In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*. Vienna, New York: Springer Wien New York. © CISM, 1997. Available on−line from http://um.org.

Student Modeling in the ACT Programming Tutor: Adjusting a Procedural Learning Model With Declarative Knowledge

Albert T. Corbett and Akshat Bhatnagar*

Human Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

Abstract. This paper describes a successful effort to increase the predictive validity of student modeling in the ACT Programming Tutor (APT). APT is an intelligent tutor constructed around a cognitive model of programming knowledge. As the student works, the tutor estimates the student's growing knowledge of the component production rules in a process called *knowledge tracing*. Knowledge tracing employs a simple two-state learning model and Bayesian updates and has proven quite accurate in predicting student posttest performance, although with a small systematic tendency to overestimate test performance. This paper describes a simple three-state model in which the student may acquire non-ideal programming rules that do not transfer to the test environment. A series of short tests assess students' declarative knowledge and these assessments are used to adjust knowledge tracing in the tutor. The resulting model eliminates over-prediction of posttest performance and more accurately predicts individual differences among students.

Mastery learning holds out the promise that virtually all students can master a domain if the domain knowledge is analyzed into a hierarchy of component knowledge units and if learning is structured so that students master prerequisites before moving to higher level knowledge (Bloom, 1968; Carroll, 1963; Keller, 1968). Meta-analyses confirm that mastery learning yields higher achievement levels (Kulik et al., 1990), but achievement gains in conventional mastery learning fall short of early expectations (Resnick, 1977; Slavin, 1987).

The ACT Programming Tutor (APT) is an intelligent tutoring system that employs a detailed cognitive model of programming knowledge in an attempt to achieve mastery learning. Our goal in the tutor is to monitor the student's growing procedural knowledge in the course of problem solving, provide sufficient learning opportunities for mastery and accurately predict students' test performance. This paper describes an important step forward in this modeling process: By incorporating an independent measure of students' prerequisite declarative knowledge we substantially improve the predictive validity of the modeling process.

In this paper we briefly describe the learning environment, the cognitive model, the learning and performance assumptions that underlie knowledge tracing, and the empirical validity of knowledge tracing. We describe a battery of declarative knowledge assessments we have developed and describe the improved predictive accuracy that is achieved by incorporating them into the model.

* This research was supported by the Office of Naval Research grant N00014-95-1-0847. We thank Dana Heath and Michele Mellott for assistance in data collection.

1 The ACT Programming Tutor

APT is a problem solving environment in which students learn to write short programs in Lisp, Pascal or Prolog. Each of these three modules is constructed around a language-specific cognitive model of the knowledge the student is acquiring. The cognitive model enables the tutor to trace the student's solution path through a complex problem solving space, providing feedback on problem solving actions and, if requested, advice on steps that achieve problem solving goals. This process, which we call *model tracing*, has been shown to speed learning by as much as a factor of three and to increase achievement levels relative to those of students working on their own (Anderson et al., 1995).

Figure 1 displays the ACT Programming Tutor Lisp Module midway through an exercise. The student has previously read text presented in the window at the lower right and is completing a sequence of corresponding exercises. The problem description appears in the upper left window and the student's solution appears in the code window immediately below. The student selects operator templates and types constants and identifiers in the user action window in the middle right. In this figure the student has encoded the operator *defun*, which is used to define a new operator, and has entered the operator name, declared input variables and begun coding the body of the definition. The three angle-bracket symbols in the figure, <EXPR1>, <PROCESS1> and <EXPR0>, are placeholders which the student will either replace with additional Lisp code or delete. Communications from the tutor appear in the Hint window in the lower left. In this figure the student has asked for a hint on how to proceed.

The tutor also tracks the student's growing procedural knowledge across problems in a process we call *knowledge tracing*, which is the focus of this paper. In *knowledge tracing*, the student is represented as an overlay of the ideal model (Goldstein, 1982). The Skill Meter in the upper right corner of Figure 1 depicts the tutor's model of the student's knowledge state. Each entry in the Skill Meter represents a production rule in the cognitive model of programming knowledge. The shading represents the probability that the student knows the rule and a check mark indicates that the student has mastered the rule.

