbols appear in the top level menu. Rather, operators are organized by categories. To enter the arithmetic operator +, for example, the student selects the entry **Arithmetic Functions**. Then the student selects + from the menu that appears. Certain symbols, e.g., numbers and variables must be typed. In this case, the student selects **Type In** from the top level menu, a more specific category from the resulting submenu, then types the symbol. In the type-in version, students type all code symbols in

their solutions without any menu operations. If the student makes a mistake, the tutor interrupts immediately to notify the student. The tutor allows the student to try again, and does not provide any explanatory feedback. At any time, however, the student can ask for help at a goal node.

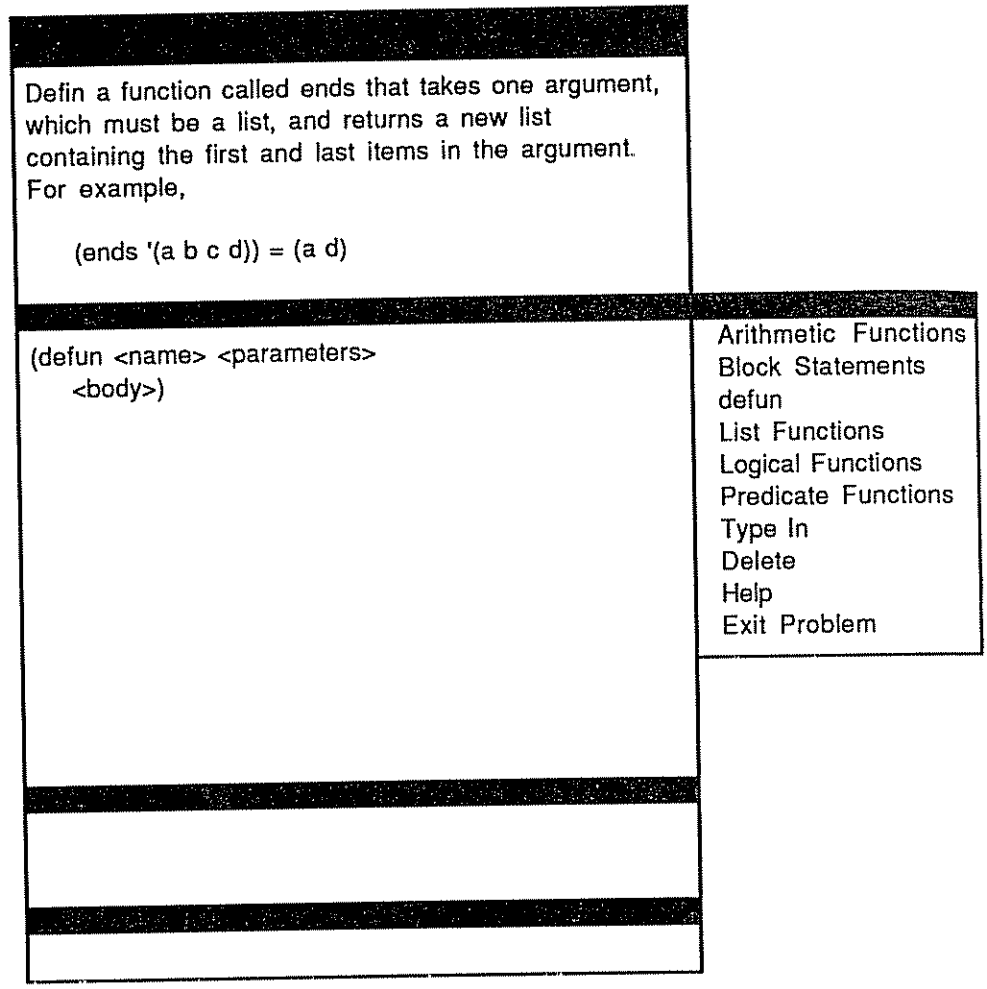

Figure 1. A snapshot of the screen as the student begins an exercise with the programming tutor.

The tutor employs a model tracing paradigm to provide assistance to students (Anderson and Reiser, 1985; Corbett, Anderson and Patterson, 1990). The tutor contains a production-system model of the knowledge students need to complete the exercises in the curriculum. This student model allows the tutor to solve the exercises step-by-step along with the student and to provide help if a student makes a mistake. The architecture of the program is similar to the CMU Lisp Tutor, which has been shown to be an effective practice environment (Anderson and Reiser, 1985; Corbett and Anderson, 1991). The architecture

of the newer tutor is more general than that of the Lisp Tutor, however, and can accept student models for a variety of languages. Currently, it also has a student model for Pascal and Prolog. (See also Gray and Atwood, in press, concerning a COBOL application).

