AD-A247 698

LEARNING BY EXPLAINING EXAMPLES TO ONESELF: A
COMPUTATIONAL MODEL(U) PITTSBURGH UNIV PA LEARNING
RESEARCH AND DEVELOPMENT CENTER K VANLEHN ET AL.

1/1

UNCLASSIFIED 17 FEB 92 N00014-86-K-0678

NL

END
FILMED

DTIC

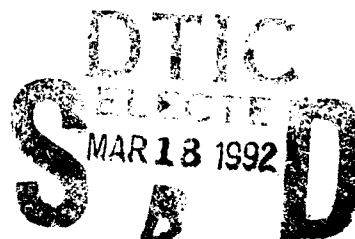MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

AD-A247 698

Learning by explaining examples to oneself:
A computational model

Kurt VanLehn and Randolph M. Jones

Learning Research and Development Center
University of Pittsburgh

February, 1992

DTIC
ELECTE
MAR 18 1992
S B D

92-06873

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
Unclassified	

2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	Approved for public release: Distribution unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
University of Pittsburgh		Computer Science Division Office of Naval Research

6c. ADDRESS (City, State, and ZIP Code)	7b. ADDRESS (City, State, and ZIP Code)
Learning Research and Development Center 3939 O'Hara Street Pittsburgh, PA 15260	800 N. Quincy Street Arlington, VA 22217-5000

8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
Same as Monitoring Organization		N00014-86-K-0678

8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

Learning by explaining examples to oneself: A computational model

12. PERSONAL AUTHOR(S)
Kurt VanLehn and Randolph M. Jones

13a. TYPE OF REPORT	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 92/02/17	15. PAGE COUNT

16. SUPPLEMENTARY NOTATION To appear in A. Meyrowitz & S. Chipman (Eds.) Cognitive Models of Complex Learning. Boston: Kluwer Academic Publishers.

17	COSATI CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Learning, Problem-solving, Cognitive Modeling, Analogy, Explanation-based Learning

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Several investigations have found that students learn more when they explain examples to themselves while studying them. Moreover, they refer less often to the examples while solving problems, and they read less of the example each time they refer to it. These findings, collectively called the self-explanation effect, have been reproduced by our cognitive simulation program, Cascade. Moreover, when Cascade is forced to explain exactly the parts of the examples that a subject explains, then it predicts most (60 to 90%) of the behavior that the subject exhibits during subsequent problem solving. Cascade has two kinds of learning. It learns new rules of physics (the task domain used in the human data modeled) by resolving impasses with reasoning based on overly-general, non-domain knowledge. It acquires procedural competence by storing its derivations of problem solutions and using them as analogs to guide its search for solutions to novel problems.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

Learning by explaining examples to oneself: A computational Model[*]

Kurt VanLehn and Randolph M. Jones
Learning Research and Development Center
University of Pittsburgh

Abstract

Several investigations have found that students learn more when they explain examples to themselves while studying them. Moreover. they refer less often to the examples while solving problems, and they read less of the example each time they refer to it. These findings, collectively called the self-explanation effect. have been reproduced by our cognitive simulation program. Cascade. Moreover. when Cascade is forced to explain exactly the parts of the examples that a subject explains. then it predicts most (60 to 90%) of the behavior that the subject exhibits during subsequent problem solving. Cascade has two kinds of learning. It learns new rules of physics (the task domain used in the human data modeled) by resolving impasses with reasoning based on overly-general. non-domain knowledge. It acquires procedural competence by storing its derivations of problem solutions and using them as analogs to guide its search for solutions to novel problems.

THE TWO MAJOR OBJECTIVES OF THE CASCADE PROJECT

As Tom Dietterich pointed out in the keynote address of the 1990 Machine Learning Conference. one of the biggest challenges in machine learning is to get machines to learn from ordinary instructional material. such as that used to train scientists, engineers and technicians. Not

1

only is this an exciting intellectual challenge, but it might help alleviate the notorious problem of getting expertise out of the culture of experts and into an operable form. The expert systems community recently realized that not all experts are good at explicating and explaining their knowledge, but instructors vary in quality too, so a common practice nowadays is to acquire knowledge for an expert system from an expert who is also a good instructor. Often, there are textbooks written by very good instructors. Utilizing this material requires having the kind of system that Dietterich envisioned.

The program described here, Cascade, is a direct response to Dietterich's challenge, for it can learn how to solve Newtonian mechanics problems from the same materials that undergraduates learn from. However, it is only a partial solution to the problem, because Cascade cannot read. The information in the prose parts of the textbook is given to it in a predigested form. As will be demonstrated later, this information is not as helpful in solving problems as one might think. People and Cascade acquire much of their problem solving skill by solving problems and by studying the textbook's worked example problems.

The second objective of the research presented here is to integrate and deepen the theory of skill acquisition. As theories go, the theory of cognitive skill acquisition is in its infancy. Theories range all the way from the highly integrated, nomological theories of certain natural sciences to loose collections of ideas which can be woven together to explain phenomena. The current theory of cognitive skill acquisition is in the collection-of-ideas stage. Given almost any behavior, a cognitive scientist can often string together ideas from psychology and AI that will offer at least a plausible explanation of the phenomenon. This is certainly an advance over the state of the art 25 years ago. However, the theory is not as integrated as it could be. For instance, no one has built a computational model of skill acquisition that starts as a novice and slowly becomes an expert while being trained on the same material as human students. Several models of pieces of this process have been built, including Sierra (VanLehn, 1990), Pups (Anderson & Thompson, 1989) and X (Pirolli, 1987). The reason that cognitive science has no simulated students is not just because it is technically difficult, but because we do not know which of the many ideas floating around should be woven together. Moving the theory of cognitive skill acquisition out of the collection-of-

2

ideas stage and into a stage of integrated student simulations will require deep thought and significant new empirical work. Development of student simulations should go hand-in-hand with these empirical advances, because such simulations are the only way to demonstrate the computational coherence and empirical coverage of an integrated theory of cognitive skill acquisition. Cascade is intended to be a step further in that it incorporates new empirical evidence from a study by Chi, Bassok, Lewis, Reimann and Glaser (1989). However, Cascade is far from a complete simulation, because some important cognitive processes, such as reading, have been deliberately omitted from the model.

Interesting new educational technology may result from developing the simulated students that are required of an integrated theory of cognitive skill acquisition. For instance, a simulated student might be a valuable tool for training teachers. Simulators have been successful adjuncts in training other skills, ranging from flying airplanes to trading stocks. It may be a worthwhile investment to use simulators to train teachers in the skills of selecting material to teach, organizing it, explaining it, detecting student misconceptions and remediating them. In addition to teacher training, there are other potential applications for simulated students as well (VanLehn, 1991b).

The lack of an integrated theory prevents development of many applications, not just educational ones. The problem is that a theory that is in the collection-of-ideas stage often provides multiple or vague explanations of phenomena, which means that it can make only ambiguous or vague predictions at best. Yet many applications, such as simulation, require the theory to make unambiguous, precise predictions. Until our understanding of cognitive skill acquisition is good enough that we can make such predictions, many applications are beyond our reach.

This chapter is intended as a summary of the results so far from the Cascade project. The project has gone through three major phases. In the first phase, the program was developed and shown capable of learning Newtonian mechanics correctly (VanLehn & Jones, in press). In the second phase, the major findings from the Chi et al. study were simulated (VanLehn, Jones & Chi, in press). In the third phase, protocols of each of the 9 subjects in the Chi study were simulated individually. The third phase is ongoing, so we can present only some of the planned analyses. In particular, we evaluate the overall fit of Cascade to the protocols,

3

which is important for seeing how well Cascade functions as a simulated student and as a knowledge acquisition system that would satisfy Dietterich's challenge. This chapter follows the historical development by first describing the Chi et al. (1989) study, then describing Cascade, then describing how Cascade accounts for the Chi et al. findings, then describing how it simulates individual subjects.

THE SELF-EXPLANATION EFFECT

One of the major open issues in cognitive skill acquisition is understanding what happens when people study examples. (An example is a problem together with a solution that is printed or demonstrated for the student.) Much research has shown that when people are given instruction consisting of theory, examples and explanations, they rely heavily on the examples (e.g., Anderson, Farrell & Saurers, 1984; Sweller & Cooper, 1985). In some cases they seem to ignore the theory and explanations, and in other cases their learning is actually retarded by them (e.g., LeFevre & Dixon, 1986; Charney, Reder, & Kusbit, 1990; Ward & Sweller, 1990). Because examples seem to do much more of the teaching than was previously thought, it is important to understand how they work.

