User Modeling and User-Adapted Interaction **10:** 81–108, 2000.

Modeling Student Knowledge: Cognitive Tutors in High School and College

ALBERT CORBETT, MEGAN MCLAUGHLIN
and K. CHRISTINE SCARPINATTO
Human Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, E-mail: corbett+@cmu.edu

(Received 16 November 1999; in final form 30 June 2000)

Abstract. This paper examines the role of adaptive student modeling in cognitive tutor research and dissemination. Cognitive tutors™ are problem solving environments constructed around cognitive models of the knowledge students are acquiring. Over the past decade we in the Pittsburgh Advanced Cognitive Tutor (PACT) Center at Carnegie Mellon have been employing a cognitive programming tutor in university-based teaching and research, while simultaneously developing cognitive mathematics tutors that are currently in use in about 150 schools in 14 states. This paper examines adaptive student modeling issues in these two contexts. We examine the role of student modeling in making the transition from the research lab to widespread classroom use, describe our university-based efforts to empirically validate student modeling in the ACT Programming Tutor, and conclude with a description of the key role that student modeling plays in formative evaluations of the Cognitive Algebra II Tutor.

Key words: student modeling, adaptivity, intelligent tutoring systems, mastery learning, empirical validation

Intelligent tutoring systems appeared on the scene as interesting artificial intelligence challenges and promising educational environments in the late 1970s (Sleeman and Brown, 1982). Our research lab began developing a type of intelligent tutor we call *cognitive tutors* for programming and mathematics in the early 1980s for use both as psychological research tools and as teaching tools. The initial motivation was to evaluate and develop Anderson's (1983) ACT*[1] cognitive theory, a unified theory of the nature, acquisition and use of human knowledge. Over the following decade this cognitive tutor research served both to validate the ACT* production-rule model of problem solving knowledge (Anderson, Conrad and Corbett, 1989; Anderson, Corbett, Koedinger and Pelletier; 1995), and to refine the learning assumptions in the more recent ACT-R theory

[1] As the theory has evolved over the past quarter century, ACT has variously expanded as the Adaptive Control of Thought, the Adaptive Character of Thought and currently as the Atomic Components of Thought.

(Anderson, 1990; 1993; Anderson and Lebiere, 1998). At the same time cognitive tutors proved to be effective learning environments. Students working with cognitive tutors completed problem solving activities in as little as 1/3 the time needed by students working in conventional problem solving environments and performed as much as a letter grade better on post-tests than students who completed standard problem solving activities (Anderson et al. 1995). Successive generations of the ACT Programming Tutor have been used to teach self-paced programming courses each semester at Carnegie Mellon since 1984 and successive generations of geometry proof tutors were successfully piloted in closely monitored field studies in the Pittsburgh Public Schools (Anderson, Boyle and Yost, 1986; Koedinger and Anderson, 1993).

Over the past decade, our programming and mathematics tutors have supported divergent research agenda. The ACT Programming Tutor has served as the primary vehicle in controlled laboratory research that investigates issues in feedback timing, transfer and student modeling. In contrast, the mathematics tutors have served as the vehicles to explore an important educational issue: What is required for a complex educational technology to make the transition from closely-monitored university research projects and in-house teaching to widespread use in real-world classrooms? This dissemination effort has been bearing fruit and in the 1999–2000 academic year our Cognitive Tutor Algebra and Geometry courses are in use in about 150 schools in 14 states.

This paper examines the relations between adaptive student modeling and the transition from research lab to real-world deployment. The first two sections of this paper introduce cognitive tutor technology. In Section 1 we briefly describe two tutors, the APT Lisp Tutor which has been employed extensively in university research and teaching and the Cognitive Algebra II Tutor which was developed for widespread classroom use. Section 2 describes the common architecture and modeling assumptions that underlie these tutors. The two sections that follow focus on the pragmatics of real-world deployment. Section 3 summarizes five essential factors that enabled us to bring cognitive tutor technology out of the research laboratory and into the classroom. Success in these dissemination efforts depended on the learning and motivational benefits of adaptive student modeling, but depended equally on making the technology more "useable" by embedding it in full courses that address important educational needs and by providing professional development support for teachers. Section 4 summarizes differences we observed between cognitive tutor use in the university and in high schools and describes the relatively minor tuning of student modeling that was needed to accommodate the high school setting. The final two sections characterize the research role of student modeling in the differing settings. As described in Section 5, our basic research in the university has largely focused on validating our student modeling assumptions. In contrast, student modeling has primarily served as a formative evaluation in our high school classroom research, as described in Section 6.

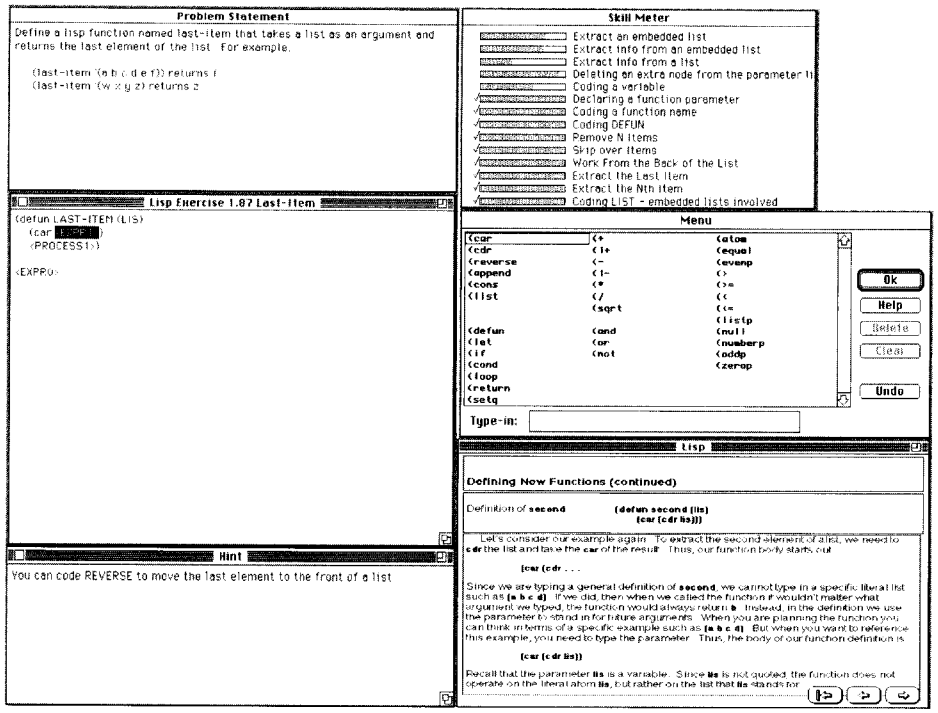

Figure 1. The APT Lisp Tutor interface.

1. Cognitive Tutor Technology

Cognitive tutors are fundamentally problem solving environments. Figure 1 displays the ACT Programming Tutor (APT) Lisp Module midway through a typical programming problem. The student has previously read text presented in the window at the lower right and is completing a sequence of corresponding programming problems. The problem description appears in the upper left window and the student's solution appears in the code window immediately below. The interface is similar to a structure editor; the student selects Lisp 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* used to define a new operator, has entered the operator name, declared an input variable and begun coding the body of the program definition. The three angle-bracket symbols in the figure, $< EXPRI >$, $< PROCESSI >$ and $< EXPRO >$ serve as both syntax nodes and potential subgoal reminders. The student will either replace each node with additional Lisp code or delete it.