1.1 The Cognitive Model

The tutors reflect the ACT-R theory of skill knowledge (Anderson, 1993). This theory assumes a fundamental distinction between declarative knowledge and procedural knowledge. Declarative knowledge is factual or experiential. For example, the following sentence and example in the Lisp text would be encoded declaratively:

The Lisp function *car* takes a list and returns the first element.
For example, *(car '(a b c d))* returns *a.*

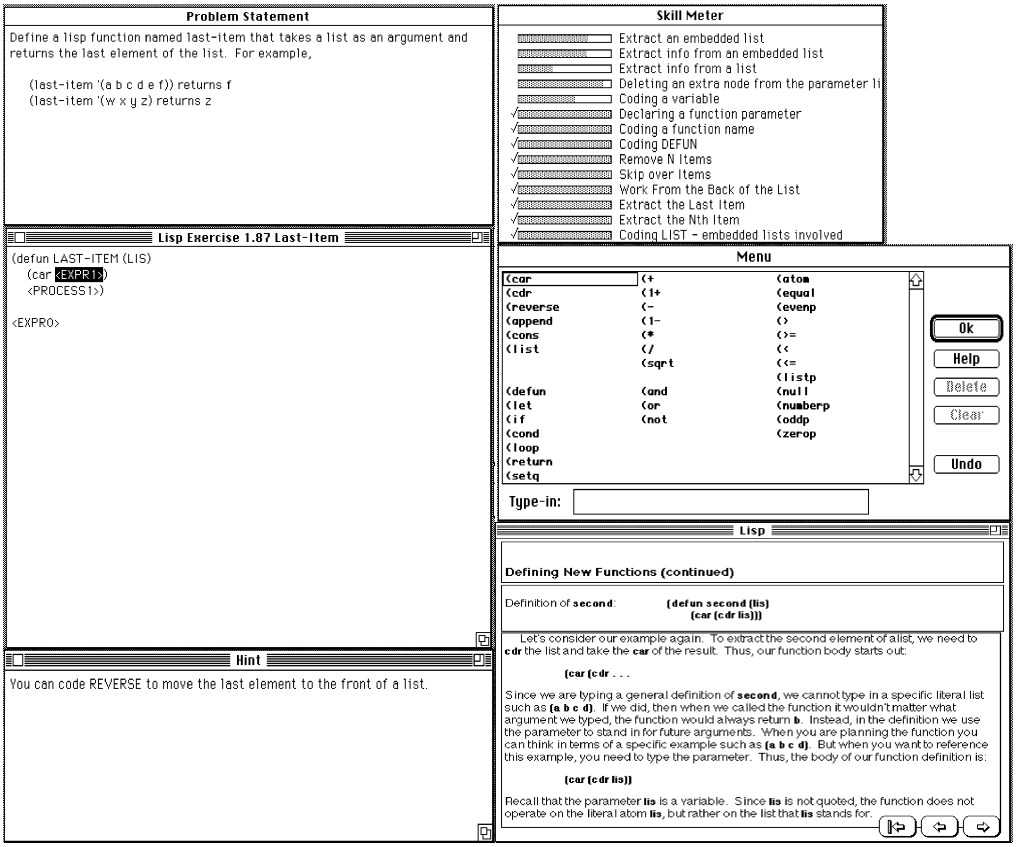

Figure 1. The APT Lisp Tutor interface.