Chi et al. (1989) took a direct approach to understanding how students study examples. They collected protocols as subjects studied examples in classical particle dynamics, the first topic in a typical first-year college physics course. Nine subjects studied the first three chapters of a college textbook, then read the prose part of a chapter on Newton's laws. They took a test on their understanding of the chapter, then studied 3 examples and solved 25 problems. Protocols were taken as they studied the examples and solved the problems. On the basis of the scores on problem solving, the subjects were divided into two groups. The 4 students with the highest scores were called the Good solvers; the 4 students with the lowest scores were called Poor solvers, and one student was not analyzed (see Chi and VanLehn, 1991, for a discussion of the subjects' backgrounds and the median-split procedure). Since the students in both groups scored the same on pre-tests, the Good solvers seemed to have learned more during the experiment. Using protocol analysis, Chi et al. attempted to find out how the Good solvers managed to learn more than

4

the Poor solvers from the same material. They found four differences:

1. The Good solvers uttered more self-explanations as they studied examples, whereas the Poor solvers' comments were mostly paraphrases of the examples' statements.

2. All students commented frequently on whether they understood what they had just read. The Good solvers tended to say that they did not understand what they had just read, whereas the Poor solvers tended to say that they did understand. However, the Poor solvers' scores show that they understood less than the Good solvers. This indicates that the Poor solvers' self-monitoring was less accurate than the Good solvers'.

3. During problem solving, the Poor solvers tended to refer back to the examples more often than the Good solvers.

4. When the Good solvers referred to the examples, they read fewer lines than the Poor solvers. The Poor solvers tended to start at the beginning of the example and read until they found a useful line, whereas the Good solvers started reading in the middle of the example and read only one line.

Similar findings have also been observed in protocol studies of students learning Lisp (Pirolli & Bielaczyc, 1989; Bielaczyc & Recker, 1991), electrodynamics (Fergusson-Hessler & de Jong, 1990) and biology (Chi, de Leeuw, Chiu, & LaVancher, 1991). This cluster of findings is called the self-explanation effect.

THE CASCADE MODEL

A consensus has emerged in both machine learning and cognitive psychology that it is important to distinguish two kinds of learning:

- One kind of learning is responsible for getting knowledge from the environment into the mind of the agent. This is called knowledge acquisition in the cognitive skill acquisition literature and knowledge-level learning in machine learning. There are many possible learning processes, depending on the type of instructional

information available in the environment and the type of knowledge to be acquired. The Cascade project, for instance, focuses on how agents can learn college physics by studying worked example exercises and solving problems.

- The other kind of learning increases the effectiveness of knowledge that is already in the mind of the agent. This is called knowledge compilation or knowledge tuning in the cognitive skills literature, and symbol-level learning in machine learning. This class of learning mechanisms includes explanation-based learning (EBL), chunking (Newell, 1990), production composition (Anderson, 1983), and many others. Some of these mechanisms provide explanations for robust findings in the skill acquisition literature (e.g., Anderson, 1987: Newell, 1990).

In order to determine whether the learning in the Chi et al. study was knowledge acquisition or knowledge compilation, and to set the stage for developing a simulated student, we began by developing a problem solver that could solve the problems in the study.[1] The solver was based on past mechanics problems solvers (Bundy et al., 1979: Larkin, 1983: Novak & Araya, 1980) as well as our informal inspection of the Chi et al. protocols. The resulting solver had 62 physics rules and a host of mathematical and common sense rules. These 62 rules became the target knowledge base. The first goal of the Cascade project was to understand when they were learned and how.

In order to find out where the target rules could be learned, two people who were not involved in the development of the knowledge base determined whether each rule was mentioned anywhere in the textbook prior to the point where the examples were introduced. There was 95% agreement between the two judges, and disagreements were settled by a third judge. They determined that only 29 of the 62 rules were mentioned in the text. The other 33 rules would have to be learned during example studying or problem solving.[2] This indicates that knowledge

---

[1] Actually, it could solve only 23 of the 25 problems. The other two involved kinematics knowledge that we did not bother to formalize. These two problems will be ignored throughout the remainder of the chapter.

[2] They could also be recalled from earlier training in physics, but there is evidence that this seldom occurred (Chi & Vanlehn, 1991: VanLehn, Jones & Chi, in press).

acquisition must be going on during example studying and/or problem solving. Knowledge compilation alone would not suffice to explain the subjects' learning.

Cascade models two basic activities: explaining examples and solving problems. Knowledge acquisition goes on during both. Because the type of physics problems used in Chi's study involve only monotonic reasoning, Cascade uses a rule-based, backwards chaining theorem prover (similar to Prolog) to implement both activities. A physics example is presented to Cascade as a set of propositions representing the givens of the problem, a list of sought quantities, and the lines of the problem's solution. For instance, the example of Figure 1 is represented with the information of Table 1. Cascade explains each line by proving that it follows from the givens and the preceding lines. To solve a problem, Cascade is presented with propositions representing the problem's givens and is asked to prove a proposition of the form "The value of Q is X" for each sought quantity Q. In the process of proving the proposition, Cascade derives a value for the variable X, thus solving that part of the problem. Although this model of problem solving and example explaining is clearly too simple to cover all task domains, it suffices for physics and other task domains dominated by monotonic reasoning.

Cascade includes two kinds of analogical problem solving. Both types of analogy begin by retrieving an example and mapping the example's givens to the current problem's givens. These retrieval and mapping processes usually correspond to overt behavior. The subjects flip through the textbook pages in order to locate an example, then look back and forth between the example and problem, comparing the diagram and text that describe the example's problem with the diagram and text describing the problem they are trying to solve. This behavior generally occurs only once per problem. All the examples and most of the problems are accompanied by diagrams, and usually the subjects would search for an analogous example after looking at the diagram and before reading the problem. Thus, we think that the major process of retrieving an analogous problem is based on recalling, finding and comparing diagrams. This retrieval process was not modeled in Cascade. The system was simply told which examples the subjects retrieved, and forced to retrieve the same ones.

7

Problem: The figure on the left below shows a block of mass m kept at rest on a smooth plane, inclined at an angle of 35 degrees with the horizontal, by means of a string attached to the vertical wall. What are the magnitudes of the tension force and the normal force acting on the block?
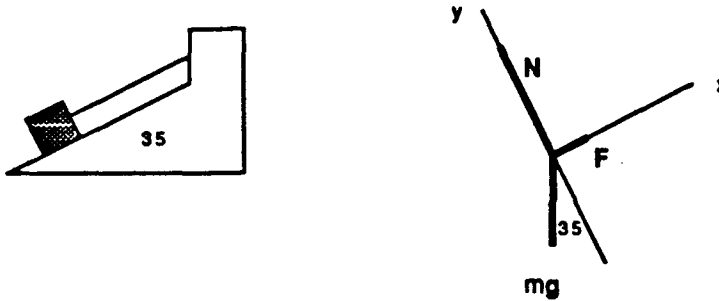
Solution:

(1) We choose the block as the body.

(2) The forces acting on the block are shown in the free-body diagram on the right.

(3) Because we wish to analyze the motion of the block, we choose ALL the forces acting ON the block. Note that the block will exert forces on other bodies in its environment (the string, the earth, the surface of the incline) in accordance with the action-reaction principle: these forces, however, are not needed to determine the motion of the block because they do not act on the block.

(4) Since the block is unaccelerated, we obtain:

$$F+N+mg=0.$$

(5) It is convenient to choose the x-axis of our reference frame to be along the incline and the y-axis to be normal to the incline (see figure above, right).

(6) With this choice of coordinates, only one force, mg, must be resolved into components in solving the problem.

(7) The two scalar equations obtained by resolving mg along the x- and y-axes are:

$$F - mg \sin 35 = 0 \quad \text{and} \quad N - mg \cos 35 = 0.$$

(8) From these equations F and N can be obtained if m is given.

Figure 1: An example

Table 1: An English version of the representation of the example of Figure 1

Problem givens:

The current situation is named Ix.

