Implicit Versus Explicit Learning of Strategies in a Non-procedural Cognitive Skill

Kurt VanLehn[1], Dumiszewe Bhembe[1], Min Chi[1], Collin Lynch[1], Kay Schulze[2], Robert Shelby[3], Linwood Taylor[1], Don Treacy[3], Anders Weinstein[1], and Mary Wintersgill[3]

[1] Learning Research & Development Center, University of Pittsburgh, Pittsburgh, PA, USA
{VanLehn, Bhembe, mic31, collinl,lht3, andersw}@pitt.edu
[2] Computer Science Dept., US Naval Academy, Annapolis, MD, USA
schulze@artic.nadn.navy.mil
[3] Physics Department, US Naval Academy, Annapolis, MD, USA
{treacy, mwinter}@artic.nadn.navy.mil

Abstract. University physics is typical of many cognitive skills in that there is no standard procedure for solving problems, and yet a few students still master the skill. This suggests that their learning of problem solving strategies is implicit, and that an effective tutoring system need not teach problem solving strategies as explicitly as model-tracing tutors do. In order to compare implicit vs. explicit learning of problem solving strategies, we developed two physics tutoring systems, Andes and Pyrenees. Pyrenees is a model-tracing tutor that teaches a problem solving strategy explicitly, whereas Andes uses a novel pedagogy, developed over many years of use in the field, that provides virtually no explicit strategic instruction. Preliminary results from an experiment comparing the two systems are reported.

1 The Research Problem

This paper compares methods for tutoring non-procedural cognitive skills. A *cognitive* skill is a task domain where solving a problem requires taking many actions, but the challenge is not in the physical demands of the actions, which are quite simple ones such as drawing or typing, but in deciding which actions to take. If the skill is such that at any given moment, the set of acceptable actions is fairly small, then it is called a *procedural* cognitive skill. Otherwise, let us call it a *non-procedural* cognitive skill. For instance, programming a VCR is a procedural cognitive skill, whereas developing a Java program is a non-procedural skill because the acceptable actions at most points include editing code, executing it, turning tracing on and off, reading the manual, inventing some test cases and so forth. Roughly speaking, the sequence of actions matters for procedural skills, but for non-procedural skills, only the final state matters. However, skills exists at all points along the continuum between procedural and non-procedure. Moreover, even in highly non-procedural skills, some sequences

J.C. Lester et al. (Eds.): ITS 2004, LNCS 3220, pp. 521–530, 2004.

of actions may be unacceptable, such as compiling an error-free Java program twice in a row without changing the code or the compiler settings.

Tutoring systems for procedural cognitive skills can be quite simple. At every point in time, because there are only a few actions that students should take, the tutor can give positive feedback when the student's action matches an acceptable one, and negative feedback otherwise. When the student gets stuck, the tutor can pick an acceptable next action and hint it. Of course, in order to give feedback and hints, the tutor must be able to calculate at any point the set of acceptable next actions. This calculation is often called "the ideal student model," the "expert model." Such tutors are often called *model tracing* tutors.

It is much harder to build a tutoring system for non-procedural cognitive skills. Several techniques have been explored. The next few paragraphs review three of them.

One approach to tutoring a non-procedural skill is to teach a specific problem-solving procedure, method or strategy. The strategy may be well-known but not ordinarily taught, or the strategy may be one that has been invented for this purpose. For instance, the CMU Lisp tutor (Corbett & Bhatnagar, 1997) teaches a specific strategy for programming Lisp functions that consists of first inferring an algorithm from examples, then translating this algorithm into Lisp code working top-down and left-to-right. The basic idea of this approach is to convert a non-procedural cognitive skill into a procedural one. This allows one to use a model tracing tutor. Several model tracing tutors have been developed for non-procedural cognitive skills (e.g., Reiser, Kimberg, Lovett, & Ranney, 1992; Scheines & Sieg, 1994).