The APT Lisp Tutor, like all cognitive tutors, is built around a cognitive model of the problem solving knowledge students are acquiring. This cognitive model is an expert system that can solve the same problems students are asked to solve and

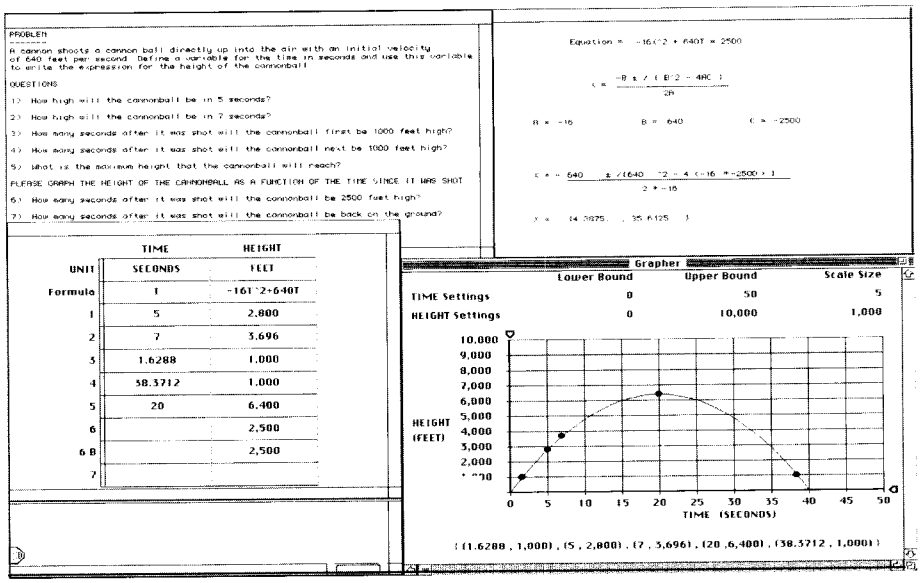

Figure 2. The Cognitive Algebra II Tutor interface near the end of a vertical motion problem.

in the same ways that students solve them. The cognitive model enables the tutor to trace each student's individual problem solving path in a process we call model tracing, providing step-by-step help as needed. The tutor provides feedback on each problem solving action, by accepting correct actions and flagging errors instead of accepting them. The tutor also provides problem solving advice in the lower left window upon request of the student. There are generally three levels of advice available for each problem solving action. The first level reminds or advises the student on an appropriate goal. The second level provides general advice on solving the goal. Finally, the third level provides concrete advice on solving the goal in the current context.

As the student works, the tutor also models the student's knowledge of the component rules in the underlying cognitive model and employs this model to individualize the student's path through the course curriculum. This student model is displayed in the Skill Meter in the upper right corner of Figure 1. Each bar in this window represents a component rule in the cognitive model and the shading represents the probability that the student knows the rule. The checkmarks indicate rules that the student has "mastered." (Linton et al. 2000, describe a related skill meter that depicts student knowledge of text editing commands).

Figure 2 displays the interface for a vertical motion lesson in the Cognitive Algebra II Tutor, near the conclusion of a problem. The problem statement window in the upper left corner describes a problem situation, as in the program-

ming tutor, and in this example poses seven questions. Students answer the questions by filling in the worksheet in the lower left corner. Students (1) identify relevant quantities in the problem and label the columns accordingly; (2) enter appropriate units in the first row of the worksheet; (3) enter a symbolic formula for each quantity in the second row; and (4) answer the questions in the successive table rows. Students graph the function with the graphing tool in the lower right corner. The quadratic formula tool in the upper right is available for use in answering questions, and is one example of a suite of symbol manipulation tools in the tutor. Again, this tutor is constructed around a cognitive model of algebra problem solving knowledge and provides both feedback on problem solving actions and problem solving advice upon request of the student. This tutor also displays its model of the student's knowledge state in a Skill Meter (not visible in the Figure 2 screen configuration) and employs the model to tailor the problem sequence for each student.

2. Student Modeling Assumptions

The cognitive model underlying each cognitive tutor reflects 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 on equation solving in an algebra text would be encoded declaratively:

> When the quantities on both sides of an equation are divided by the same value, the resulting quantities are equal.
>
> For example, if we assume $2X = 12$, then we can divide both sides of the equation by 2, and the two resulting expressions will be equal, $X = 6$.

ACT-R assumes that skill knowledge is encoded initially in declarative form through experiences such as reading and that early in practice the student solves problems by applying general procedural rules to the domain-specific knowledge. As a consequence of this early activity, domain-specific procedural knowledge is acquired and with practice, both declarative and procedural knowledge are strengthened so that performance grows more rapid and reliable. Like many cognitive theories (cf. Kieras and Bovair, 1986; Newell, 1990; Reed et al., 1985) 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 goal-oriented production can be derived from the declarative example above through practice in solving equations:

> IF the goal is to solve an equation for variable X and the equation is of the form $aX = b$,
>
> THEN divide both sides of the equation by a to isolate X.

2.1. ADAPTIVITY: MODEL TRACING

Each cognitive tutor employs its embedded cognitive model for two purposes, which we call *model tracing* and *knowledge tracing*. In model tracing the goal is to follow each student's individual solution path through a complex problem space that can contain many successful paths. In each problem solving cycle the student selects an interface element to work on (e.g. a worksheet cell in the mathematics tutors or a syntax node in the programming tutors), and performs a problem solving action (e.g. typing a numeric expression or a programming operator name). Each interface element is linked to an internal representation of a problem solving goal with further links to relevant information in the current problem solving state. The tutor applies its production rules to the goal the student implicitly selects and generates a set of one or more applicable rules that satisfy the goal. The student's action is compared to the actions this set of applicable rules would generate. If the student action matches a production rule action it is assumed that the student has fired the same cognitive rule and the actions are carried out. If not, the tutor reports that it does not recognize the student's action.

Tracing the student's step-by-step solution enables the tutor to provide individualized instruction in the problem solving context. Prototypically our tutors provide immediate feedback on each problem solving action: recognizably correct actions are accepted and unrecognized actions are rejected. Our tutors do not generally try to diagnose student misconceptions and do not automatically give problem solving advice. Instead, they allow the student maximum opportunity to reason about the current problem state. The tutors do provide a feedback message if the student appears confused about the nature of the current problem state or a problem solving action. For example, the Cognitive Algebra II Tutor will alert a student who enters the right answer for one worksheet cell in an adjacent cell. The tutors provide goal-directed problem solving advice upon request. As described earlier, the tutors provide three general levels of advice: a reminder of the current goal, a general description of how to achieve the goal, and a description of exactly what problem solving action to take. Each of these three levels may be represented by multiple messages.

2.2. ADAPTIVITY: KNOWLEDGE TRACING

The goal in *knowledge tracing* is to draw inferences about the student's growing knowledge of the component rules in the cognitive model. Our cognitive tutors employ a simple learning model for this purpose, in which each production rule can only be in one of two states; it has either been learned by the student or remains unlearned. A rule can make the transition from the unlearned to the learned state prior to problem solving in the tutor (e.g. through other classroom activities) or at each opportunity to apply the rule in problem solving. The model does not incorporate forgetting; rules do not make the transition in the other direction. Perform-

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

Figure 3. The learning and performance parameters in knowledge tracing.

ance in applying a rule is governed by its learning state, but only probabilistically. If a rule is in the learned state, there is some chance 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 employs two learning parameters and two performance parameters, as displayed in Figure 3. At the nth opportunity to apply a rule, the following equations are employed to update the probability that the rule is in the learned state, contingent on whether the student performs a correct problem solving action (C) or commits an error (E):

$$p(L_n \mid C_n) = p(L_{n-1} \mid C_n) + (1 - p(L_{n-1} \mid C_n)) * p(T) \quad (1)$$

$$p(L_n \mid E_n) = p(L_{n-1} \mid E_n) + (1 - p(L_{n-1} \mid E_n)) * p(T) \quad (2)$$

That is, 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 rule was already in the learned state contingent on the evidence (whether the nth action is correct or an error); and (2) the probability that the rule will make the transition to the learned state if it is not already there. The posterior probabilities $p(L_{n-1} \mid C_n)$ and $p(L_{n-1} \mid E_n)$ expand as Bayes theorem, as shown here:

$$p(L_{n-1} \mid C_n) = p(L_{n-1}) * p(C \mid L)/(p(L_{n-1}) * p(C \mid L) + p(U_{n-1}) * p(C \mid U)) \quad (3)$$

$$p(L_{n-1} \mid E_n) = p(L_{n-1}) * p(E \mid L)/(p(L_{n-1}) * p(E \mid L) + p(U_{n-1}) * p(E \mid U)) \quad (4)$$

This computational scheme is a variation of one described by Atkinson (1972) and following Atkinson, 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.