Ix is a standard-gravity situation.

Block-ix is a block.

String-ix is a massless string.

Plane-ix is an frictionless inclined plane.

Block-ix slides on Plane-ix.

String-ix is tied to Block-ix.

Block-ix is at rest.

Block-ix is above Plane-ix.

String-ix is above Block-ix.

String-ix is to the right of Block-ix.

The inclination of Plane-ix is 35.

The inclination of String-ix is 35.

The mass of Block-ix is m.

Problem soughts:

The magnitude of the tension force on Block-ix due to String-ix.

The magnitude of the normal force on Block-ix due to Plane-ix.

Solution lines:

The set of bodies of Ix is Block-ix.

The set of arrows on the free-body diagram for Block-ix is {an arrow at inclination 35 pointing up, an arrow at inclination 115 pointing up, an arrow at inclination 90 pointing down}.

The set of axes on the free-body diagram for Block-ix is {an x-axis at inclination 35, a y-axis at inclination 115}.

The magnitude of the tension force on Block-ix due to String-ix is $0 - 0 + (1(mg)\sin(35))$.

The magnitude of the normal force on Block-ix due to Plane-ix is $0 - (1(mg)\cos(35)) + 0$.

9

One of the two kinds of analogy is used to make search control decisions. It comes into play when Cascade has two or more rules for achieving a goal and it needs to select among them. It uses the analogical mapping to see if the example's derivation has a goal that is equivalent to the goal that it is currently working on. If it finds an equivalent old goal, the rule that achieves the old goal is chosen for achieving the new goal. This type of analogy is called analogical search control, because it uses the example as a source of advice on which of several alternatives to try first. For instance, a student might say, "I cannot tell whether I should project this onto the x-axis or the y-axis. At an analogous point in the example, they projected onto the x-axis, so I'll try that too." Analogical search control is also used in the Eureka system (Jones, 1989, this volume).

The second type of analogy is used when Cascade cannot find a rule that will apply to the current goal. It uses the analogical mapping to try to find a line in an old example that it can convert into an appropriate rule. It looks for a line in the example's solution that mentions the current goal (or rather, a goal equivalent to the current goal under the mapping). Most lines are equations, so it is simple to convert a line to a temporary rule which can then be used to try to achieve the goal. For instance, a student might say, "I need some way to get the tension of string A. The example has a line saying that string 1's tension is $mg\sin(30)$. Those two strings are analogous, and 30 degrees is analogous to 45 degrees in this problem, so I bet that the tension of string A is $mg\sin(45)$." This type of analogy is called transformational analogy, after a similar method explored by Carbonell (1986). As Carbonell discovered, transformational analogies often yield wrong answers.

A major difference between the two kinds of analogy is that analogical search control refers to the rules that achieved goals during the solution of an example, whereas transformational analogy only refers to the lines of the solution. When Cascade explains an example, it stores in memory a set of triples, each of which contains the example's name, a goal and the rule that achieved it. These triples are what analogical search control searches through. If an example is not explained, then no derivation is recorded, so analogical search control cannot get any advice from that example. On the other hand, transformational analogy refers only to the solution lines. These are present regardless of whether the example

10

is explained. since they merely represent what the student can see as they look at the page containing the example. Thus, transformational analogy can function even if the example has not been explained.

Cascade's main knowledge acquisition method is called explanation-based learning of correctness or EBLC (VanLehn, Ball & Kowalski, 1990). The basic idea is to divide knowledge into domain knowledge and non-domain knowledge. Domain knowledge represents rules that the student believes to be correct and appropriate for the task domain. Non-domain knowledge represents rules that are believed to be incorrect or relevant only to other task domains. The most important non-domain rules for learning are overly general rules. They can apply to many situations, but they often draw incorrect conclusions. For instance, a domain rule is "If there is a tension force F caused by a string S, and the tension in the string is T, then the magnitude of the tension force is also T." An overly general rule is, "If there is an entity F, with a part S, and a property of part S has value T, then a property of the entity F also has value T." This rule happens to be a generalization of the domain rule, but as argued in VanLehn and Jones (in press), not all domain rules have plausible overly general counterparts.

The basic idea of EBLC is to use overly general rules whenever domain rules fail, then save a specialization of the overly general rule as a new domain rule if all goes well. For instance, the domain rule just mentioned is learned by specialization of the overly general rule. EBLC begins when Cascade reaches an impasse that is caused by missing rules in the knowledge base. An impasse is defined to be an occasion when the current goal matches none of the known domain rules or problem givens. Impasses can be caused by missing domain knowledge or by reaching the end of a dead end path in the search space which could have been avoided by making a better search control decision earlier. Cascade explicitly checks for the latter possibility before deciding that an impasse is caused by missing knowledge. To resolve a missing-rule impasse, Cascade tries to use an overly general rule to achieve the stuck goal. If the use of such a rule ultimately leads to achieving the current top level goal (i.e., to explain a line or to find the value of a sought), then Cascade forms a new domain rule that is a specialization of the overly general one. The specialization is chosen so that it is also a generalization of the particular usage. For instance, on one problem Cascade could not determine the

11

pressure in a part of a container even though it knew the pressure in the whole. Since there was no alternative solution to the problem using its domain rules, Cascade decided that it was at a missing-rule impasse. It applied the overly-general rule, "If an object is composed of parts, then the property values of the parts and the wholes are the same." This rule application ultimately led to a solution of the problem. Cascade then formed a new domain rule, "If a container has a part, then the pressure in the part is equal to the pressure in the whole." Thus, Cascade learned a correct rule of physics by specializing an overly general rule in order to resolve an impasse caused by missing domain knowledge.

Cascade has a second technique for learning new rules. It applies only when it is explaining an example and attempting to prove a proposition that has no variables. If it cannot prove the proposition with either domain rules or overly general rules, then it gives up and simply accepts that the proposition is true. It also builds a rule that sanctions this in future similar cases. The rules say, in essence, that if a later problem is analogous to this problem, then the analog to this proposition can be assumed true for that problem too. This type of learning is called analogical abduction.

From a machine learning point of view, Cascade does both knowledge-level learning (via EBLC and analogy abduction) and symbol-level learning (via the saving of derivations, which are used by analogical search control). EBLC and analogy abduction are both triggered by impasses, so they will often be referred to as impasse-driven learning.

As a summary, Table 2 lists Cascade's main processes. Notice that a new one has been slipped in. Cascade can be told to ignore an example line instead of self-explaining it, a trivial process labeled "acceptance" in the table.

Cascade's learning is similar to those proposed by existing ... ies of skill acquisition. We believe that analogical search control can eventually provide an account for the practice effects usually explained by knowledge compilation (Anderson, 1983), chunking (Newell, 1990) and other learning mechanisms. EBLC is similar to proposals by Schank (1986), Lewis (1988), Anderson (1990) and others, which also acquire new knowledge at impasses by specializing existing, overly general knowledge. Although all these models of skill acquisition are similar in spirit, they differ in significant ways. For more on the Cascade system and a

12

Table 2: Cascade's major processes

Example studying

- Self-explanation: Prove a line via backwards chaining.

- Acceptance: Ignore the example line.

Problem Solving

- Regular problem solving: Find a value for a sought via backwards chaining. At search control choice points, use analogical search control to decide which rule to apply.

- Transformational analogy: Find a line in an example that could be adapted to achieve the current goal.

Impasse-driven learning

- Explanation-based learning of correctness (EBLC): Apply an overly general rule. If that leads to success, save a specialization as a new domain rule.

- Analogy abduction: Like transformational analogy, except a rule is built so that future occurrences of the goal will be handled the same way.

detailed comparison with its predecessors, see VanLehn and Jones (in press).

MODELING THE SELF-EXPLANATION EF-FECT WITH CASCADE

A simple hypothesis for explaining the four major differences between Good and Poor solvers is that Good solvers chose to explain more example lines than Poor solvers. To test this, several simulation runs were made. All these simulations began with the same initial knowledge. The initial domain knowledge consisted of the 29 physics rules that three judges found to be present in the text (see the discussion at the begin- ning of the preceding section). The rest of the initial knowledge base consists of 45 non-domain rules, of which 28 represented common sense physics (e.g., a taut rope tied to a object pulls on it) and 17 represented over-generalizations, such as "If there is a push or a pull on an object at a certain angle, then there is a force on the object at the same angle." See VanLehn, Jones and Chi (in press) for a list of the overly general rules.