Procedural knowledge, in contrast, is goal-oriented and mediates problem-solving behavior. ACT-R assumes that skill knowledge is encoded initially in declarative form through experiences such as reading and that domain-specific procedural knowledge results from problem solving. With practice, declarative and procedural knowledge are strengthened so that performance grows more rapid and reliable. Like many cognitive theories, ACT-R assumes that procedural knowledge can be represented as a set of independent production rules that associate problem states and problem-solving goals with actions and consequent state changes. The following two goal-oriented productions can be derived from the declarative example above through practice in writing function calls and evaluating function calls respectively:

IF the goal is to code an expression that returns the first element of a list.
THEN code the operator *car* and set a goal to code the list as its argument.

IF the goal is to evaluate an application of *car* to a list,
THEN write the first element of the list.

2 Knowledge Tracing

As suggested above, the learning assumptions of ACT-R are complex. With practice, both declarative and procedural knowledge are strengthened in memory and student performance improves accordingly. Modeling these relationships on-line as students practice is computationally expensive and not warranted by the relatively sparse data the tutor provides. Instead, we have substituted a simpler set of learning and performance assumptions in knowledge tracing in our tutors.

Knowledge tracing assumes a simple two-state learning model. Each production rule is either in the learned state or in the unlearned state. A rule can make the transition from the unlearned to the learned state prior to practice or at each opportunity to apply the rule in practice. Further, there is no forgetting; rules do not make the transition in the other direction. Performance in applying a rule is governed by its learning state, but only probabilistically. If a rule is in the learned state, the student may nevertheless slip and make a mistake. If the rule is in the unlearned state, there is some chance the student will guess correctly. As the student practices, the tutor maintains an estimate of p(L) for each rule, the probability that the rule is in the learned state. At each opportunity to apply a rule in problem solving, the estimate of p(L) for the rule is updated, contingent on whether the student's action is correct or not. The Bayesian computational procedure is a variation on one described by Atkinson (1972). This procedure employs two learning parameters and performance parameters as displayed in Figure 2. These parameters are estimated empirically for each rule.

The following equation is used in knowledge tracing to update the estimate of the student's knowledge state:

$$p(L_n) = p(L_{n-1} | \text{evidence}) + (1 - p(L_{n-1} | \text{evidence})) * p(T) \quad (1)$$

The probability that a rule is in the learned state following the nth opportunity to apply the rule, $p(L_n)$, is the sum of two probabilities: (1) the posterior probability that the ideal rule was already in the learned state contingent on the evidence (whether or not the nth action is correct) and (2) the probability that the rule will make the transition to the learned state if it is not already there. We use a Bayesian inference scheme to estimate the posterior probability that the rule is already in the learned state $p(L_{n-1} | \text{evidence})$. Following Atkinson (1972) the probability p(T) of a transition from the unlearned to the learned state during procedural practice is independent of whether the student applies the rule correctly or incorrectly.

Individual differences among students are also incorporated into the model in the form of four weights, one for each of the four parameter types, wL_0, wT, wG and wS. When the model is adjusted for a student, each of the four probability parameters for each rule is converted to odds form (p/(1–p)), it is multiplied by the corresponding subject-specific weight, and the resulting odds are converted back to a probability. A best fitting set of weights for each subject is generated with a

curve-fitting program for research purposes. In the tutor, these weights are estimated dynamically by means of regression equations based on raw error rates.

Knowledge tracing is employed in the tutor to implement mastery learning. In each section of the tutor curriculum the student reads an accompanying text that introduces a set of coding rules. The tutor follows with a set of exercises that provide practice on the rules. The student continues practicing exercises in a section until reaching a criterion knowledge probability for each rule in the set. That mastery criterion in the tutor is a knowledge probability of 0.975.

$p(L_0)$	Initial Learning	The probability that a rule is in the learned state prior to the first opportunity to apply the rule
p(T)	Acquisition	The probability that a rule will move from the unlearned to the learned state at each opportunity to apply the rule
p(G)	Guess	The probability that a student will guess correctly if a rule is in the unlearned state
p(S)	Slip	The probability that a student will slip (make a mistake) if a rule is in the learned state

Figure 2. The learning and performance parameters in knowledge tracing.