A second approach is to simply ignore the students' actions and look only at the product of those actions. Such tutoring systems act like a grader in a course, who can only examine the work submitted by a student, and has no access to the actions taken while creating it. Such tutors are usually driven by a knowledge base of condition-advice pairs. If the condition is true of the product, then the advice is relevant. Recent examples include tutors that critique a database query (Mitrovic & Ohlsson, 1999) or a qualitative physics essay (Graesser, VanLehn, Rose, Jordan, & Harter, 2001). Let us call this approach *product critiquing*.

Instead of critiquing the product, a tutoring system can critique the process even if it doesn't understand the process completely. Like product critiquing tutors, such a tutor has a knowledge base of condition-advice pairs. However, the conditions are applied as the student solves the problem. In particular, after each student action, the conditions are matched against the student's action and the state that preceded it. For instance, in the first tutoring system to use this technique (Burton & Brown, 1982), students played a board game. If they made a move that was significantly worse than the best available move, the tutor would consider giving some advice about the best available more. Let us call this approach *process critiquing*.

The distinctions between a process critiquing tutor and a model tracing tutor are both technical and pedagogical. The technical distinction is that a model tracing tutor has rules that recognize *correct* actions, whereas the process critiquing tutor has rules that recognize *incorrect* actions. Depending on the task domain, it may be much easier to author one kind of rule than the other. The pedagogical distinction is that model

tracing tutors are often used when learning the problem solving strategy is an instructional objective. The strategy is usually discussed explicitly by the tutor in its hints, and presented explicitly in the texts that accompany the tutor. In contrast, the process critiquing tutors rarely teach an explicit problem solving strategy.

All three techniques have advantages and disadvantages. Different ones are appropriate for different cognitive skills. The question posed by this paper is which one is best for a specific task domain, physics problem solving. Although the argument concerns physics, elements of it may perhaps be applied to other task domains as well.

2 Physics Problem Solving

Physics problem solving involves building a logical derivation of an answer from given information. Table 1 uses a two-column proof format to illustrate a derivation. Each row consists of a proposition, which is often an equation, and its justification. A justification refers to a domain principle, such as Newton's second law, and to the propositions that match the principle's premises. The tradition in physics is to display only the major propositions in a derivation. The minor propositions, which are often simple equations such as $a\_x=a$, are not displayed explicitly but instead are incorpo-

Table 1. A derivation of a physics problem

Problem: A motorboat cruises slowly to the mouth of the harbor, covering 230 m in 460 s at a constant velocity. It then speeds up to a cruising speed of 5 m/s in 60 seconds. What is its average acceleration while speeding up? Assume it always travels in a straight line. Let time 1 be when the motorboat starts the 230 m journey; let time 2 be when it starts speeding up; and let time 3 be when it reaches cruising speed.

	Proposition	**Justification**
1	d12_x = 230 m	Given (where d12 is the displacement of the motorboat from time 1 to 2, the x-axis is horizontal, and d12_x is the x-component of d12)
2	t12 = 460 s	Given (where t12 is the duration from time 1 to 2)
3	t23 = 60 s	Given (where t23 is the duration from time 2 to 3)
4	v3_x = 5 m/s	Given (where v3 is the motorboat's velocity at time 3)
5	v12_x = d12_x / t12	Definition of average velocity applied over the time interval from time 1 to time 2 (where v12 is the average velocity of the motorboat during time 1 to 2)
6	v12_x = 0.50 m/s	Algebraically solve 1, 2, 5 for v12_x
7	a23_x = (v3_x-v2_x)/t23	Definition of average acceleration applied over the time interval from time 2 to 3 (where a23 is the average acceleration of the motorboat during time 2 to 3, and v2 is the velocity of the motorboat at time 2)
8	v2_x = v12_x	Constant velocity
9	a23_x = a23	Projection
10	a23 = 0.075	Algebraically solve 3, 6, 7, 8, 9 for a23

rated algebraically into the main propositions. The justifications are almost never displayed by students or instructors, although textbook examples often mention a few major justifications. Such proof-like derivations are the solution structures of many other non-procedural skills, including geometry theorem proving, logical theorem proving, algebraic or calculus equation solving, etc.