In principle, Cascade can use regular problem solving or transforma- tional analogy at any goal. For the sake of these experiments, we gave it a fixed strategy. It would first try regular problem solving. If that failed due to missing domain knowledge, then impasse-driven learning was applied. Transformational analogy was used only as a last resort.

The first simulation was intended to model a very good student who explains every line of every example. Cascade first explained the 3 ex- amples in the study, then it solved the 23 problems. (The 2 problems that are not solvable by the target knowledge were excluded.) It was able to correctly solve all the problems. It acquired 23 rules: 8 while explaining examples and 15 while solving problems. All but one of the rules was learned by EBLC; analogical abduction learned the other. The new rules are correct physics knowledge, allowing for the simplicity of the knowledge representation. Moreover, they seem to have the right degree of generality in that none were applied incorrectly and none were inapplicable when they should have been applicable. However, some of the rules dealt with situations that only occurred once in this problem

14

set, so they were never used after their acquisition.

The second simulation was intended to simulate a very poor student who does no self-explanation. Because none of example lines were explained, there was no opportunity for EBLC to learn new rules during example studying, nor were any derivations left behind for use by analogical search control during later problem solving. Cascade was given the same 23 problems given to it in the good student simulation. It correctly solved 9 problems. Apparently these problems require only knowledge from the text. As Cascade solved these problems, Cascade learned 3 correct rules via EBLC. On 6 other problems, Cascade found an incorrect solution. EBLC did not occur on these problems. On the remaining 8 problems, Cascade failed to find any solution or its search went on for so long that it was cut it off after 20 minutes. Although EBLC was used extensively on these problems, the rules produced were always incorrect. On the assumption that a poor student would not believe a rule unless it led to a correct solution to a problem, rules acquired during failed solution attempts were deleted. Thus, the poor student simulation acquired only 3 rules and solved only 9 problems correctly.

Explaining the self-explanation correlations

Cascade should be able to explain the four differences observed by Chi et al. (1989) between Good and Poor solvers. Assuming that the number of self-explanatory utterances is directly proportional to the number of lines explained during example studying, the job facing Cascade is to explain why explaining more lines causes better scores on quantitative post-tests (finding 1), more accurate self-monitoring (finding 2) and more frequent (finding 3) and more economical reference to the examples (finding 4).

The contrast between the good and poor student simulations indicates that Cascade can reproduce the positive correlation between the number of example lines explained and the number of problems solved correctly. During the good student simulation, it explained all the example lines and got all 23 problems correct; on the poor student simulation, it explained none of the example lines and got 9 of the problems correct. Knowing the operation of Cascade, it is clear that having it explain an intermediate number of lines would cause it to correctly answer an in-

termediate number of problems. So the two extreme points (the two simulations) plus Cascade's deterministic design are sufficient to demonstrate the main finding of the self-explanation effect.

One of the major advantages of a simulation like Cascade is that one can run it many times with different components turned off in order to ascertain why it succeeds. In particular, 20 rules were learned by the good student simulation and not by the poor. For each rule, we can find out why self-explanation allowed Cascade to learn it.

First, when more lines are explained, Cascade is more likely to stumble across a gap in its domain knowledge. Such missing knowledge causes impasses, which lead to impasse-driven learning and the acquisition of new rules during example explaining. Of the 20 rules that were learned during the good student simulation and not the poor, 8 (40%) were learned while explaining examples.

Analogical search control also aided the good student simulation's learning. When more lines are explained, more derivations become available for analogical search control. Analogical search control tends to keep Cascade on solution paths during problem solving, and this means that any impasses that occur are more likely to be due to missing domain knowledge. Thus, EBLC is more often applied to appropriate impasses, and thus more often generates correct domain rules. Of the 20 rules, 9 (45%) require analogical search control for their acquisition.

The acquisition of rules during example studying helps produce contexts during problem solving that allow EBLC to learn more rules during problem solving even without the aid of analogical search control. Of the 19 rules, 3 (15%) can be acquired during problem solving even when analogical search control is turned off. These new rules also contributed to the improvement in problem solving. Table 3 summarizes the learning of the two runs.

Cascade provides a simple explanation of the correlation between the amount of self-explanation and the accuracy of self-monitoring statements. The explanation assumes that negative self-monitoring statements (e.g., "I don't understand that") correspond to impasses, and that positive self-monitoring statements (e.g., "Ok, got that.") occur with some probability during any non-impasse situation. When more example lines are explained, there are more impasses, and hence the proportion of negative self-monitoring statements will be higher. In the

16

Table 3: Rules learned during Good and Poor student simulations

Good	Poor	When acquired
8	0	Example studying
		Problem solving
3	3	No ex. studying rules, no analogical search control
3	0	With ex. studying rules, no analogical search control
9	0	With ex. studying rules, with analogical search control
23	3	Total

extreme case of the poor student simulation, where no example lines are explained, all the self-monitoring statements during example processing would be positive, which is not far off from Chi et al.'s observation that 85% of the Poor solver's self-monitoring statements were positive.

The third and fourth findings involve the frequency and specificity of analogical references during problem solving. The number of references made by analogical search control and transformational analogy were counted. We assumed that only some of the analogical search control references to the derivation were overt, and that the others were mental references that would not show up in the Chi et al. data. This gave us a prediction of the frequency of analogical references. To get a prediction of the specificity of analogical references (i.e., the number of example lines read per reference), we counted the number of lines read by transformational analogy before it found one it could use, and we assumed that someone using analogical search control would go directly to the line whose derivation contained the sought goal. Given these assumptions, the good student simulation produced fewer and more specific analogical references than the poor student simulation, thus modeling the Chi et al. finding (see VanLehn, Jones & Chi, in press, for details).

Discussion

Although we controlled Cascade's behavior during example studying, by either telling it whether to explain the examples or not, its behavior during problem solving was determined solely by how much it learned

17

during example studying. Qualitatively, the behaviors of the Good and Poor runs were quite similar to the behaviors of the Good and Poor students during problem solving. The good student simulation tended to stay on solution paths, use regular problem solving more often than transformational analogy, and learn something from the occasional impasses it encountered. The Poor student simulation tended to wander down unproductive paths, use transformational analogy more often, and learn nothing from the many impasses that it encountered.

These properties of Cascade's problem solving behavior are consistent with a preliminary analysis by Chi, VanLehn & Reiner (1988), who analyzed the protocols of a Good solver and a Poor solver as they solved the same problem. The Poor solver's protocol was divided into 77 episodes, and of these, 30 (39%) resulted in impasses.[3] Many of these impasses seemed to result in acquiring incorrect beliefs. In contrast, the protocol of the Good solver was divided into 31 episodes, of which only 7 (23%) resulted in impasses. In 6 of these, the Good solver seemed to learn a correct piece of knowledge. This preliminary analysis indicates that the Poor solvers had proportionally more impasses (39%) than the Good solvers (23%) while problem solving, and that the resulting knowledge was more often incorrect. This is just what Cascade did, too.

Example studying took up a relatively small proportion of the time that subjects spent during the study. Not only were there only 3 examples compared to 25 problems, the subjects spent less time on average studying an example than solving a problem. The learning strategy of self-explanation was active only during example studying, so it comes as a surprise that such a proportionally small change in work habits caused such a large change in the amount learned. Perhaps the most important result from the Good/Poor simulations is an explanation for this counterintuitive finding. The simulations showed that only 40% of the rules learned by the good student simulation and not by the poor were learned during example studying. The others were learned during problem solving. This came as somewhat of a surprise to us. There were two basic reasons that self-explanation increases learning during problem solving.

---

[3] An impasse was identified as an outcome of an episode whenever the student believes that the next step that should be executed cannot be performed. Most (98%) of the impasses were identified by explicit statements such as "I don't know what to do with the angle," or "So that doesn't work either."

18

- The rules learned earlier allowed Cascade to travel down correct solution paths and reach impasses at places were it was indeed missing knowledge. Without these rules, the poor student simulation could not reach these productive impasses.

- The derivational triples acquired during rederivation of the example lines served as search control advice during problem solving, thus tending to keep the good student simulation on solution paths that led to productive, missing-knowledge impasses. The poor student simulation tended to wander off the solution paths, and reach impasses where there was nothing valuable to be learned.