In knowledge tracing the tutor is seeded with a best fitting set of four parameter estimates (two learning parameters and two performance parameters) for each of the problem solving rules. These estimates are derived in advance by fitting the knowledge tracing assumptions to a full set of tutor performance data for a representative group of students. The model's learning and performance assumptions enable us to predict the probability that a student will correctly apply an appropriate rule at each step in problem solving with Equation 5:[2]

$$p(C_{is}) = p(L_{rs}) * (1 - p(S_r)) + (1 - p(L_{rs})) * p(G_r) \tag{5}$$

That is, the probability that a student s will perform a correct problem solving action on step i, $p(C_{is})$, is the sum of two products: (1) the probability that an appropriate rule r is in the learned state for student s times the probability of a correct response if the rule is in the learned state ($p(S_r)$ is the slip parameter in Figure 3); and (2) the probability that an appropriate rule r is not in the learned state for student s times the probability of a correct guess if the rule is not in the learned state ($p(G_r)$ is the guess parameter in Figure 3). We derive best fitting parameter estimates for each rule by fitting the model's learning and performance equations to the sequence of correct and incorrect applications of the rule by each student in a representative group that has worked through the tutor curriculum.

Individual differences among students are also dynamically incorporated into the model in the form of four weights, one for each of the four parameter types, wL_0, wT, wG and wS. In adjusting the model for a student, each of the four probability parameters for each rule is converted to odds form ($p/(1-p)$), multiplied by the corresponding subject-specific weight, and the resulting odds converted back to a probability. These weights are re-estimated dynamically in each curriculum section by means of regression equations that relate raw error rate to best-fitting weights for a representative group of students. Each section begins with a small fixed set of required problems. When the student completes the required problems in each section, the regression equations are used to estimate the individual difference weights from the student's cumulative total errors across all required problem sets in the lesson. These revised weights are employed to revise the learning parameter estimates for all rules before remediation begins.

2.3. Cognitive mastery learning

The goal of student modeling and adaptivity in cognitive tutors is to promote efficient learning and to foster change in the student's (knowledge) state, more than to accommodate student preferences and promote efficient performance relative to the student's state. In this regard, our objectives are similar to OWL (Linton

[2] This equation assumes the student is always on a recognizable solution path, which is essentially true, since the model tracing process does not accept incorrect problem solving actions.

et al., 2000) and P-TIMS (Strachan, Anderson, Sneesby and Evan, 2000) (and less related to the efforts described in Billsus and Pazzani, 2000, and Fink and Kobsa, 2000). Unlike OWL and P-TIMS, however, cognitive tutors are primarily learning environments constructed around defined problem solving tasks and cognitive models, so they can interpret each student action (model tracing) and draw inferences about the student's knowledge state from each student action (knowledge tracing). Model tracing promotes learning by providing problem solving advice customized to the student's problem solving state. Knowledge tracing is employed in the tutors to promote cognitive mastery learning. In cognitive mastery learning, each tutor curriculum section introduces a set of cognitive rules and the tutor provides a set of problems that draw upon those rules. The sequence of problems in each section is tailored to the student's needs and the student continues solving problems in a section until reaching a criterion probability of knowing each one of the rules in the set. That mastery criterion in the tutor is a knowledge probability of 0.95. This procedure is similar to individualized student-paced mastery learning systems based on Keller's Personalized System of Instruction, which have been shown to be effective in raising mean achievement scores (Block and Burns, 1977; Kulik, Kulik and Bangert-Drowns, 1990; Kulik, Kulik and Cohen, 1979). In these systems, however, practice and assessment are usually treated as distinct phases. Students work through study/test cycles until obtaining a criterion test performance level. In knowledge tracing, by contrast, assessment is continuously integrated with practice; students simply continue problem solving until reaching a hypothetical knowledge state. Tests can be used to validate the tutor's mastery decisions but are not employed in the tutor to make mastery decisions.

3. Disseminating Cognitive Tutors in the Real World

We have developed three cognitive mathematics tutors that are currently in use in real-world mathematics classrooms, beginning with a Cognitive Algebra I Tutor (Koedinger, Anderson, Hadley and Mark, 1997), and more recently a Cognitive Geometry Tutor (Koedinger and Cross, 1999) and the Cognitive Algebra II Tutor (Corbett, Trask, Scarpinatto and Hadley, 1998). Cognitive Tutor Algebra and Geometry courses that employ these tutors are being taught in about 150 schools in 14 states this year, including public and private schools, middle and high schools and urban, suburban and rural schools.

How did adaptive student modeling contribute to the success of this dissemination effort? We have identified five factors that were critical in the successful transition from the research lab to widespread mathematics classroom (Corbett, Koedinger and Anderson, 1999; Corbett, Koedinger and Hadley, in press):

- Opportunity: Addressing an Important Need
- Usability: Integrated Courses and Technology

- Usability: Professional Development and Support
- Effectiveness: Achievement Gains
- Effectiveness: Classroom Impact

Note that while "scaling up" issues can pose a formidable barrier in real-world deployment of user modeling and user-adapted systems (Billsus and Pazzani, 2000; Linton, et al., 2000), they did not pose a challenge in the transition of cognitive tutors to real-world classrooms. This is largely because model tracing and knowledge tracing computations are relatively fast on modern workstations and users are more tolerant of any delays that do occur in a learning environment. In fact, the first three factors in the list are not directly related to student modeling at all. Instead, they are concerned with integrating cognitive tutor technology into an effective and useable educational package that addresses an important educational goal. In contrast, the last two issues concerning demonstrable effectiveness depend directly on model tracing and knowledge tracing. In the remainder of this section, we expand briefly on each of these topics. In the following section, we discuss the relatively minor tuning of adaptive student modeling that was prompted by this transition from the lab to the classroom.

3.1. Opportunity: Addressing an Important Need

Teachers and administrators are actively looking for new solutions in K-12 mathematics for three related reasons: (a) American high school students lag behind the rest of the developed world in high school mathematics achievement (U.S. Department of Education, 1998); (b) the National Council of Teachers of Mathematics (NTCM, 1989) has recommended far-reaching reforms in high school mathematics curriculum, teaching and assessment; and (c) at least 49 states have or are defining statewide standards and assessments that increasingly are employed to evaluate schools and govern student graduation.

3.2. Usability: Integrated Courses and Technology

Rather than developing cognitive tutors as supplemental activities that teachers need to integrate into their classes, we developed full Algebra and Geometry courses with the respective cognitive tutors as essential components. These courses provide a paper curriculum (combination text and workbook), a teacher's edition of the curriculum, student (homework) assignments, assessments and a cognitive tutor user's guide. Typically, three class periods per week employ small-group problem solving activities and whole-class instruction to introduce topics. Students spend the remaining two class periods a week working with the cognitive tutors in applying these constructs in individual problem solving activities.

3.3. USABILITY: PROFESSIONAL DEVELOPMENT AND SUPPORT

All new teachers complete multi-day pre-service training sessions that cover (a) NCTM curriculum, teaching, and assessment standards addressed by the courses; (b) management of small group problem solving; (c) cognitive tutor technology; and (d) effective teacher activities in the cognitive tutor lab. In addition, we install the cognitive tutor software on-site, and provide both telephone and email hotline support for both curriculum and technical questions.