Although AI has developed many well-defined procedures for deductive problem solving, such as forward chaining and backwards chaining, they are not explicitly taught in physics. Explicit strategy teaching is also absent in many other non-procedure cognitive skills.

Although no physics problem solving procedures are taught, some students do manage to become competent problem solvers. Although it could be that only the most gifted students can learn physics problem solving strategies implicitly, two facts suggest otherwise. First, for simpler skills than physics, many experiments have demonstrated that people can learn implicitly, and that explicit instruction sometimes has no benefit (e.g., Berry & Broadbent, 1984). Second, the Cascade model of cognitive skill acquisition, which features implicit learning of strategy, is both computationally sufficient to learn physics and an accurate predictor of student protocol data (VanLehn & Jones, 1993; VanLehn, Jones, & Chi, 1992).

If students really are learning how to select principles from their experience, as this prior work suggests, perhaps a tutoring system should merely expedite such experiential learning rather than replace it with explicit teaching/learning. One way to do that, which is suggested by stimulus sampling and other theories of memory, is to ensure that when students attempt to retrieve an experience that could be useful in the present situation, they draw from a pool of *successful* problem solving experiences. This in turn suggests that the tutoring system should just keep students on successful solution paths. It should prevent floundering, generation of useless steps, traveling down dead end paths, errors and other unproductive experiences. This pedagogy has been implemented by Andes, a physics tutoring system (VanLehn et al., 2002). The pedagogy was refined over many years of evaluation at the United States Naval Academy. The next section describes Andes' pedagogical method.

3 The Andes Method for Teaching a Non-procedural Skill

Andes does not teach a problem solving strategy, but it does attempt to fill students' episodic memory with appropriate experiences. In particular, whenever the student makes an entry on the user interface, Andes colors it red if it is incorrect and green if it is correct. Students almost always correct the red entries immediately, asking Andes for help if necessary. Thus, their memories should contain either episodes of green, correct steps or well-marked episodes of red errors and remediation.

The most recent version of Andes does present a small amount of strategy instruction in one special context, namely, when students get stuck and ask for help on what to do next. This kind of help is called "next-step help" in order to differentiate it from asking what is wrong with a red entry. Andes' next-step help suggests applying a major principle whose equation contains a quantity that the problem is seeking. Even

if there are other major principles in the problem's solution, it prefers one that is contains a sought quantity. For instance, suppose a student were solving the problem shown in Table 1, had entered the givens and asked for next-step help. Andes would elicit *a23* as the sought quantity and the definition of average velocity (shown on line 7 of Table 1) as the major principle.

Andes' approach to tutoring non-procedural skills is different from product critiquing, process critiquing and model tracing. Andes gives feedback during the problem solving process, so it is not product critiquing. Like a model-tracing tutor, it uses rules to represent correct actions, but like a process-critiquing tutor, it does not explicitly teach a problem solving strategy. Thus, is pedagogically similar to a process-critiquing system and technically similar to a model-tracing system.

Andes is a highly effective tutoring system. In a series of real-world (not laboratory) evaluations conducted at the US Naval Academy, effect sizes ranged from 0.44 to 0.92 standard deviations (VanLehn et al., 2002).

However, there is still room for improvement, particularly in getting students to follow more sensible problem solving strategies. Log files suggest that students sometimes get so lost that they ask for Andes' help on almost every action, which suggests that they have no "weak method" or other general problem solving strategy to fall back upon when their implicit memories fail to show them a way to solve a problem. Students often produce actions that are not needed for solving the problem, and they produce actions in an order that conforms to no recognizable strategy. The resulting disorganized and cluttered derivation makes it difficult to appreciate the basic physics underlying the problem's solution.

We tried augmenting Andes' next-step help system to explicitly teach a problem solving strategy (VanLehn et al., 2002). This led to such long, complex interactions that students generally refused to ask for help even when they clearly needed it. The students and instructors both felt that this approach was a failure.