The Lisp Tutor has a type-in interface, but in the course of developing the General Programming Languages Tutor we speculated that a mouse driven menu system might be more successful. Since menus are recognition based, they can make it easier for a novice to begin interacting with a system (Miller, 1988). We thought that at minimum, menus might reduce the time required to complete the tutor exercises. Beyond that, we speculated that menus might improve learning rate by reducing the cognitive load of entering code. To assess these possibilities, we implemented both interfaces for the tutor and compared them as described below.

Method

Subjects

Thirty-eight undergraduates participated in the experiment for pay. Math SAT and prior programming experience were controlled across groups.

Design

Students worked through 10 lessons with the tutor, completing 63 exercises. The first six lessons introduced new topics and were accompanied by text. Lessons 7-10 repeated the topics covered in lessons 3-6. Half the students used the menu-driven interface and half used the type-in interface for the first six lessons. For the remaining four lessons, nine students switched from the menu to type-in interface and ten switched from the type-in to menu interface. Students completed a paper and pencil test and filled out a questionnaire following the second, sixth and tenth lessons.

Results

Post tests

Results of the three paper and pencil post tests are displayed in Table 1. A strict accuracy measure is employed: percent of exercises completed without error. The upper half of the table displays the first two post test results for the two groups. As can be seen, students working with the type-in interface performed more accurately on both the first post test, ($F(1,36) = 5.27, p < .05$) and the second one ($F(1,36) = 6.77, p < .05$). Half the subjects switched interfaces prior to the final four lessons and post test. Thus, the menu and type-in columns are each divided into two columns at the bottom of Table 1. The two leftmost columns, respectively, represent students who started with menus and (1) stayed with menus or (2) switched to typing. Similarly, the remaining two columns represent students who started out typing and either stayed with typing or switched to menus. As can be seen, the three groups who used the type-in interface at some point performed similarly on the final post test, while the students who worked with menus throughout lagged behind. An analysis of variance of these results is marginally significant $F(3,34) = 2.22$, $p = .10$.

Table 1
Post Test Performance: Percent of Exercises Correct

	Menu		Type	
Test 1	38%		53%	
Test 2	11%		25%	
	Menu	Type	Type	Menu
Test 3	12%	24%	26%	30%

Tutor Exercises

Table 2 displays three measures of the students' performance in completing the tutor exercises. Again, the table is split, with the first six lessons at the top and the last four lessons at the bottom. The first measure in each half of the table is the mean time spent working on all exercises in a lesson. The two remaining measures describe performance at the level of individual goals and production firings. In these analyses, a production is said to have fired correctly when a student enters a correct code symbol on his/her first attempt at a goal. The two production-level measures are (a) mean time to enter a correct symbol and (b) the probability that a subject does not enter a correct symbol on his/her first attempt at a goal. (The first goal in each exercise is excluded from all analyses, since the time measures for this goal would include reading time for the problem description. Lesson 1 is excluded from the analyses, since the exercises involve at most 2-3 productions, one of which is automatically excluded.)

Table 2
On-line Performance Measures in Completing the Lessons:
Mean Time to Complete the Exercises in a Lesson, Mean Correct Production Firing Time, Probability of an Error at a Goal

Lessons 2-6	Menu		Type	
Time/Lesson	24.8 min		22.1 min	
Time/Correct	11.6 sec		11.0 sec	
Errors	0.26%		0.18%	
Lessons 7-10	Menu	Type	Type	Menu
Time/Lesson	16.8 min	16.7 min	15.6 min	18.7 min
Time/Correct	7.7	7.6	7.7	9.2
Errors	0.23	0.24	0.19	0.21

There was no reliable difference across the two groups in time to complete the exercises nor in time for a correct production firing. However, students in the menu condition were more likely to make an error than students in the type-in condition, $F(1,36) = 8.48$, $p < .01$.

There were no reliable differences among the time and accuracy measures for the final four lessons. The group that switched from typing to menus appears to have taken the longest to complete the exercises. If this is a real effect, it seems to be an interface rather than learning difference, since students in this group were as accurate as other groups, but took longer to enter a symbol.