3.4. EFFECTIVENESS: ACHIEVEMENT GAINS

Assessing the effectiveness of our cognitive tutor mathematics courses poses an interesting challenge, because these courses have different objectives than traditional academic mathematics courses. The cognitive tutor courses place a greater emphasis on the application of algebra and geometry to authentic problem situations and reasoning among multiple representations (tables, graphs, symbolic expressions, natural language), while traditional courses place a greater emphasis on formal computation. To evaluate the effectiveness of the cognitive tutor courses, we administer multiple tests that assess both the new curriculum standards targeted in our cognitive tutor classes and traditional standards targeted in comparison academic mathematics classes. In the first assessment of Cognitive Tutor Algebra I in 1993–1994 (Koedinger et al., 1997) Cognitive Tutor Algebra I students scored twice as high on new-standards assessments of Algebra problem solving and reasoning among multiple representations and 10% higher on standardized test assessments of traditional Algebra goals. These results were subsequently replicated in Pittsburgh and in Milwaukee.

Does adaptive student modeling contribute to this success? In these quasi-experimental summative evaluations of full courses, several factors vary between the cognitive tutor condition and the comparison condition, including course objectives and classroom teaching style. We can only associate the relative benefits in the cognitive tutor condition with the full course package, rather than with the technology specifically. However, we can observe that the magnitude of the relative achievement gains in the cognitive tutor courses is comparable with results we have obtained in direct experimental evaluations of model tracing in the programming and geometry proof tutors (Anderson et al., 1995) and of knowledge tracing in the programming tutor (Corbett and Anderson, 1995).

3.5. EFFECTIVENESS: CLASSROOM IMPACT

Janet Schofield (1995) conducted an extended field study evaluating the impact of cognitive tutors in a Pittsburgh mathematics classroom. She documented a transformation in the teacher–student relationship in which teachers serve more as collaborators in learning. Teachers focus more attention on students who are

having problems, compared to traditional whole-class instruction, and can engage in prolonged interactions with individual students while the other students in the classroom are making productive problem solving progress with the cognitive tutors. Schofield also documented enhanced student motivation. She observed that students arrive in class on time or early, log-in and continue working through the period and often after the bell.

These effects on the teacher–student relationship and on student motivation can be directly related to adaptive student modeling. Students can take more control of their own learning and teachers can focus on extended interactions with students who are struggling primarily because model tracing is giving most students just the help they need to make productive progress. Schofield notes that one source of student motivation in the cognitive tutor lab classroom is the competition to drive up the learning probability shadings in the Skill Meter and to be the first to have the skills checked off. We also believe that students are motivated because model tracing provides just the support they need to successfully complete challenging mathematics problems.

In summary, the successful transition of cognitive tutor technology from the research laboratory to widespread classroom use depended in part on the success of adaptive student modeling and in part on embedding the technology in a complete educational package that addresses an important need. This transition did not require any substantial revisions of student modeling nor of cognitive tutor technology more generally. In the following section we examine differences between the university research environment and high school classroom environment and the relatively minor tuning of student modeling that accompanied widespread dissemination.

4. Comparing Programming Tutor Research in the University and Mathematics Tutor Research in the Real-World

While the cognitive tutor architecture is robust across settings, a variety of differences can be identified between the use of the APT Lisp Tutor in research and teaching at Carnegie Mellon and the use of the Cognitive Mathematics Tutors in middle and high schools. Some of these differences have an impact on our use of student modeling in the two domains.

4.1. Student Populations

The college students who enroll in our self-paced programming course and participate in cognitive tutor research studies at Carnegie Mellon are academically very successful. For example, their average score on the Mathematics SAT I, a popular U.S. standardized college admission test, is 650, or about 1.5 standard deviations above the national mean. In contrast, the Algebra and Geometry cognitive tutor mathematics courses are employed with a wide range of mainstream and high-risk

student populations. The majority of students in our principal Pittsburgh high school research and development site do not attend four-year colleges and most do not take the Math SAT I. When we administer Mathematics SAT I questions, average scores are equivalent to about 400, or about 1 standard deviation below the national mean.

This difference in student populations does not affect research design. While test performance levels are lower in absolute terms in the high school mathematics research than in the college-level programming research, it is worth noting that the *learning gains* supported by cognitive tutors technology are similar in both domains (Anderson et al., 1995; Koedinger et al., 1997).

4.2. Time on Task in the Cognitive Tutor Classroom

One important difference in the way cognitive tutors are deployed at the college and high school levels concerns time on task. The APT Programming Tutor is employed in self-paced introductory programming courses. Students come in on their own schedule to read on-line text and complete programming problems. Students spend as much time as is required to reach cognitive mastery in completing the tutor lessons. Students in the Cognitive Tutor mathematics courses work with the cognitive tutors during their regular mathematics periods, typically two class periods out of five periods per week. Over the entire academic year students spend only about 25–45 hr total in cognitive computer activities.

Given the fixed time on task available in the cognitive tutor mathematics classrooms and the need for students to complete the majority of the curriculum, we employ a modified version of cognitive mastery in the mathematics tutors. The tutors do monitor students' growing knowledge of the component rules in the cognitive model and tailor the sequence of problems to each student's weaknesses. However, a maximum number of problems is set in each curriculum section. A student "graduates" from each section if he or she has mastered every one of the component skills, but is "promoted" after completing the maximum number of problems, even if all of the component skills have not been "mastered". For example, last year in our Pittsburgh High School research site, the graduation rate averaged 18% across sections. That is, an average of 18% of students mastered all the skills and "graduated" in each curriculum section.

4.3. Teacher Support and Help Message Content

Students working with the APT Programming Tutor at Carnegie Mellon work in an unsupervised laboratory. Under these circumstances, it is essential that the bottom-level help message for each problem solving step advise the student on an exact problem solving action that will succeed. In contrast, there is always a teacher present in the high school cognitive mathematics tutor labs and the help messages are designed to encourage effective teacher interactions with students. Multiple levels of help are available that advise the student on an appropriate goal

and provide progressively more specific advice on achieving the goal. However, the bottom-level help message often stops short of providing a concrete acceptable problem solving action, so that the student will ask the teacher for help instead of simply copying a literal action offered by the tutor. One of the main goals in pre-service teacher training is to encourage the teacher not to simply provide an answer in response to a student's help request but to engage the student in a discussion of the problem solving situation.

4.4. Assessments

There are several differences in achievement testing in the university-based research and high school mathematics projects concerning (a) experimental control; (b) comparison groups; and (c) test structure. In our university-based research, students work entirely independently. In our high school deployments, in contrast, students in each class are all in the computer lab simultaneously and students at adjacent workstations help each other to varying degrees. This help can be educationally effective, but can reduce the validity of knowledge tracing. In the university research we have tight control over comparison conditions. Generally we hold curriculum content constant and manipulate tutorial variables. For example, we may vary the content or timing of help messages for different groups of students working through the same sequence of programming problems. In our high school mathematics dissemination project, course content and cognitive tutor technology are typically confounded, since the comparison students are in traditional mathematics courses which are not using the tutor and do not have the same course objectives. As mentioned above, we can compare courses as a whole, by developing summative evaluations that target both cognitive tutor course objectives and traditional objectives, but cannot assess the effectiveness of the technology specifically, as we have in the university setting. Finally, achievement tests in the APT Lisp Tutor research are typically conducted on-line, either in the same interface as the tutor (but without tutorial support) or in conventional programming environments. The programming exercises themselves are just like the tutoring exercises. Our high school mathematics assessments, in contrast, emphasize transfer, both to paper and pencil and to problem structures that vary from the tutor interface. Consequently, research on the validity of student modeling, as described in the next section, has focused on the programming tutor.

4.5. Domain and interface differences – adapting to student history

Differences in the problem domains and interfaces have led us to introduce a modeling feature to the Cognitive Algebra II Tutor that is not present in the APT Lisp Tutor. In the Algebra II Tutor we have incorporated a minimal model of problem solving history. The APT Lisp Tutor fosters top-down program decomposition. For example, only when the student selects the *defun* operator in

PROBLEM SITUATION

We own a company that makes and sells two different models of televisions. We need to decide how many Optimas and how many Futuras to make. The Optima sells for $250 and the Futura sells for $500. Our goal is to reach $750,000 in total income.