It is doubtful that these interactions would have been discovered without a simulation as detailed as Cascade.

These results taught us about self-explanation per se, but the use of idealized student simulations leaves open the question of whether Cascade can actually model a real student. The next study tackles this question.

MODELING THE PROTOCOLS OF INDIVIDUAL SUBJECTS

The objective of the study reported in this section was to find out how close Cascade can come to modeling individual subjects. This study was undertaken in the same spirit as the ones in Newell and Simon (1972): Given a protocol, how closely can a simulation be fit to it? One difference between this study and those of Newell and Simon is that the task domain is physics, which is arguably a much richer task domain than the ones they studied. However, a more important difference is that considerable learning took place during our protocols.

A third difference is that our protocols are much longer than Newell and Simon's protocols, which made it impossible to employ their method of analysis. Each of the 9 subjects contributed protocols for 3 examples and 25 problems, so there were 252 protocols to analyze. Each protocol averaged about 12 pages, for a total of 3000 pages. Creating problem behavior graphs for all of them would be far too much work. Thus, part of the challenge in this study was to devise feasible methods for measuring the match between the behaviors of Cascade and the subjects.

19
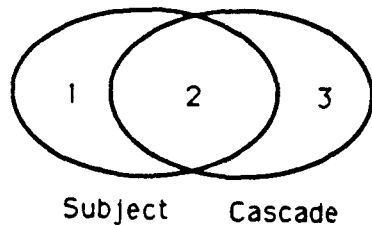
Figure 2: Matching the behaviors of Cascade and a subject

Figure 2 shows how the match between the behaviors of Cascade and a subject can be viewed. Region 1 represents behavior that the subject exhibited and Cascade did not. Region 2 represents the behaviors that are the same for both agents. Region 3 represents Cascade behaviors that the subject did not exhibit. The behaviors in region 3 have two sources. Some are computational expediencies: We couldn't get Cascade to do exactly what the subject did, so we had it do something else instead. That "something else" shows up in region 3. A Cascade behavior will also be put in region 3 if it is plausibly something that the subject did, but the protocol happens to show no signs of it occurring. For instance, it is known that not all cases of impasse-driven learning show up as hesitations or negative comments in protocols (VanLehn, 1991a). Whenever Cascade's impasse-driven learning is not reflected by overt signs of an impasse in the subject's protocol, that behavior is classified as region 3 behavior.

In all of the analyses presented below, we tried to determine two ratios: the amount of Cascade behavior that is matched by the subject (region 2 divided by the union of regions 1 and 2), and the amount of subject behavior that is matched by Cascade (region 2 divided by the unions regions 2 and 3). In order to make these comparisons, we had to find a way to count behaviors, which implies choosing a unit of analysis. This was not hard for the first ratio, because Cascade's behavior is well defined. For instance, we usually used a goal as the unit of analysis and

20

counted the number of goals generated by Cascade that were matched
or unmatched by the subject. It was not easy to determine a unit of
analysis for the other ratio, the percentage of subject behavior matched
by Cascade. A variety of units were used, depending on the type of
analysis being conducted.

Five analyses were conducted (see Table 4). Because we are more
interested in getting Cascade to simulate the subjects' acquisition of
physics rules than in getting it to simulate the chronology of their rea-
soning, four of the analyses ignored the order in which Cascade and
the subject made inferences. Both Cascade's behavior and the subject's
behavior were reduced to sets of inferences. Set intersections and differ-
ences were calculated, just as shown in Figure 2. However, we cannot
entirely ignore the chronology of inferencing, since the earlier study in-
dicated that analogical search control affects the location of impasses,
which in turn determines what can be learned during problem solving.
So a fifth analysis was conducted in order to see if the subjects' choices
during problem solving could be predicted by analogical search control.
After a description of how Cascade was fitted to the protocols, each of
these analyses will be presented.

Fitting Cascade

Fitting Cascade means setting values for parameters so that the pro-
gram's behavior matches the given subject's behavior as closely as pos-
sible. The parameters represent the products of cognitive processes that
are not modeled by Cascade, and yet Cascade's performance depends
on the outputs of these unmodeled processes, so they cannot be ignored
entirely.

There are two major types of parameters. The first controls initial
knowledge, which refers to the knowledge possessed by a student or Cas-
cade just prior to studying the examples. The student's initial knowledge
comes from reading the first several chapters of the textbook and from
their earlier studies of physics and mathematics. Cascade does not model
these processes, so it must be given an initial knowledge base. Cascade's
initial knowledge base was always a subset of a fixed "rule library." The
library consists of 3 buggy physics rules,[4] the 62 rules that constitute

---

[4] One buggy rule applies $F = ma$ to any force and not just a net force. Another

Table 1: Analyses comparing Cascade's behavior to the subjects' behavior

1. How many of Cascade's example studying inferences were also made by the subject?

2. How many of the subject's example studying inferences were also made by Cascade?

3. How many of Cascade's problem solving inferences were also made by the subject?

4. How many of the subject's problem solving inferences were also made by Cascade?

5. Do the search control decisions made by the subject match those made by Cascade?

the target domain knowledge, and the 45 non-domain rules mentioned earlier. Selecting an initial knowledge base can be viewed as setting 110 binary parameters, one for each rule in the library, where 1 means that the rule is included in the initial knowledge base, and 0 means that the rule is excluded.

The second type of parameter controls the depth of self-explanation. When studying examples, subjects choose to explain some lines but not others. Even when they do explain a line, they may explain it only down to a certain level of detail and decide to take the example's word for the rest. For instance, they might explain most of the line, $F_{n_1} = -F_g \cos(30)$, but not bother to explain where the minus sign comes from. Cascade does not model how the subjects decide which lines to explain and how deeply to explain them.[5] To simulate the output of this decision

asserts that the mass of a body is equal to its weight. The third assumes that the sign of all projections is positive.

[5] There are many possible reasons for why subjects do not explain everything. For instance, the subjects may feel that they already know everything that they could learn from explaining the line, or they may feel that explaining such details can be left until such time as they really need to know them. Deciding how deeply to explain

making process, extra propositions were entered into the descriptions of examples. Whenever Cascade is about to explain something, it first checks to see if accept(P) is in the example's description, where P is the thing it is about to explain. If an accept(P) is found, Cascade merely accepts P as explained without any further processing. Viewed as parameter setting, this amounts to associating a binary parameter with every explainable object, and setting it to 1 if it should be explained and 0 if it should be accepted.

The accept propositions are set by inspecting the subject's protocol. If the subject merely reads a line and says nothing else about it, then an accept proposition is entered for the whole line. If the subject omits discussion of a detail in a line, then an accept is placed around the Cascade goal that corresponds to that detail. In this fashion, the data completely determine which lines and parts of lines are explained by Cascade.

On the other hand, there is no way to easily determine what the subject's initial knowledge is. The whole protocol must be examined. As will be seen later, we sometimes made mistakes in selecting the initial knowledge. We should have fixed our mistakes, rerun the simulations and redone the comparisons of the program's output with the protocols. This will require months of work, so for this chapter, we are forced to report the analyses with our imperfect choices of initial knowledge left intact.

How many of Cascade's explanations are matched?

This section discusses the behavior of Cascade and the subjects as they explained examples. Goals were used as the basis for dividing Cascade's behavior into countable units. Each goal produced by Cascade was classified according to the method used to achieve it:

- Regular explanation: Cascade used one of the domain rules.

- Impasse with learning: Cascade reached an impasse, successfully applied an overly general rule, and learned a new domain rule via EBLC or analogical abduction.

---

a line is a fascinating topic for future research.

- Accept without explanation: The goal was not processed any further, but merely accepted as true without proof, because the example's description contained an "accept" proposition for it.

Aggregating across the simulation runs of all 9 subjects, there were 1121 goals. We located each of these goals in the subjects' protocols, and based on the talk surrounding them, classified them into the same three categories plus a new one:

- Impasse with learning: If the subject paused, complained about the goal or in some other way showed signs of being stuck, then we classified the goal as being achieved by impasse-driven learning.

- Regular explanation: If the subject merely mentioned the goal or its conclusions without any fuss, or the subject said nothing at all about this goal but did mention its subgoals, then we classified the goal as being solved by regular explanation.