Questionnaires

Each questionnaire contained seven to twelve questions that assessed students' self-monitoring of the learning process and how well they liked the tutor. Each of these questions employed a seven-point rating scale. The first two questionnaires asked how difficult the questions seemed and how well students had learned the material. On the second questionnaire, the exercises were rated more difficult in the menu condition than in the type-in condition (ratings = 5.5 vs 4.8, $F(1,36) = 3.29$, $p < .08$). Otherwise, there were no differences. The questionnaires also asked if the tutor had helped the students complete the exercises more quickly or understand them better than than they might working on their own. On the first questionnaire, students in the menu condition believed more strongly than those in the type-in condition that the tutor had helped them complete the exercises faster (5.8 vs 5.2, $F(1,36) = 3.12$, $p < .09$). Otherwise, there were no differences. In reality, students in the type-in condition finished the exercises as fast as faster. Finally, a variety of questions asked the students how much they liked the tutor, the code-entering facilities and the help facilities. There were no differences on the first questionnaire. Interestingly, on the second questionnaire, students in the menu condition liked the tutor better than those in the type-in condition (5.3 vs. 4.4, $F(1,36) = 3.39$, $p < .07$), liked the code entering facilities better (4.7 vs. 4.0, $F(1,36) = 3.01$, $p < .10$) and liked the help facilities better (5.2 vs 4.0, $F(1,36) = 12.68$, $p < .01$). The help facilities were identical across the two conditions, but the overall pattern here suggests students in the menu condition are happier with the tutor at the end of the sixth lesson.

On the final questionnaire, there were reliable differences among groups on two questions. The group that switched from a type-in interface to a menu interface liked the tutor less than the other conditions (rating = 3.7 vs. 5.4, 5.7 and 5.2, $F(3,34) = 4.19$, $p < .05$). The 3.7 rating is reliably smaller than each of the other three ratings. This same group found the interface harder to use than in the first six lessons, while the other three groups found their interface easier to use (rating = 5.0 vs. 3.0, 2.7 and 3.0, $F(3.34) = 4.98$, $p < .01$). The 5.0 rating is reliably greater than each of the other three ratings. There were no other reliable differences among the groups in rating how hard the exercises were, how much they liked the coding and help facilities, or whether the tutor helped them understand the material better or more quickly.

Discussion

Students using a menu driven interface made more mistakes in completing the tutor exercises and performed substantially worse on the post tests. There are at least three possible reasons for the post test result. One is that the students in the menu condition merely suffered from interface incompatibility in the post test, since a paper-and-pencil test

is recall-based. However, the fact that students in the menu condition were also making more mistakes while completing the tutor exercises suggests that there is more to the post test result than interface incompatibility. The evidence is that the menu-driven interface actually interfered with learning. We can imagine two reasons. First, menu selection may foster a trial-and-error approach to selecting code symbols at the expense of thinking more deeply about goals. Second, menu-selection, particularly with hierarchical menus, may represent a divided attention task, since students must repeatedly look away from the code, stop thinking in the exercise goal space, and think about where the desired symbol is in the menu. In effect, we may have increased cognitive load by introducing menus.

The questionnaire data are weakly suggestive concerning this issue. Initially, students working with menus believed more strongly that the tutor helped them finish quickly. If anything, students took longer in the menu condition, but students may have been confusing ease of problem solving with speed. If so, the ratings are consistent with the trial-and-error hypothesis. Dividing attention would be expected to increase perceived load. On the other hand, a trial-and-error approach in the face of uncertainty might be expected to reduce perceived load. By the second questionnaire, though, students in the menu condition perceive the exercises as more difficult, which appears to support the divided attention hypothesis. However, when exercises get hard, (which they begin to in lesson 3, judging from the post tests) students may also find it more burdensome to solve them with a trial-and-error method.

References

Anderson, J.R., Corbett, A.T., Fincham, J.M., Hoffman, D. and Pelletier, R. (in press). General principles for an intelligent tutoring architecture. In V. Shute & W. Regian (eds.) *Cognitive approaches to automated instruction.*

Anderson, J.R. and Reiser, B.J. (1985). The Lisp Tutor. *Byte, 10,* (4), 159-175.

Corbett, A.T. and Anderson, J.R. (1991). Feedback control and learning to program with the CMU Lisp Tutor. Talk presented at the Annual Meeting of the American Educational Research Association.

Corbett, A.T., Anderson, B.J. and Patterson, E.G. Student modelling and tutor flexibility in the Lisp Intelligent Tutoring System. In C. Frasson & G. Gauthier (eds.) *Intelligent tutoring systems: At the crossroads of artificial intelligence and education.* Norwood, N.J.: Ablex.

Gray, W.D. and Atwood, M.E. (in press). Transfer, adaptation and the use of intelligent tutoring technology: The case of Grace. In M. Farr & J. Psotka (eds.) *Intelligent computer tutors: Real world applications.* New York: Taylor & Francis.

Miller, J.R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. Polson & J. Richardson (eds.) *Intelligent tutoring systems.* Hillsdale, N.J.: Erlbaum.