QUESTIONS

(1) If the company makes $300,000 from selling Optimas, how much income do we need from selling Futuras?
(2) If the company sells 500 Futuras, how much income do we need from selling Optimas?
(3) If the company makes $600,000 from Futuras, how many Optimas do we need to sell?
(4) If the company sells no Futuras, how many Optimas do we need to sell?
(5) If the company sells 1000 Optimas, how many Futuras do we need to sell?

Figure 4. A general linear form problem situation and questions.

	Number of Optimas	Number of Futuras	Optima Income	Futura Income	Total Income
Units	tvs	tvs	$	$	$
Formulas	x	y	$250x$	$500y$	$250x + 500y$
1	1200	900	300,000	450,000	750,000
2	2000	500	500,000	250,000	750,000
3	600	1200	150,000	600,000	750,000
4	3000	0	750,000	0	750,000
5	1000	1000	250,000	500,000	750,000

Figure 5. Worksheet solution to questions displayed in Figure 4.

the Lisp module, do editor nodes for the function name, parameters and body appear. Similarly, in the function body, editor nodes for arguments only appear after the function has been encoded. Each student action and help request can be related to an explicit superordinate goal in the existing problem state and the order of previous student actions has been relatively unimportant.

Cognitive Algebra II Tutor problems have a contrasting structure, as displayed in Figure 4. In this general linear form problem students answer five questions. They answer these questions by filling in a blank worksheet with five columns and seven rows. The end product of this problem solving activity is displayed in Figure 5. Students are required to label a column before filling in any cells in the column, but there are no other constraints on the order in which the student can fill in cells,

and the tutor can evaluate any problem solving action. A challenge arises in adapting to the student's cognitive state when the student asks for help. Help messages are organized around the questions asked in the problem and to adapt to the temporal sequence in which the student is answering questions, the tutor maintains a record of which question the student addressed in his or her immediately prior problem solving action. For example, if the student has just typed 5000 in the lower left cell of the table, the tutor records that the student was most recently working on Question 5. This enables the tutor to adapt help responses in three ways:

- If the student asks for help without selecting a cell in the worksheet, the tutor will look for the last question the student worked on and provide help on the next suitable step in that question.
- If the student asks for help in labeling a blank column, the tutor will check the last question the student was working on and and current state of that question in providing help.
- Finally, if the student asks for help in any cell corresponding to a question value, the message is couched in a phrase that is aligned with the student's most recent action (e.g. "Yes, let's start a new question" or "To continue with Question 2" or "Let's go back to Question 4").

5. Basic Student Modeling Research – Validating Knowledge Tracing in the APT Programming Tutor

In the final sections of the paper, we contrast the role of adaptive student modeling in our university basic research and in formative evaluations of the cognitive tutors in high school classrooms. In this section we review basic research that employed the APT Lisp Tutor to empirically validate knowledge tracing and cognitive mastery learning. In the following section we examine the formative use of knowledge tracing to refine the cognitive model in the Cognitive Algebra II Tutor.

5.1. Validating Knowledge Tracing

As described in Section 2.2, the learning and performance assumptions that underlie knowledge tracing can be used to predict student problem solving performance both in the tutor and in a post-test environment. A sequence of studies demonstrated that when four individual difference weights, one for each of the two learning and two performance parameters, are introduced for each student (as described in Section 2.2), the knowledge tracing model that guides cognitive mastery learning predicts student test performance quite accurately (Corbett and Anderson, 1995). Table 1 displays the results from the final study in this sequence. In this study 25 students worked through five sections in the APT Lisp Tutor curriculum, and completed cumulative tests following the first, fourth and fifth sections. In each section students completed a small set of required problems, then completed additional remedial

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

	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] the correlation coefficient, r, of actual and expected accuracy across students

problems until reaching the cognitive mastery criterion for every cognitive rule in the section. Across all five sections, students completed 38 required problems and an average of 29 remedial problems.

The students' average actual performance level (percent correct) on each of the three tests is displayed in the first column of Table 1. Overall accuracy is high (averaging 83% correct) as is expected since students have worked to reach cognitive mastery in the tutor. The knowledge tracing model's average predictions of student performance for the three tests is displayed in the second column. As can be seen, the model predicts students' average accuracy quite well for the three tests, although it consistently overestimates student performance by about 8%. The third column in the table presents a measure of how well the model predicts each individual student's performance. It displays the correlation of actual test performance and predicted test performance across the twenty-five students. The first test is fairly easy with relatively little variation in actual performance among students and the correlation coefficient, $r = 0.24$ is not significant. The following tests are more challenging with a greater range in actual performance across students. The correlation of actual and expected performance on the second test is marginally reliable ($r = 0.36$, $p < 0.10$) and the correlation on the third test is reliable and quite strong ($r = 0.66$, $p < 0.01$).

In addition to predicting student performance well, the knowledge tracing process was quite successful in guiding students to mastery. Fifty-six percent of students in this cognitive mastery learning condition achieved "A" level performance (90% correct) in this study. A second group of 21 students completed just the 38 required problems in the five curriculum sections without any remediation and only 24% of students in this comparison group reached "A" level performance.

5.2. Limitations of Cognitive Mastery Learning

While the knowledge tracing model is quite accurate, it consistently overestimates test performance accuracy by almost 10%, as in Table 1. Two follow-up studies examined the nature of this consistent overestimation (Corbett and Knapp, 1996; Corbett and Bhatnagar, 1997). These studies explored three possible sources of

Table 2. Actual and expected proportion of exercises completed correctly across students in three tests.

	Standard knowledge tracing fit			Revised fit with factual knowledge		
	Mean proportion correct			Mean proportion correct		
	Actual	Expected	Correlation[a]	Actual	Expected	Correlation[a]
Test 1	0.98	0.97	−0.28			
Test 2	0.90	0.93	0.34	0.90	0.91	0.55
Test 3	0.85	0.90	0.57	0.85	0.85	0.81

[a] correlation of actual and expected test accuracy across students.

the model's consistent over-prediction, (a) a decrease in student motivation between tutor problem solving and testing; (b) forgetting between tutor problem solving and testing; and (c) incomplete transfer of knowledge between the tutor environment and the test environment. These studies argued against a motivational explanation, since the students who fell short of predicted test performance actually worked *harder* on the test than the more successful students, as measured by time to complete the test. Similarly, the forgetting hypothesis was rejected because the order in which material was presented across successive tutor curriculum sections (hence the retention intervals), was unrelated to the magnitude of the model's overprediction of performance on the cumulative tests.

Instead, it was argued that the model's systematic tendency to overestimate performance arose because some students were learning suboptimal productions that were sufficient to complete the tutor problems, but did not transfer to immediate tests. To take a simple example, students may acquire rules that hinge on specifics of the tutor interface, e.g. use an 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 and to performing below expectations in testing. We assessed students' factual knowledge and found a strong, statistically significant inverse correlation between each student's factual knowledge score and the extent to which the tutor's knowledge tracing model overestimated the student's test performance, $r = -0.63$. That is, the weaker the student's factual knowledge, the more the model overestimates the student's test performance. When we developed a revised knowledge tracing model that incorporates students' declarative knowledge scores to estimate the probability that students learn productions that *transfer* to the tests, we obtained a better fit to a set of test data displayed in Table 2. In this study sixteen students worked to cognitive mastery in the first five sections of the APT Lisp Tutor curriculum and completed three cumulative tests as in the previous experiment. The left side of Table 2 presents the fit of the standard knowledge tracing model and the results replicate the pattern in Table 1. Again, average actual test performance in the first column is high, averaging 91% correct. The second column displays the model's average performance predictions. Again these average predictions are quite

Table 3. The number of tutor problems required to reach cognitive mastery in the standard and augmented feedback APT Lisp Tutor interfaces and subsequent test performance

	Standard interface			Augmented feedback interface		
	Tutor problems	Test p(correct)	Test $p > 0.9$	Tutor problems	Test p(correct)	Test $p > 0.9$
Test 2	39.8	0.86	0.50	38.7	0.96	0.90

accurate, although they slightly overestimate student performance. The correlations of actual student performance and the model's performance predictions across the sixteen students are displayed in the third column. The pattern of correlations largely replicates the pattern in Table 1, although only the correlation for Test 3 is reliable ($r = 0.57, p < 0.05$). The right side of Table 2 displays the fit of the revised model to the results of the second and third tests. This model employs a measure of each student's factual knowledge to estimate the probability that the students are learning rules that will transfer to test. As can be seen the model predicts average test performance almost perfectly and the correlation of actual and predicted performance across students increases by more than 20 points for each test.