- Accept without explanation: If the subject said nothing about this goal nor its subgoals, then the goal was classified as being accepted without explanation.

- Impasse and accept: Sometimes subjects clearly tried to explain a goal, but couldn't do it at all, so they just accepted the goal without proof. This is different from the other kind of acceptance, where the subject did not even try to explain the goal. It is different from the other kind of impasse because no learning occurs.

Table 5 shows the 1121 goals and how they were classified. Most ($1061 = 9 + 2 + 654 + 396$) goals were processed the same way by both Cascade and the subject, so 95% of Cascade's behavior was matched by subject behavior, which is highly significant ($p << .001$, Chi-squared test). In order to get a qualitative understanding of the shortcomings in Cascade's model of the protocols, each of the off-diagonal cells is discussed.

There were 7 cases where Cascade learned a rule and the subjects were coded as accepting the goal. All 7 cases occur at the same point, on a line where the example says, "Consider the knot at the junction of the three strings to be the body." Explaining this line causes Cascade

24

Table 5: Proportions of Cascade actions matched by subject actions

Cascade	Subjects				
	Impasse resolved	Impasse unresolved	Regular explanation	Accept silently	Totals
Impasse + EBLC	9	0	10	0	19
Impasse + Anal. Abduct.	0	2	0	7	9
Regular explanation	0	0	654	39	693
Accept w/o explanation	0	4	0	396	400
Totals	9	6	664	442	1121

to learn a new rule via analogical abduction. However, only 2 of the 9 subjects commented about this rule during example studying. The other 7 said nothing at all about the line wherein this rule would be learned, so they were coded as accepting the goal without proof. We could have made Cascade accept the goal as well, which meant that it wouldn't learn the knot rule. During later problems that had three strings converging on a knot, this would cause Cascade to reach an impasse and use transformational analogy. Unfortunately, Cascade's transformational analogy mechanism is not powerful enough to make use of the knot-is-a-body line in the example. When we increased its power so that it could use this line, it became too powerful and would draw analogies that were so far fetched that no subject would consider them. This led us to invent analogical abduction, which is a novel type of machine learning (see VanLehn & Jones, in press, for discussion). In order to test it out, it was included in Cascade. However, it is clear from this analysis that there are empirical problems with it. If an example's line really does cause analogical abduction, which is a form of impasse driven learning, then more subjects should have shown impasses. We now believe that transformational analogy is actually the source of transfer between the example line and problem solving, and that the two subjects who had impasses here should be classified as "impasse and accept," rather than as learning a new rule. As will be seen later, there are other signs that Cascade's model of transformational analogy is flawed.

There were 10 cases where Cascade learned a rule and the subjects were coded as doing regular explanation. There are two possible explanations of this discrepancy. Either the subjects really were doing impasse-driven learning, but they showed no signs of it in the protocol, or the subjects knew the rule already and were simply applying it here rather than learning it. Since the data are consistent with both explana-

25

tions, the interpretation depends on the prior probabilities of possessing the rules in question. All but one of the ten rules were not mentioned in the textbook, so they are less likely to be in the subjects' initial knowledge. The rule that is mentioned in the textbook determines the sign of a projection of a vector onto a negative axis. However, three subjects showed clear signs of impasses when this rule was first used, and three used a buggy version of the rule that always assigned a positive sign. The correct sign rule appears to be hard to learn and/or recall from the text for 6 of the 9 subjects, so it was probably not known by the other subjects either. Thus, because none of the rules involved seem likely to be in the subjects' initial knowledge, we suspect that all 10 cases in this cell of Table 5 correspond to impasse-driven learning events that were not displayed by the subjects.

There were 4 cases where the subject clearly tried to explain a goal but failed. On subsequent problems, the subject would explicitly refer back to these points in the examples and use transformational analogy. This is just what the two subjects who were coded as doing analogical abduction do, so that is why we now believe that there is no evidence for analogical abduction.

There are 39 cases where Cascade did regular explanation and the subjects were coded as doing accepts. Of these, 35 occurred when Cascade was trying to explain a force diagram. According to its rules, figuring out which forces exist and determining their directions are subgoals deeply embedded beneath the goal of drawing the force diagram. Some subjects discussed the forces without mentioning the force diagram. Either they were silently explaining many of the details of the force diagram, or they ignored the force diagram and "just knew" that it was important to explain the forces. We think the latter is more plausible, but we cannot easily model it without changing the goal structure embedded in Cascade's rules or Cascade's model of accepting the example's statements without proof.

Stepping back from the details, there are several results from the analysis in Table 5. It is clear now that Cascade needs several kinds of revisions: (1) Analogical abduction should be eliminated and transformational analogy strengthened. (2) The goal structure and/or the acceptance mechanism need to be revised in order to handle some subjects' explanations of free-body diagrams. (3) The subjects showed impasse-

like behavior on 4 occasions, but seemed to learn nothing from them. This is currently not an option with Cascade. It should be changed so that one of its responses to an impasse is just to accept the stuck goal as true without learning anything.

Another result concerns the handling of impasses. Cascade had 19 impasses where learning occurred, and of these, the subjects showed impasse-like behavior on 9. If we believe the codings, then 48% of the subject's impasses where learning occurred were visible in protocol data, and the other 52% were the victim of the usual incompleteness of protocol data. In the VanLehn (1991a) study of impasses in strategy discovery, there were 11 learning events, of which 8 (73%) were marked by impasse-like behavior on the subject's part. This is consistent with the impression that one gets from reading the protocols, which is that the subject in the VanLehn (1991a) study verbalizes much more of her thoughts than the subjects in the Chi et al. (1989) study. Thus, although the proportion of "silent" impasses is higher in the present study, it is not inconsistent with the earlier study's proposition.

How many of the subjects' explanations are matched?

The preceding section evaluated the match in one direction only, by seeing how much of Cascade's behavior is matched by subject behavior. This section evaluates how much of the subject's behavior is matched by Cascade behavior.

In order to do this, we extended an analysis by Chi and VanLehn (1991). They first coded every utterance in the example studying protocols as either a physics explanation, a mathematical explanation or one of several other kinds of utterances. They then coded each physics explanation into an if-then rule that presented the gist of the subject's comment in a uniform, more easily understood format. We extended this analysis by including the mathematical explanations as well and by correcting what we felt were a few minor mistakes in the earlier analysis of physics explanations.[6] Finally, we determined whether each rule apparently used by the subjects was also contained in Cascade's knowledge.

---

[6] It was usually clear what the action sides of these rules should be, but inferring the preconditions and the generality of the rules often required making some assumptions. Sometimes we disagreed with the assumptions made in the earlier analysis.

Table 6: Subject's explanations during example studying

Explanations	Categories	
143	Matches Cascade (63%)	
61	Outside task domain (27%)	
	23	Mathematical manipulations
	9	Editting part a to solve part b
	16	Extra example lines
	13	Units or terminology
	61	Total
23	Inside task domain (10%)	
	8	Incorrect explanations, retracted
	6	Acceleration and motion
	3	Abstract, partial plans
	0	Global planning from Chi et al.
	6	Miscellaneous, opportunistic
	23	Total
227	Grand total	

Some of this knowledge appeared explicitly as Cascade rules, while some of it was implicit in Cascade's rule interpreter (e.g., algebraic knowledge).

Of the 227 total explanation episodes found in the protocols, we determined that 143 (63%) were matched by Cascade explanations, while 84 were not. This indicates that Cascade models a large portion of the subjects' explanatory behavior. However, there are also many explanations that Cascade does not appear to account for. We categorized these explanations in order to determine whether Cascade should be expected to make them (see Table 6).

Of the 84 explanations, 61 concerned reasoning that was outside the domain of cognition being modeled. There were four classifications:

- 23 explanations concerned mathematical inferences that Cascade did not need to make because it had either been given the information in the problem statement (e.g., certain geometric information was provided to it), or it did not simplify its answers.

- 9 explanations occurred exclusively on part *b* of one example. Part *a* of this example describes a static situation where a string is hold-

ing a block on an inclined plane. Part *b* of the example asks one to find the acceleration of the block when the string is cut. Subjects explained part *b* by "editing" the explanation of part *a*. For instance, one editing rule was "If a force is removed from a static case and there is no friction, the body will move." Cascade does not reason about part *b* of the example in this manner. Rather, the system treats parts *a* and *b* as two distinct examples.