It seems clear in retrospect that a general problem solving strategy is just too complex and too abstract to teach in the context of giving students hints. It needs to be taught explicitly. That is, it should be presented in the accompanying texts, and students should be stepped carefully through it for several problems until they have mastered the procedural aspects of the strategy. In other words, students may learn even better than Andes if taught in a model-tracing manner.

4 An Experiment: Andes Versus Pyrenees

This section describes an experiment comparing two tutoring systems, a model tracing tutor (Pyrenees) with a tutor that encourages implicit learning of strategies (Andes). Pyrenees teaches a form of backward chaining called the Target Variable Strategy. It is taught to the students briefly using the instructions shown in the appendix. Although Pyrenees uses the same physics principles and the same physics problems as Andes, its user interface differs because it explicitly teaches the Target Variable Strategy.

4.1 User Interfaces

Both Andes and Pyrenees have the same 5 windows, which display:

- The physics problem to be solved
- The variables defined by the student
- Vectors and axes
- The equations entered by the student
- A dialogue between the student and the tutor

In both systems, equations and variable names are entered via typing, and all other entries are made via menu selections. Andes uses a conventional menu system (pull down menus, pop-up menus and dialogue boxes), whereas Pyrenees uses teletype-style menus.

For both tutors, every variable defined by the student is represented by a line in the Variables window. The line displays the variable's name and definition. However, in Pyrenees, the window also displays the variable's state, which is one of these:

- *Sought*: If a value for the variable is currently being sought, then the line displays, e.g., "mb = SOUGHT: the mass of the boy."
- *Known*: If a value has been given or calculated for a variable, then the line displays the value, e.g., "mb = 5 kg: the mass of the boy."
- *Other*: If a variable is neither Sought nor Known, then the line displays only the variables name and definition, e.g., "mb: the mass of the boy."

The Target Variable Strategy's second phase, labeled "applying principles" in the Appendix, is a form of backwards chaining where Sought variables serve as goals. The student starts this phase with some variables Known and some Sought. The student selects a Sought variable, executes the Apply Principle command, and eventually changes the status of the variable from Sought to Other. However, if the equation produced by applying the principle has variables in it that are not yet Known, then the student marks them Sought. This is equivalent to subgoaling in backwards chaining. The Variables window thus acts like a bookkeeping device for the backwards chaining strategy; it keeps the current goals visible.

As an illustration, suppose a student is solving the problem of Table 1 and has entered the givens already. The student selects *a23* as the sought variable, and it is marked Sought in the Variable window. The student executes the Apply Principle command, selects "Projection" and produces the equation shown on line 9 of Table 1, *a23_x=a23*. This equation has an unknown variable in it, *a23_x*, so it is marked Sought in the Variable window. The Sought mark is removed from *a23*. Now the cycle repeats. The student executes the Apply Principle command, selects "definition of average acceleration," produces the equation shown on line 7 of Table 1, removes the Sought mark from *a23_x*, and adds a Sought mark to *v2_x*. This cycle repeats until no variables are marked Sought. The resulting system of equations can now be solved algebraically, because it is guaranteed to contain all and only the equations required for solving the problem.

In Andes, students can type any equation they wish into the Equation window, and only the equation is displayed in the window. In Pyrenees, equations are entered only by applying principles in order to determine the value of a Sought variable, so its

equation window displays the equation plus the Sought variable and the principle application, e.g., "In order to find W, we apply the weight law to the boy: $W = mb*g$."

Some steps, such as defining variables for the quantities given in the problem statement, are repeated so often that students master them early and find them tedious thereafter. Both Andes and Pyrenees relieve students of some of these tedious steps. In Andes, this is done by predefining certain variables in problems that appear late in the sequence of problems. In Pyrenees, steps in applying the Target Variable Strategy, shown indented in the Appendix, can be done by either the student or the tutor. When students have demonstrated mastery of a particular step by doing it correctly the last 4 out of 5 times, then Pyrenees will take over executing that step for the student. Once it has taken over a step, Pyrenees will do it 80% of the time; the student must still do the step 20% of the time. Thus, student's skills are kept fresh. If they make a mistake when it is their turn, then Pyrenees will stop doing the step for them until they have re-demonstrated their competence.