While cognitive mastery learning, as guided by knowledge tracing, is effective in raising student test scores, this analysis of student factual knowledge indicates that the benefits of providing additional remedial problems of the same type is ultimately limited by the quality of the student's declarative knowledge. Consequently, in a subsequent study we modified the APT Lisp Tutor interface to provide augmented feedback during problem solving on factual knowledge about data structures and operator functionality (Corbett and Trask, 2000). This factual knowledge was specifically relevant to the problem solving tasks that are assessed in the second test in Tables 1 and 2. To assess this augmented feedback two groups of students worked to cognitive mastery in the study. One group of eighteen students employed the standard programming tutor interface and the other group of 21 students employed the augmented feedback interface. The impact of this manipulation on Test 2 is displayed in Table 3.

The number of tutor problems required to reach cognitive mastery learning is displayed in the first column for each interface in the table. As can be seen, these numbers are almost identical, 39.8 vs. 38.7. However, subsequent test accuracy is higher in the augmented feedback condition. Mean test accuracy is reliably higher, 0.96 vs. 0.86, $F(1, 37) = 8.7, p < 0.01$. Even more impressively, the proportion of students who reach "A" level performance (probability correct greater than or equal to 0.90) is substantially, and reliably higher in the augmented feedback condition than in the standard feedback condition, 0.90 vs. 0.50, ($z = 2.8, p < 0.01$).

PROBLEM SITUATION: A cannon shoots a cannon ball directly up into the air with an initial velocity of 640 ft/s. Define a variable for the time in seconds and use this variable to write the expression for the height of the cannonball.

QUESTIONS

(1) How high will the cannonball be in 5 s?
(2) How high will the cannonball be in 7 s?
(3) How many seconds after it was shot will the cannonball first be 1000 ft high?
(4) How many seconds after it was shot will the cannonball next be 1000 ft high?
(5) What is the maximum height that the cannonball will reach?

PLEASE GRAPH THE HEIGHT OF THE CANNONBALL AS A FUNCTION OF THE TIME SINCE IT WAS SHOT.

(6) How many seconds after it was shot will the cannonball be 2500 ft high?
(7) How many seconds after it was shot will the cannonball be back on the ground?

Figure 6. Problem statement for a vertical motion quadratics problem.

	Time	Height
Units	Seconds	Feet
formula	x	$-16x^2 + 640x$
1	5	2,800
2	7	3,696
3	1.6288	1,000
4	38.3712	1,000
5	20	6,400
6		2,500
7		

Figure 7. A partially completed worksheet for the problem situation displayed in Figure 6.

To summarize, this sequence of knowledge tracing studies indicates that when (a) individualized curriculum sequencing is guided by the tutor's knowledge tracing model; and (b) augmented feedback is employed to support the optimal encoding of conceptually difficult rules in the cognitive model, a cognitive tutor can realize the previously unfulfilled promise of mastery learning in helping almost all students become "A" students.

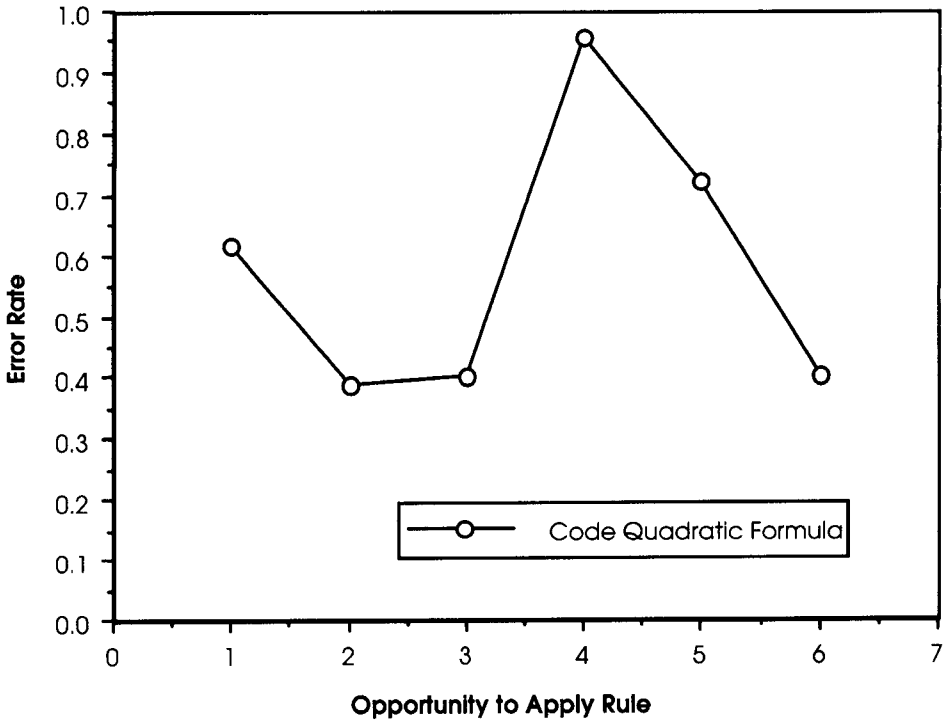

Figure 8. The learning curve for a hypothetical cognitive rule that generates a quadratic expression for a vertical motion problem. Error rate is plotted for successive opportunities to apply the rule.

6. Student Modeling and Formative Evaluations of the Cognitive Algebra II Tutor

In the Cognitive Algebra II Tutor knowledge tracing and detailed analyses of the production rule learning curves have been employed principally for formative evaluations. For example, consider the lesson on the quadratic formula and vertical motion, from which the problem in Figure 2 is drawn. Summative evaluations of learning gains indicate that this is a very successful lesson. In the 1997–1998 academic year, cognitive Algebra II students scored 2% correct on a pre-test that preceded the cognitive tutor lesson and scored 52% correct on the post-test that followed the lesson. Similarly, in a 1998–1999 assessment, cognitive Algebra II students scored 54% correct in a year-end assessment, while comparable students in a traditional Algebra II course scored 8% correct. Nevertheless, careful examination of student learning curves in the vertical motion lesson provides guidance for improving tutor effectiveness, as described in the following sections.