- Sometimes an example would contain a few lines that would emphasize aspects of the problem that were not germane to solving it or would discuss limiting cases (see Figure 1, line 3). Although we did not ask Cascade to explain these lines, the subjects sometimes would, and 16 of their explanations were of this type.

- 13 explanations involved miscellaneous comments about the examples that were judged to represent knowledge outside of the task domain as formalized in Cascade. For instance, we did not bother to model reasoning with units, so the statement, "In the English system, slugs are mass and pounds are weight," is considered outside the task domain.

The remaining 23 explanations are all relevant to the domain modeled by Cascade, so it should probably generate them. They fell into three classifications.

- 8 explanations appear to have been generated tentatively then retracted. They are all incorrect statements about physics, and the subjects seem to have revised their explanation a short time later. An example is, "If the two bodies in an Atwood's machine have the same acceleration, then they are not moving." How the subjects generated these conclusions is a bit of a mystery, although some are clearly the results of overly general rules. For instance, one subject said that one can calculate the tension in a string by adding the tensions in its parts. He probably generated this explanation by applying an overly general part-whole rule that works correctly for quantities such as volume, mass and weight.

- 6 explanations concerned the relationship between motion and acceleration. Of these, 4 explanations stated essentially the same

29

thing: "If a body moves, then it has a non-zero acceleration." Another explanation said, "If a body has motion on an axis, then it has acceleration on that axis," and the last explanation said, "If acceleration is zero, then nothing is moving." These all stem from the same incorrect conception of acceleration as speed, which is very common and hard to remove (Reif, 1987). Cascade should also be equipped with this misconception. Even though these new rules would alter Cascade's model of the subjects' explanation of examples, we do not expect that they would change Cascade's behavior on later problems. This is because the problems all deal with acceleration and do not mention concepts like velocity or "motion."

- 3 explanations articulated an abstract, partial plan for solving the problem. For instance, one explanation said "The weight of the block in the string example can be computed by figuring out the tension in the strings." This rule represents the top level of an abstract plan for determining the weight of the block. Cascade does not do hierarchical planning, but this rule provides evidence that perhaps it should. In particular, we would probably find more evidence for planning in the problem solving protocols. Planning did not have much of a chance to emerge during example studying because subjects are mostly led by the hand through the example solutions, so there is no real need to plan.

- 6 explanations defy classification, so they are simply listed below

    1. The body is the thing that the forces are acting on.

    2. Tension is important because it transmits the force between the blocks.

    3. The acceleration in an Atwood's machine is caused by gravity.

    4. If the right mass is greater than the left mass in an Atwood's machine, then the machine will accelerate downward.

    5. An upward force can act against gravity to keep a body from falling down.

    6. Most forces are gravitational.

30

All 6 explanations are true statements. However, they are not relevant to the goal of solving the problem, which is why Cascade did not make them.

This analysis indicates that of the 227 explanations uttered by subjects, 143 (63%) were matched by Cascade's explanations, 61 (27%) were outside the task domain being modeled, and 23 (10%) are explanations that Cascade should make but does not.

Cascade embodies a hypothesis about explanation, which is that explanation of solution lines in physics examples consists of rederiving them. This is a kind of local explanation, in that Cascade focuses only on the current solution line. It does not step back and try to see a global pattern that spans all the solution lines in an example. This may seem somewhat unusual, as other models of example explaining (e.g., VanLehn, 1990; Reimann, in press) emphasize global explanation. However, from one point of view, there is little point in global explanation of physics examples. Because later solution lines use results from earlier solution lines, doing local explanation of the later lines ties them together with the earlier lines, yielding an overall coherent structure. From another point of view, there is great benefit in global explanation, because it turns out that all the examples have a similar chronological structure: one first chooses bodies, then draws a diagram showing all the forces acting on the bodies, chooses coordinate axes, instantiates Newton's law along each axis, and solves the resulting system of equations. The textbook even mentions this procedure. One might expect the subjects to look for such a global, chronological structure, perhaps by first locally explaining all the solution lines, then reflecting on the whole solution to see if the overall structure made sense. However, the subjects produced only 3 global explanation statements. Seeing the overall chronological pattern in solutions does not appear to be a major concern for these subjects, perhaps because the logical structure suffices to make lines cohere. This is consistent with Sweller's work, which has shown that chronological patterns embedded in solutions are often overlooked when subjects are focussed on obtaining a goal (Sweller & Levine, 1982).

Another hypothesis about explanation embodied in Cascade is that all inferences are ultimately directed towards the top level goal of explaining the current solution line. Cascade uses a backwards chaining theorem prover, which means that it starts with the top level goal, chooses and

31

applies a rule, and then focuses on the first of the several subgoals cre-
ated by the rule's application. When all subgoals have been achieved, it
executes the rule, which finally draws a conclusion. This means that all
inferences are done in order to satisfy some goal, and that goal is ulti-
mately a subgoal of the top level goal. This makes Cascade a narrowly
focused, methodical explainer. It could be that people are more oppor-
tunististic and make observations (i.e., drawing conclusions) whenever
they notice that they can be drawn. In the extreme, they might do for-
ward chaining, drawing all possible conclusions about a problem while
paying no attention whatsoever to the solution lines or the overall goal
of the problem. However, only 6 of the 227 explanations appear to be
opportunistic, in that they were not matched by Cascade's goal-directed
inferences.

How much of Cascade's problem solving is matched?

We turn now to considering problem solving behaviors. The com-
parison between Cascade and the subjects is made difficult by two fac-
tors. First, the protocols are huge. There are approximately 2700 pages
of problem solving protocol, as compared to 300 pages of the example
studying protocol. Second, problem solving is less constrained than ex-
ample studying. Rederiving a line in a solution takes at the very most
only a few minutes, whereas solving a problem can take almost an hour.
Subjects seldom get badly lost while rederiving a solution line, whereas
when solving a problem, subjects often wander down several unproduc-
tive paths before finding a solution or giving up. Getting Cascade to
follow the subjects on a long doomed search path can be difficult.

An important methodological problem is finding a fair way to evaluate
the fit of a simulation and a protocol. Suppose one evaluated the fit
by counting the actions taken by the simulation that are matched by
subject actions, and dividing by the total number of actions taken by
the simulation. It often happens that the actions of the simulation and
the subject disagree at some point. This often causes them to diverge
and follow separate paths for a while, perhaps even a long while. The
longer the divergent paths, the worse the fit, even though the blame
is due to one false move earlier in all cases. Thus, simply comparing
matching to mismatching actions is unilluminating, for it confounds the

quality of the simulation with properties of the search space, namely, the lengths of certain paths.

Our approach is to equip Cascade with a variety of parameters, which we call "nudges," whose main purpose is to nudge Cascade back onto the subject's search path whenever it would otherwise wander off. The fit between the simulation and the subject's protocol is measured by counting the number of times the simulation had to be nudged. An additional benefit of nudging for measuring fit is that each nudge represents a piece of unexplained cognition. By taking a census of the nudges, one can rank types of unmodeled cognition and discover which ones are affecting problem solving behavior the most. Here are the types of nudges used:

- When Cascade solves a problem, it normally tries transformational analogy only after it tries regular domain knowledge. However, some subjects apparently prefer to use transformational analogy in some cases even when their behavior on earlier cases demonstrates that they have the appropriate domain knowledge and could potentially use it here. In order to force Cascade to follow the subjects, propositions of the form **trafo-only(G)** were placed in the problem's description whenever G is a goal that the subject preferred to achieve via transformational analogy. We call such cases of transformational analogy "forced."

- By default, the top-level goals of a problem require identifying a body and drawing a free-body diagram before finding the sought quantities. If the subject did not draw a free-body diagram, we eliminated these goals from the statement of the problem.

- On rare occasions, subjects came up with analogical mappings that Cascade could not generate. In such cases, we simply gave Cascade the subject's mapping or modified the problem representations so that the subject's mappings could be generated.

- Subjects did not always use the buggy $F = ma$ rule, which lets F be an individual force instead of the net force, when it was applicable (i.e., we could not figure out exactly what preconditions the subjects had for this rule). Perhaps they rightly believed that it was overly general, and thus they would only use it as a last resort. At

33

any rate, we controlled its usage by entering `ignore(F=ma_wrong)` into the descriptions of some problems but not others.