4.2 Experimental Subjects, Materials, and Procedures

The experiment used a two-condition, repeated measures design with 20 students per condition. Students were required to have competence in high-school trigonometry and algebra, but to have taken no college physics course. They completed a pre-test, a multi-session training, and a post-test.

The training had two phases. In phase 1, students learned how to use the tutoring system. In the case of Pyrenees, this included learning the target variable strategy. During Phase 1, students studied a short textbook, studied two worked example problems, and solved 3 non-physics algebra word problems. In phase 2, students learned the major principles of translational kinematics, namely, the definition of average velocity $v=d/t$, the definition average acceleration $a=(vf-vi)/t$, the constant-acceleration equation $v=(vi+vf)/2$ and freefall acceleration equation, $a=g$. They studied a short textbook, studied a worked example problem, solved 7 training problems on their tutoring system and took the post-test.

4.3 Results

The post-test consisted of 4 problems similar to the training problems. Students were not told how their test problems would be scored. They were free to show as much work as they wished. Thus, we created two scoring rubrics for the tests. The "Answer rubric" counted only the answers, and the "Show work" rubric counted only the derivations leading up to the answers but not including the answers themselves. The Show-work rubric gave more credit for writing major principles' equations than minor ones. It also gave more credit for defining vector variables than scalar variables.

Table 2 presents the results. Scores are reported as percentages. A one-way ANOVA showed that the pre-test means were not significantly different. When students post-tests were scored with the Answer rubric, their scores were not significantly different according to both an one-way Anova ($F(29)=.888$, $p=.354$) and an

Ancova with the pre-test as the covariate (F(28)=2.548, p=.122). However, when the post-test were scored with the Show-work rubric, the Pyrenees students scored reliably higher than the Andes students according to both an Anova (F(29)=6.076, p=.020) and an Ancova with the pre-test as the covariate (F(28)=5.527, p=.026).

5 Discussion

Pyrenees requires students to focus on applying individual principles, whereas Andes requires only that students write equations. Moreover, Andes allows students to combine several principle applications algebraically into one equation. Thus, the Andes students may have become used to deriving answers while showing less work. This would explain why they had lower Show-work scores.

However, having learned an explicit problem solving strategy did not seem to help Pyrenees students derive correct answers. This may be due to a floor effect—three of the four test problems were too difficult for most students regardless of which training they received. Also, during the test, students had to do their own algebraic manipulations, while during training, the tutors handled all the algebraic manipulations for them so that they could concentrate on learning physics.

This was the first laboratory evaluation of Andes and of Pyrenees, so we learned a great deal about how to improve such evaluations. In the next experiment in this series, we plan to pace the instruction more slowly and to give students more examples. We need to devise a testing method that doesn't require students to do their own algebra. Most importantly, we need a way to measure floundering, which we expect Pyrenees will reduce, and across-chapter transfer, which we expect Pyrenees will increase.

Although this experimental results should be viewed with caution due to the many improvements that could be made to the evaluation methods, the results are consistent with our hypothesis that Andes students learn problem solving strategies implicitly, which limits their generality and power relative to an explicitly taught strategy. When Pyrenees taught a problem solving strategy explicitly, its students employed a qualitatively better strategy on post-tests, but this did not suffice to raise their Answer score relative to the Andes students.

Table 2. Means and Standard Errors

	Andes	**Pyrenees**	**p (2-tailed)**
N	17	14	
Pretest	39.9 (4.99)	44.4 (5.62)	.555
Posttest Show-work	30.4 (2.71)	40.1 (2.84)	.020
Posttest Answer	35.3 (6.45)	26.8 (6.13)	.354
Adjusted Show-work	30.8 (2.54)	39.7 (2.80)	.026
Adjusted Answer	36.8 (4.94)	25.0 (5.44)	.122

Acknowledgements. This research was supported by the Cognitive Science Program of the Office of Naval Research under grant N00014-03-1-0017 to the University of Pittsburgh and grant N0001404AF00002 to the United States Naval Academy.