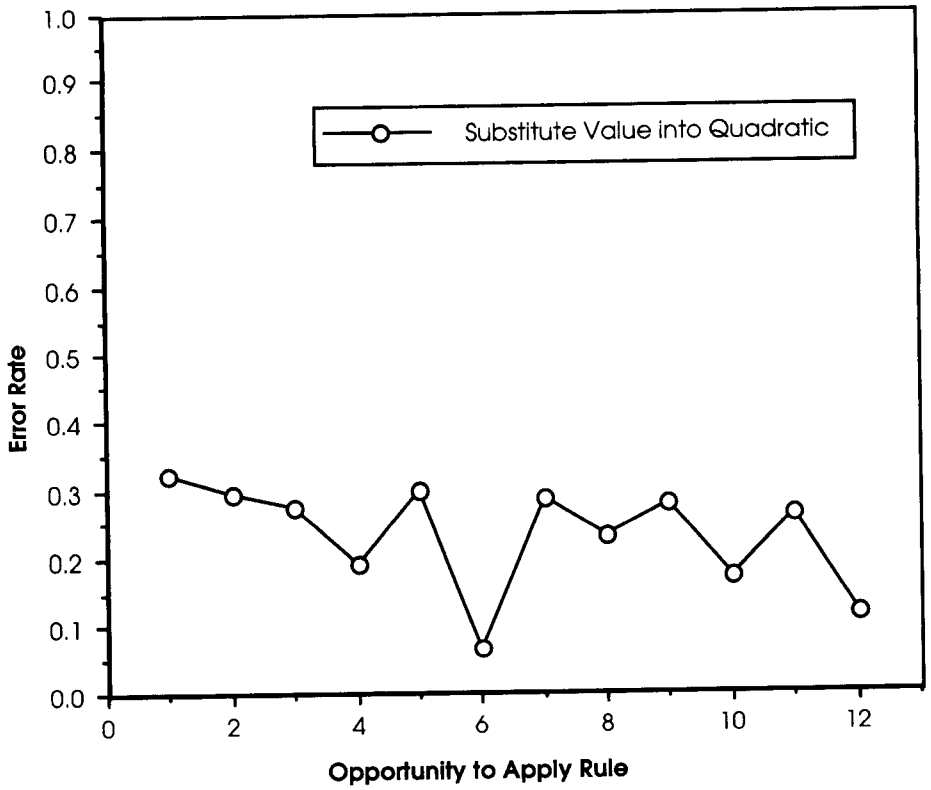

Figure 9. The learning curve for a hypothetical cognitive rule that substitutes a given time into a quadratic expression to find projectile height in a vertical motion problem. Error rate is plotted for successive opportunities to apply the rule. Odd-numbered points represent the first opportunity to apply the rule in successive problems and even-numbered points represent the second opportunity to apply a rule in each problem.

6.1. Detecting overgeneral production rules

Figure 6 enlarges the problem description for the vertical motion problem in Figure 2, and Figure 7 enlarges a partial worksheet solution. Consider the cognitive production rule that generates the quadratic expression in the formula row of the tutor worksheet, i.e. the production that generates the expression $-16x^2 + 640x$ in Figure 7. This production needs to fire once in each vertical motion problem, when the student types in the expression with the appropriate problem-specific constant terms. Figure 8 displays the learning curve for this production rule for a group of 34 students. The x-axis in this graph represents successive opportunities to apply this production rule in problem solving. Since the rule is only applied once in each tutor problem, each successive point in this graph is drawn from a successive problem in the curriculum. The y-axis represents error rate across students for their first

Table 4. Best fitting learning parameters for two production rule models of the learning curve data in Figure 9.

Action: Substitute a given time into a quadratic expression to compute projectile height	$p(L_0)$	$p(T)$	r
All opportunities (overgeneralized production rule)	0.83	0.44	0.37
First opportunity in successive problems (retrieving an appropriate rule from long term memory)	0.89	0.22	0.53
Second opportunity in successive problems ("do the same thing": reapplication of a rule in a problem)	0.75	0.94	0.44

$p(L_0)$ = probability the student has learned a rule prior to its first application.
$p(T)$ = probability of learning a rule at each opportunity to apply it.
r = correlation of the actual and predicted error rates across learning curve points.

attempt to apply this production at each successive opportunity. We expect error rate to decrease monotonically across opportunities and it does across the first three points, but then rises abruptly at the fourth point. This is an indicator of an over-general rule in the cognitive model. Inspecting the problem sequence reveals the overgeneralization. In the first three problems of the sequence, the constant term c in the quadratic formula $ax^2 + bx + c$ is 0, yielding an expression of the form $ax^2 + bx$. In the next three problems the constant c is a positive integer. While the original cognitive model coded formula knowledge as a single rule, a psychologically valid model needs to represent this knowledge as two separate rules.

Figure 9 depicts an empirical learning curve that displays a different form of overgeneralization. This production employs a quadratic expression to answer questions such as Questions 1 and 2 in Figure 7. In these questions elapsed time is given and the student must compute the corresponding vertical height. For example, to answer Question 1, the student needs to substitute the given time 5 for x in the quadratic expression $-16x^2 + 640x$ to compute the answer 2800. To answer Question 2, the student substitutes 7 into the same expression to compute the answer 3696. While error rate in this learning curve is gradually falling, it has a jagged profile in which each odd-numbered point tends to have a higher error rate than the preceding even numbered point. This is a pattern that rarely appears in the programming tutor data, but frequently turns up in the mathematics tutors, when students are asked to perform essentially the same task more than once in a problem. Each odd-numbered point in the graph represents the first opportunity in successive problems to apply the rule that substitutes a given time into a quadratic expression to compute a corresponding height. Each even-numbered point represents the second opportunity to apply this same rule within a problem. At the first opportunity, the student must recognize the question type, retrieve the appropriate production rule from long-term memory and apply it correctly. At the second opportunity, the student need only recognize the similarity to the first question; it is not necessary to retrieve the appropriate rule from long-term memory. Instead, the student can simply "do the same

Table 5. Best fitting learning probabilities for students' vertical motion problem solving.

Production Rule	$p(L_0)$	$p(T)$
Table Header (Labels and Units)	0.92	0.84
Symbolic Formulas	0.34	0.47
Code Given Time	0.82	1.0
Compute Height Given Time	0.78	0.39
Code Given Height	0.79	0.78
Compute Time Given Heights	0.49	0.32

$p(L_0)$ = probability the student has learned a rule prior to its first application.
$p(T)$ = probability of learning a rule at each opportunity to apply it.

thing" as on the prior question. When it is necessary to retrieve the rule from long term memory again in the following problem, accuracy declines. Note that this "do the same thing" production that fires for even number points in Figure 9 is an example of the type of suboptimal production rule that will not transfer to a test environment and can lead to overestimates of test performance.

Table 4 displays the hazard to knowledge tracing of overgeneralizing a rule. This table displays best fitting parameter estimates of the two learning parameters and goodness of fit for the learning curve data in Figure 9. The first row in the table displays the parameter estimates when all the points in Figure 9 are fit as the learning curve for a single overgeneralized production rule. The second row shows the parameter estimates in fitting just the odd-numbered points which represent the first opportunity to fire this hypothetical production in each problem. The third row shows the parameter estimates for just the even-numbered points, which represent the second opportunity in each problem to fire this hypothetical production. Note that the estimated probability that students already know these alternative production rules at the first opportunity to apply them, $p(L_0)$, is reasonably similar across the three rules, but the estimated probability that the students will learn the productions at each opportunity to apply them in problem solving, $p(T)$, varies substantially. When we fit the full empirical learning curve with a single overgeneralized production, as in the original model, we estimate that the probability of learning the production at each opportunity to apply it is 0.44. This is double the true learning rate in the second row of Table 4, 0.22, that is obtained when we fit only the odd number points in Figure 9 in which the student is required to retrieve the appropriate rule from long-term memory. The overgeneralized rule has a larger estimated learning rate because it conflates a second less useful rule ("do the same substitution I did on the last question") which fires for the even numbered points in Figure 9. As displayed in third row of Table 4, students readily learn this less useful rule; the best fitting estimate of $p(T)$ is 0.94. In summary, when we decompose the overly general rule, we get both true parameter estimates and better fits for the two distinguishable rules.