2.1 Empirical Evaluation of Knowledge Tracing

Knowledge tracing has been shown to be an effective learning tool. Students perform better on tests with knowledge tracing and remediation than in a condition in which students work through a fixed set of required problems in the tutor (Anderson et al., 1989; Corbett and Anderson, 1995a). While the most capable students may require no additional remedial problems to perform well in the tutor and on tests, other students may complete three or four times the number of required problems in remediation.

In addition, we have completed two assessments of the predictive validity of knowledge tracing (Corbett and Anderson, 1995b). The two studies had similar outcomes; the results of the second study are displayed in Table 1. As can be seen, knowledge tracing is reasonably accurate at predicting average performance on tests and moderately sensitive to individual differences. The correlation between actual and expected performance is marginally reliable for the second test and reliable and quite strong for the third test. The tests are cumulative. In the third test in this study, 56% of students in the knowledge tracing condition achieved test scores of about 90% correct or better, compared to 24% in a comparison condition in which students completed a minimum set of required problems.

Overestimating students' test performance. A consistent result in these two studies is that the knowledge tracing process tends to overestimate student performance. In Table 1 performance is overestimated by an average of about 8%. It is important to understand the nature of this overesti-

mation in progressing toward our goal of enabling all students to reach mastery. There are at least two possible explanations for the model's overestimation: retention and transfer.

Table 1. Actual and expected proportion of exercises completed correctly across students in each of the three tests (Corbett and Anderson, 1995b).

	Mean Proportion Correct		
	Actual	Expected	Correlation[a]
Test 1	0.88	0.94	0.24
Test 2	0.81	0.89	0.36
Test 3	0.81	0.86	0.66

[a] Correlation between actual and expected proportion correct across students.

Since the tests in these studies are cumulative, the first possibility is that students fail to meet the tutor's expectations because they are forgetting material from earlier curriculum sections. The first test follows the first curriculum section, the second test follows the fourth curriculum section and the third test follows the fifth curriculum section. By the time they take the second and third tests, students may have forgotten material from the first few sections.

The second possibility is that students are acquiring suboptimal rules in the tutor environment that are sufficient to perform adequately in that environment but do not fully generalize to the test environment. For example, students may acquire rules that are sufficient when domain knowledge is partitioned into curriculum sections as in the tutor but insufficient when the partitioning is eliminated in the cumulative tests. Students may even acquire rules that hinge on specifics of the tutor interface, e.g., "use the operator that is not yet checked off in the skill meter". Students who have not adequately learned prerequisite declarative knowledge will be most vulnerable to forming such suboptimal rules. In a recent study we collected data on students' declarative knowledge that enable us to examine this possibility.

3 Declarative Knowledge Measures

As described earlier, students read text in each curriculum section before completing programming problems. We recently administered a series of tests to students that were designed to assess students' knowledge of the factual material in the text and their ability to reason with that knowledge in problem solving. One or two tests were administered after students read the text and before they completed tutor exercises in each of the first five Lisp curriculum sections. These tests tapped students': (1) factual knowledge of Lisp operators, (2) factual knowledge of list structure, (3) ability to evaluate Lisp code (at two points in the curriculum) and (4) judgments of programming problem similarity (again at two points). A factor analysis performed on these tests revealed two underlying factors, as shown in Table 2. We proposed that Factor 2 reflects declarative knowledge of the text, since tests of basic Lisp operators and list structure load heavily on this

factor, and that Factor 1 reflects students' ability to bring declarative knowledge to bear appropriately in problem solving, because of the heavy loading of students' problem similarity ratings. (See Corbett and Knapp, 1996, for more details on the declarative measures and the factor analysis.) The correlations displayed in Table 3 are consistent with this analysis. In this table we have correlated students' scores on the two declarative knowledge factors with their best fitting individual difference weights for the two learning parameters in the tutor's procedural learning model. These best fitting individual difference weights essentially reflect students' error rates in completing tutor exercises. As can be seen, scores on the Factual Knowledge Factor are highly correlated with $\ln(wL_0)$, a measure of how well the student has acquired rules prior to the first practice opportunities. In contrast, the Reasoning Factor scores are highly correlated with ln(wT), a measure of how readily students acquire rules in the course of procedural practice. In this report we employ these declarative knowledge factors to evaluate the knowledge tracing and improve predictions of test performance.