References

1. Berry, E. C., & Broadbent, D. E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology, 36A*, 209-231.
2. Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.
3. Corbett, A. T., & 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*.
4. Graesser, A. C., VanLehn, K., Rose, C. P., Jordan, P. W., & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. *AI Magazine, 22*(4), 39-51.
5. Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). Sherlock: A coached practice environment for an electronics troubleshooting job. In J. H. a. C. Larkin, R.W. (Ed.), *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches* (pp. 201-238). Hillsdale, NJ: Lawrence Erlbaum Associates.
6. Mitrovic, A., & Ohlsson, S. (1999). Evaluation of a constraint-based tutor for a database language. *International Journal of Artificial Intelligence and Education, 10*, 238-256.
7. Reiser, B. J., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (1992). Knowledge representation and explanation in GIL, an intelligent tutor for programming. In J. H. Larkin & R. W. Chabay (Eds.), *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches* (pp. 111-150). Hillsdale, NJ: Lawrence Erlbaum Associates.
8. Scheines, R., & Sieg, W. (1994). Computer environments for proof construction. *Interactive Learning Environments, 4*(2), 159-169.
9. VanLehn, K., & Jones, R. M. (1993). Learning by explaining examples to oneself: A computational model. In S. Chipman & A. Meyrowitz (Eds.), *Cognitive Models of Complex Learning* (pp. 25-82). Boston, MA: Kluwer Academic Publishers.
10. VanLehn, K., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*(1), 1-59.
11. VanLehn, K., Lynch, C., Taylor, L., Weinstein, A., Shelby, R., Schulze, k., Treacy, D., & Wintersgill, M. (2002). Minimally invasive tutoring of complex physics problem solving. In S. A. Cerri & G. Gouarderes & F. Paraguacu (Eds.), *Intelligent Tutoring Systems 1001: Proceedings of the 6th International Conference* (pp. 158-167). Berlin: Springer-Verlag.

Appendix: The Target Variable Strategy

The Target Variable Strategy is has three main phases, each of which consists of several repeated steps. The strategy is:

1 *Translating the problem statement.* For each quantity mentioned in the problem statement, you should:
1.1 define a variable for the quantity; and
1.2 give the variable a value if the problem statement specifies one, or mark the variable as "Sought" if the problem statement asks for its value to be determined. The tutoring system displays a list of variables that indicates which are Sought and which have values.
2 *Applying principles.* As long as there is at least one variable marked Sought in the list of variables, you should:
2.1 choose one of the Sought variables (this is called the "target" variable);
2.2 select a principle application such that when the equation for that principle is written, the equation will contain the target variable;
2.3 define variables for all the undefined quantities in the equation;
2.4 write the equation, replacing its generic variables with variables you have defined
2.5 (optional) rewrite the equation by replacing its variables with algebraic expressions and simplifying
2.6 remove the Sought mark from the target variable; and
2.7 mark the other variables in the equation Sought unless those variables are already known or were marked Sought earlier.
3 *Solving equations.* As long as there are equations that have not yet been solved, you should:
3.1 pick the most recently written equation that has not yet been solved;
3.2 recall the target variable for that equation;
3.3 replace all other variables in the equation by their values; and
3.4 algebraically manipulate the equation into the form V=E where V is the target variable and E is an expression that does not contain the target variable (usually E is just a number).

On simple problems, the Target Variable Strategy may feel like a simple mechanical procedure, but on complex problems, choosing a principle to apply (step 2.2) requires planning ahead. Depending on which principle is selected, the derivation of a solution can be short, long or impossible. Making an appropriate choice requires planning ahead, but that is a skill that can only be mastered by solving a variety of problems. In order to learn more quickly, students should occasionally make inappropriate choices, because this lets them practice detecting when an inappropriate choice has been made, going back to find the unlucky principle selection (use the Backspace key to undo recent entries), and selecting a different principle instead.