Finally, knowledge tracing parameter estimates enable us more generally to evaluate cognitive tutor effectiveness at the grain size of underlying production rules. For

example, Table 5 displays average best fitting learning parameters for six categories of productions in the vertical motion lesson. The learning rate in the lesson $p(T)$, is generally quite strong. This is consistent with the large gains observed in the summative evaluations described earlier and tends to be characteristic of the worksheet activities in the Cognitive Algebra II Tutor (cf. Corbettet al., 1998). Note that the best fitting $p(T)$ estimates are extremely high for the relatively routine problem solving actions such as typing column labels, which are always the same in the vertical motion lesson (time and height), and entering the given value in each question (i.e. given time in Questions 1 and 2, and given height in Questions 3 and 4). The more challenging activities are (a) typing an appropriate symbolic expression to represent each problem situation; (b) computing the height when given a time (by substituting the given time into the quadratic formula) and especially, computing the time when given a height (either with the quadratic formula or by reading it off the graph). The learning parameters for these productions (rows 2, 4 and 6 in the table) are also satisfactorily high.

7. Conclusion

Our university-based deployment of the APT Lisp Tutor in research and teaching and our outreach effort to achieve broad-based deployment of the Cognitive Mathematics Tutors have been conducted under very different circumstances and have achieved different, yet mutually reinforcing, results. Our programming tutor research has focused in large part on the validity and potential of model-based adaptivity to student learning needs. Our mathematics tutor research has demonstrated that advanced educational technology can be successfully deployed outside the university research environment and has helped identify conditions that make that deployment successful. The concluding sections of this paper have highlighted the different functional roles that student modeling has played in these two research strands. Student modeling is a principal research *topic* in our basic research, while serving primarily as a formative evaluation *tool* in the mathematics field research. Despite these important differences, we should not lose sight of an equally compelling conclusion that emerges from these research efforts. Adaptive student modeling in cognitive tutors has proven remarkably robust across both university and high school deployments and has contributed to important learning gains in both environments.

Acknowledgements

This research was supported by the Buhl Foundation, the Grable Foundation, the Howard Heinz Endowments, the Richard King Mellon Foundation, the Pittsburgh Foundation and by NSF grant number 9720359 to CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments.

References

Anderson, J. R.: 1983, *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R.: 1990, *The Adaptive Character of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R.: 1993, *Rules of the Mind*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Boyle, C. F. and Yost, G.: 1986, The geometry tutor. *Journal of Mathematical Behavior*, **5**, 5–19.

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.

Anderson, J. R. and Lebiere, C.: 1998, *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

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

Billsus, D. and Pazzani, M. J.: 2000, User Modeling for Adaptive News Access. *User Modeling and User-adapted Interaction*, **10**, 147–180 (this issue).

Block, J. H. and Burns, R. B.: 1976, Mastery learning. In: L. S. Shulman (ed.) *Review of Research in Education, Volume 4*. Itasca, IL: F. E. Peacock (AERA).

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

Corbett, A. T. and Bhatnagar, A.: 1997, Student modeling in the ACT Programming Tutor: Adjusting a procedural learning model with declarative knowledge. *Proceedings of the Sixth International Conference on User Modeling*. New York: Springer-Verlag Wein.

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

Corbett, A. T., Koedinger, K. R. and Anderson, J. R: 1999, Intelligent Computer Tutors: Out of the research lab and into the classroom. Paper presented at the annual meeting of the American Educational Research Association, Montreal, CA.

Corbett, A. T., Koedinger, K. R. and Hadley, W. S.: in press. Cognitive Tutors: From the research classroom to all classrooms. In: P. Goodman (ed.) *Technology Enhanced Learning: Opportunities for Change*. Mahwah, NJ: Lawrence Erlbaum Associates.

Corbett, A. T. and Trask, H. J.: 2000, Instructional Interventions in computer-based tutoring: Differential impact on learning time and accuracy. In: T. Turner, G. Szwillus, M. Czerwinski and F. Paterno (eds.), CHI 2000 Conference Proceedings.

Corbett, A. T., Trask, H. J. Scarpinatto, K.C. and Hadley, W.S.: 1998, A formative evaluation of the PACT Algebra II Tutor: Support for simple hierarchical reasoning. In: B. Goettl, H. Halff, C. Refield and V. Shute (eds.) *Intelligent Tutoring Systems: Fourth International Conference*, ITS '98. New York: Springer.

Fink, J. and Kobsa, A.: 2000, A review and analysis of commercial user modeling servers for personalization on the World Wide Web. *User Modeling and User-adapted Interaction*, **10**, 209–249 (this issue).

Kieras, D. E. and Bovair, S.: 1986, The acquisition of procedures from text: A production system analysis of transfer of training. *Journal of Memory and Language*, **25**, 507–524.

Koedinger, K. R. and Anderson J. R.: 1993, Effective use of intelligent software in high school math classrooms. In: P. Brna, S. Ohlsson and H. Pain (eds.) *Proceedings of AIED '93 World Conference on Artificial Intelligence in Education*. Charlottesville, VA: Association for the Advancement of Computing in Education.

Koedinger, K. R., Anderson, J. R. Hadley, W.H. and Mark, M.A.: 1997, Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education,* **8**, 30–43.

Koedinger, K. R. and Cross, K.: 1999, Tutoring answer-explanation fosters learning with understanding. In: S. P. Lajoie and M. Vivet (eds.) *Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration, Proceedings of AIED-99.* Amsterdam: IOS Press.

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.

Kulik, J. A., Kulik, C. C. and Cohen, P.A.: 1979, A meta-analysis of outcomes studies of Keller's Personalized System of Instruction. *American Psychologist,* **34**, 307–318.

Linton, F. and Schaefer, H.: 2000, Recommender Systems for Learning: Building User and Expert Models by Long-Term Observation of Application Use. *User Modeling and User-adapted Interaction* **10**, 181–207 (this issue).

National Council of Teachers of Mathematics (1989). *Curriculum and Evaluation Standards for School Mathematics.* Reston, VA: The Council.

Newell, A.: 1990, *Unified Theories of Cognition.* Cambridge, MA: Harvard University Press.

Reed, S. K., Dempster, A. and Ettinger, M.: 1985, Usefulness of analogous solutions for solving algebra word problems. *Journal of Experimental Psychology: Learning, Memory and Cognition,* **11**, 106–125.

Schofield, J. W.: 1995, *Computers and Classroom Culture.* Cambridge, England: Cambridge University Press.

Sleeman, D., and Brown, J. S.: 1982, *Intelligent Tutoring Systems.* New York: Academic Press.

Strachan, L., Anderson, J. Sneesby, M. and Evans, M.: 2000, Minimalist User Modelling in a Complex Commercial Software System. *User Modeling and User-adapted Interaction* **10**, 109–145 (this issue).

U.S. Department of Education, 1998. National Center for Education Statistics, Pursuing excellence: A study of U.S. twelfth-grade mathematics and science achievement in international context, NCES 98-049. Washington, D.C.: U.S. Government Printing Office.

Authors' Vitae

Albert Corbett is a Senior Research Scientist in the Human Computer Interaction Institute and co-director of the Pittsburgh Area Cognitive Tutor (PACT) Center at Carnegie Mellon University. Dr. Corbett received his B.A. in Psychology from Brown University and his Ph.D. in Psychology from the University of Oregon. Over the past 15 years he has brought his interests in human memory, learning and problem solving to the development and evaluation of cognitive tutors. These cognitive tutors have proven to be both rich environments for pursuing human cognition research and effective learning environments that are having a growing impact in real-world classrooms.

Megan S. McLaughlin received her B.A. in Linguistics and Psychology from the University of Virginia in 1998, and plans to receive her masters degree in Library and Information Science from the University of Pittsburgh in the Summer of 2001. She has been a Research Associate II for the PACT Center in the Human Computer

Interaction Institute at Carnegie Mellon University for two years. During her time at the PACT Center, she has contributed to the development of cognitive tutors for high school and middle school math classes.

K. Christine Scarpinatto received her B.S. in Languages in Japanese from Georgetown University and her Master of Science in Computational Linguistics from Carnegie Mellon University. In the past she has worked in the areas of natural language processing and automated translation. Presently, she is the software engineering lead for the PACT Center, where she does interface development and user modeling.