4 The Study

Sixteen students in this study worked through the first five curriculum sections in the APT Lisp curriculum. This curriculum introduces two data structures, *atoms* and *lists,* and introduces *function calls*. The first section introduces three extractor functions, *car*, *cdr* and *reverse*. The second and third sections introduce three constructor functions, *append*, *cons* and *list*. The fourth section introduces extractor algorithms—nested function calls that apply successive extractor functions to extract components of lists. In the fifth section extractors and extractor algorithms are embedded as arguments to constructor function calls. These five curriculum sections contain 30 required tutor exercises.

Table 2. Loadings of six declarative knowledge measures on two factors that emerge in a factor analysis (Corbett and Knapp, 1996).

Declarative Test	Factor 1 Reasoning	Factor 2 Factual Knowledge
Lisp Operator Facts		0.85
List Structure Facts		0.77
Extractor Evaluation 1	0.49	0.54
Extractor Evaluation 2	0.54	0.56
Problem Similarity 1	0.75	
Problem Similarity 2	0.88	

Table 3. Correlation of factor scores and individual weights across students in the study (Corbett and Knapp, 1996).

Parameter weight		Factor 1 Reasoning	Factor 2 Factual Knowledge
$\ln(wL_0)$	(learning prior to problem solving)	0.20	0.65
$\ln(wT)$	(learning during problem solving)	0.52	0.34

In each curriculum section, students read text describing Lisp, completed one or two sets of questions on the text as described above, then completed a set of required programming exercises that covered the rules being introduced. Students then completed remedial exercises as needed to bring all production rules in the section to a mastery criterion (knowledge probability > .975). Finally, students completed cumulative programming tests following the first, fourth and fifth sections. These tests contained 6, 12 and 18 programming exercises, respectively. The cumulative test exercises were similar to the tutor exercises and the test interface was identical to the tutor interface, except that students could freely edit their code and received no tutorial assistance.

4.1 Results

Table 4 displays students' actual performance on the three tests in the study and the tutor's predictions of test performance on the basis of the knowledge tracing model. The correlation of actual and expected performance across students is reliable only for the third test, $r = 0.57$, $t(14) = 2.58$, $p < .05$. As can be seen, the model slightly overpredicts student performance for the more difficult second and third tests, as in earlier studies.

Test performance overestimation and forgetting. If the systematic tendency to overestimate quiz performance reflected forgetting between learning and test, we would expect the discrepancy between actual and expected scores to be monotonically related to the curriculum section number. We would expect the model's overestimation to increase as the interval between learning and test increases, so the overestimation should be greatest for the first curriculum section on each test. The model's overestimation should then systematically decrease on each test as we move forward through the curriculum sections. Table 5 displays actual and expected performance for problems grouped by curriculum section in Tests 2 and 3. As can be seen, the overprediction is smallest, rather than largest, for the first curriculum section, and it peaks in Curriculum Section 3, which contains the most challenging constructor problems. Consequently, forgetting is not a compelling explanation of the model's overestimation of test performance.

Table 4. Actual and expected proportion of exercises completed correctly in three tests.

	Mean proportion correct		
	Actual	Expected	Correlation[a]
Test 1	0.98	0.97	–0.28
Test 2	0.90	0.93	0.34
Test 3	0.85	0.90	0.57

[a] Correlation between actual and expected proportion correct across students.

Table 5. Actual and expected proportion correct as a function of tutor curriculum section.

Curriculum section	Test 2			Test 3		
	Actual	Expected	A–E[a]	Actual	Expected	A–E[a]
1	0.91	0.93	–0.02	0.89	0.93	–0.04
2	0.79	0.90	–0.11	0.76	0.90	–0.14
3	0.65	0.91	–0.26	0.65	0.89	–0.24
4	0.79	0.84	–0.05	0.76	0.90	–0.14
5	-	-	-	0.63	0.73	–0.10

[a] Actual minus expected scores.

Test performance overestimation and declarative knowledge. Table 6 displays the correlations between students' scores on the declarative knowledge factors derived from the battery of six declarative knowledge tests and (1) students' actual performance on the test and (2) the model's overestimation of students' test performance based on the knowledge tracing model. Not surprisingly, both declarative knowledge factors are strongly correlated with students' actual performance on both tests. All four of these correlations are statistically significant. More interestingly, students' declarative knowledge is correlated with the extent to which the tutor's knowledge tracing model of procedural learning overestimates their performance on the test. In particular, the tutor's overestimation of performance is strongly related to students' basic factual knowledge, although not to their ability to reason about knowledge in the problem solving context. The lower a student's factor score on basic factual knowledge, the more the tutor overestimates how well the student will perform on the test. Both of these correlations are statistically significant: For Test 2: $r = –0.63$, $t(14) = 3.06$, $p < .01$; for Test 3: $r = –0.64$, $t(14) = 3.14$, $p < .01$.

Table 6. The correlations of students' scores on the two declarative knowledge factors with their actual performance on Tests 2 and 3 and with the model's overestimation of their performance on these two tests.

	Test 2		Test 3	
Factor	Actual Performance	Model Overestimate	Actual Performance	Model Overestimate
Reasoning in Problem Solving	0.48	–0.11	0.54	–0.13
Factual Knowledge	0.67	–0.63	0.61	–0.64

5 Integrating Declarative Knowledge into Knowledge Tracing

We implemented a simple three-state model of learning to obtain an initial quantitative estimate of the impact of individual differences in factual knowledge on knowledge tracing. As in the existing model we assume that the student may learn no rule, or may learn an "ideal" rule that is sufficient for both tutor and test performance. The third possibility in this model is that the student may learn a suboptimal rule that is sufficient for the tutor but not in the test environment. Such a suboptimal rule will be indistinguishable from an ideal rule in modeling the student's tutor performance and be indistinguishable from no rule in predicting test performance. To predict test performance for student *s* in applying rule *r* on the test in this model, we simply adjusted the tutor's estimate that the student has learned the ideal rule, $p(L_{sr})$, with an additive term proportional to the student's Factual Knowledge factor score, FK_s. We assumed this proportion was constant across subjects and rules, so the probability that student *s* had learned an ideal version of rule *r* that would successfully generalize to the quiz became $p(L_{sr}) + (0.12 * FK_s)$.

We used this model to recompute predictions of students' test performance as displayed in Table 7. Given the correlations in Table 6, this model is certain to improve the quality of fit and, as can be seen in the table, (1) the tendency to overestimate test performance is essentially eliminated and (2) the model is now more sensitive to individual differences. The 0.81 correlation between actual and expected performance for Test 3 is reliably greater than the corresponding correlation of the earlier model (0.57 in Table 4), and this difference is marginally significant, $t(13) = 2.145$, $p < .06$. The difference between the 0.55 correlation for Test 2 in the new model and the earlier model (0.34 in Table 4) does not reach significance, however, $t(14) = 1.29$. These results suggest that incorporating factual knowledge can substantially improve the predictive validity of knowledge tracing, although it should be noted that these results represent an upper bound on the contribution of factual knowledge, since we are refitting the existing data with a best fitting estimate of the factual knowledge constant.

Table 7. Actual and expected proportion of exercises completed correctly across students in Tests 2 and 3 when declarative knowledge is integrated into the knowledge tracing model.

	Mean Proportion Correct		
	Actual	Expected	Correlation[a]
Test 2	0.90	0.91	0.55
Test 3	0.85	0.85	0.81

[a] Correlation between actual and expected proportion correct across students.

6 Discussion

The slight but consistent tendency of the knowledge tracing model to overestimate student performance on tests is shown not to reflect student forgetting but rather to be predicted by the quality of students' declarative knowledge. The lower a student's score on a basic declarative knowledge factor, the greater the tendency of the tutor to overestimate the student's performance. A simple assumption that integrates declarative knowledge into the knowledge tracing model of procedural learning eliminates the model's tendency to overestimate test performance and enhances the model's sensitivity to individual differences in test scores.

The most important implication of these results concerns the path to mastery learning for all students: While knowledge tracing-based remediation in the tutor successfully raises test scores, additional problem solving of exactly the same type in the tutor will probably not yield the achievement gains sufficient for all students to reach mastery. Note that if the model's overestimation had reflected forgetting, then more practice of the same type would be just what we might prescribe. But if a student learns suboptimal rules in the tutor that capitalize on accidental characteristics of the tutor curriculum and/or interface and are more or less sufficient for success in the tutor, then there is no guarantee that additional practice of the same type will affect the rules. Instead, we need to develop interventions before and/or during tutor practice to help students form more suitable rules.

Important tasks remain before these results can be translated into effective practice. First, we need to develop measures of declarative knowledge that can be dynamically incorporated into knowledge tracing to model the student's cognitive state during the learning process. Second, we need to develop effective interventions to improve prerequisite declarative knowledge and to scaffold the use of that knowledge in problem solving. Nevertheless, the results of this study bring us an important step closer to the goal of realizing the promise of mastery learning for all students.

References

Anderson, J. R., (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R., Conrad F. G., and Corbett, A. T. (1989). Skill acquisition and the LISP Tutor. *Cognitive Science*, 13:467–505.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences* 4:167–207.

Atkinson, R. C., (1972). Optimizing the learning of a second-language vocabulary. *Journal of Experimental Psychology* 96:124–129.

Bloom, B. S, (1968). Learning for mastery. In *Evaluation Comment, 1*. Los Angeles: UCLA Center for the Study of Evaluation of Instructional Programs.

Carroll, J. B. (1963). A model of school learning. *Teachers College Record* 64:723–733.

Corbett, A. T., and Anderson, J. R., and O'Brien, A. T. (1995). Student modeling in the ACT Programming Tutor. In Nichols, P., Chipman, S., and Brennan, B., eds., *Cognitively Diagnostic Assessment*. Hillsdale, NJ: Erlbaum.

Corbett, A. T., and Anderson, J. R. (1995a). Knowledge decomposition and subgoal reification in the ACT Programming Tutor. *Artificial Intelligence and Education 1995: The Proceedings of AI-ED 95*. Charlottesville, VA: AACE.

Corbett, A. T., and Anderson, J. R. (1995b). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4:253–278.

Corbett, A. T., and Knapp, S., (1996). Plan scaffolding: Impact on the process and product of learning. In Frasson, C., Gauthier, G., and Lesgold, A., eds., *Intelligent Tutoring Systems: Third International Conference, ITS '96*. New York: Springer.

Goldstein, I. P., (1982). The genetic graph: A representation for the evolution of procedural knowledge. In Sleeman, D., and Brown, J.S., eds., *Intelligent Tutoring Systems.* New York: Academic.

Keller, F. S. (1968) "Good-bye teacher...." *Journal of Applied Behavioral Analysis* 1:79–89.

Kulik, C. C., Kulik, J. A., and Bangert-Drowns, R. L. (1990). Effectiveness of mastery learning programs: A meta- analysis. *Review of Educational Research* 60:265–299.

Resnick, L. B. (1977). Assuming that everyone can learn everything, will some learn less? *School Review* 85:445–452.

Slavin, R. E., (1987). Mastery learning reconsidered. *Review of Educational Research* 57:175–213.