In order to evaluate Cascade's ability to model a given subject, nudges were entered by trial and error. Cascade's behavior then was compared to the subject's by classifying each of the goals in its trace according to how the goal was achieved. Four classifications were used:

- Regular solving: Cascade used one of the domain rules.

- Impasse with learning: Cascade reached an impasse, successfully applied EBLC. and learned a new domain rule.

- Transformational analogy: Cascade could achieve the goal with a domain rule, so it used transformational analogy.

- Forced transformational analogy: We forced Cascade to use transformational analogy even though it could have used a domain rule to achieve the goal.

Next, each of these goals was classified according to how the subject appeared to achieve it. Four classifications were used here as well:

- Transformational analogy. If the subject referred to an example and copied parts of it over, the goal was classified as achieved by transformational analogy.

- Impasse with learning. If the subject paused, complained about the goal or in some other way showed signs of being stuck, but the subject did not refer to an example to achieve the goal, then we classified the goal as being resolved by EBLC.

- Impasse and give up. On a few impasses, the subject just gave up without seeming to resolve the impasse or learn any new rules. Giving up is not one of the options available to Cascade for handling an impasse, although it should be.

- Regular solving: If the subject solved the goal without complaining, referring to an example or pausing for inordinate amounts of time, then we classified the subject as achieving the goal via regular problem solving.

34

Table 7: How many Cascade goals are achieved the same way by subjects?

Cascade	Subjects				
	Regular solving	Transform. analogy	Impasse with learning	Impasse and give up	Totals
Regular solving	3653	47	0	0	3700
Transform. analogy	16	176	0	4	196
Forced trans. analogy	0	35	0	0	35
Impasse with learning	7	0	9	0	16
Totals	3676	258	9	4	3947

Given these classifications, the results for all nine subjects appear in Table 7. In most cases ($3653 + 176 + 9 = 3838$ or 97%), the subjects handled goals in the same way that Cascade did, which was extremely unlikely to occur by chance ($p \ll .001$, Chi-squared test). Let us examine each of the other cells to see how serious the mismatching cases are.

There were 47 cases where the subjects did transformational analogy and Cascade did not. All these were due to the simplicity of Cascade's model of transformational analogy. One subject often used vector equations as if they were scalar equations, and applied them in creative, albeit incorrect ways that Cascade could not model. We let Cascade solve those problems in its normal way and counted all 42 goals as cases where the subject did transformational analogy and Cascade did not. The other 5 cases occurred when a subject could not recall some trigonometry rules, so she mixed transformational analogy with regular problem solving in a complex way that Cascade could not model. These 47 cases indicate that Cascade's model of transformational analogy needs improvement.

There were 16 cases where Cascade did transformational analogy and the subject seemed to do regular problem solving. Transformational analogies occur when Cascade is missing the knowledge to do regular problem solving, so there are two possible explanations for each case: Either the subject knew the rules that Cascade lacked, or the subject actually did have an impasse and resolved it with transformational analogy, but they didn't refer overtly to the example because they could remember the line that they needed, and thus were not coded as performing transformational analogy. Of the 16 cases, 8 seemed to be cases of covert transformational analogy because they involve accessing the free-body diagram, which is much easier to remember than the equa-

35

tions. Two more cases seemed to be covert transformational analogy, because the subject had already referred to the example's equation 3 times earlier, and probably had committed it to memory. Five cases seemed to be caused by the subject having a buggy rule about negative signs that Cascade lacked, but Cascade was able to get the same effect with transformational analogies. The last case is similar to the five just discussed, but with a different rule. Thus, of the 16 cases, 10 seem to be covert impasses correctly predicted by Cascade, and 6 seem to result from the subject having initial knowledge that is not in Cascade's standard initial knowledge base.

In 4 cases, the subjects reached impasses and gave up. Since Cascade currently cannot give up at impasses, these cases were approximated with transformational analogy. However, Cascade did succeed in predicting the location of the impasses.

There were 35 cases of forced transformational analogy. Two subjects (9 of 35 cases) always copied the free-body diagrams and never generated them on their own, so these subjects apparently were lacking knowledge about drawing free-body diagrams. In retrospect, these subjects should have been modeled by having their initial knowledge adjusted to remove the rules about drawing free-body diagrams. The other 26 cases occurred with subjects who clearly had the requisite rules, but chose to do transformational analogy instead. Of these 26 cases, 21 involved copying a free-body diagram rather than reasoning it out from physics principles and the other 5 involved copying trigonometry functions rather than figuring out whether the function should be sine, cosine or tangent, and what the angle should be. It is certainly easier to use transformational analogy for these particular cases, and apparently the subjects felt it was safe to do so, even though transformational analogy is fallible.

There were 7 cases where Cascade did impasse-driven learning and the subjects were coded as doing regular problem solving because they showed no signs of impasses. In general, there are two possible explanations for such cases. Either the subject actually did impasse-driven learning but failed to show any signs of it in their protocol, or they already had the rule that Cascade was missing so they just applied it instead of learning it. We examined each of the 7 cases to try and determine which explanation was most plausible for each. In some problems with friction in them, Cascade must learn four rules about friction forces.

36

One subject showed no signs of an impasse at any of these locations, so we suspect that she already understood friction forces. Another subject showed signs of an impasse at three of these places, but not at the fourth; it is likely that this fourth occasion was a covert impasse. Similarly, one problem required learning four rules about pressure forces. A subject showed signs of an impasse on only two of the four occasions, so it is likely that the other two occasions are silent impasses. Thus, of the 7 cases where Cascade does impasse-driven learning and the subjects appear not to, 3 seem to be silent cases of impasse-driven learning and 4 seem to be cases where the subject already knew the rules that Cascade learned.

From this examination of the mismatching cases, it seems that Cascade would need three augmentations in order to handle all the data. (1) Transformational analogy needs to be made more powerful so that it can model the more creative (albeit incorrect) usages exhibited by subjects. (2) The model should be free to choose transformational analogy instead of regular problem solving when it estimates that transformational analogy would be easier or more reliable than regular problem solving. (3) When Cascade cannot resolve an impasse, it should be allowed to give up. All the other cases of mismatching appear to be covert versions of the predicted events, or cases where rules should have been removed from the initial knowledge base.

In short, it appears that almost everything that Cascade does is matched by subject behavior. The next section analyzes the subjects' behavior in order to see how much of it is matched by Cascade.

How much of the subjects' problem solving behavior is matched?

In order to quantify how much of the subject's thinking during problem solving could be simulated by Cascade, we adopted the same unit of analysis that was used in the preceding analyses by converting the subjects' protocols into Cascade-sized goals. As an illustration of this analysis, Appendix 1 shows a protocol and our encoding of it. Following the tradition of Newell and Simon (1972), the protocol appears in the left column, and the encoding appears in the right column.

This kind of analysis is quite time consuming, so we could not do it

37

for all 252 protocols. Thus, we selected 4 protocols that we felt were typical. Two were from Good solvers, and two were from Poor solvers. Each of these pairs consisted of one protocol that was mostly transformational analogy and one that was mostly regular rule-based reasoning. (The protocol in Appendix 1 is a Poor solver who is doing mostly transformational analogy.) Clearly, this is too small a sample to draw strong inferences, but our point in this section is just to get a rough idea of the match.

Inferences occur whenever a goal is reduced to subgoals, or a goal is achieved. Thus, by literally reading between the lines, one can tell from the encoded protocols what the subjects' inferences were. In the four protocols, there were 151 inferences, excluding trivial arithmetic and algebraic ones. We examined each, and determined that Cascade could do all but 15 of them. That is, if we were to simulate these protocols with Cascade, we would need 15 new rules and would probably have to nudge it 15 times in order to get it to apply these rules. Thus, it appears that Cascade can model about 90% of the subject's inferences during problem solving.[7]

In order to give a qualitative sense of the behavior that Cascade could not simulate, we divided the inferences into ones that seemed outside of the intended task domain of Cascade and those that Cascade really should have modeled. Those that are outside the task domain are:

- In 2 inferences, the subjects checked their work by plugging their answers back into equations and seeing if the equations balanced.

- In 2 inferences, the subjects struggled to find the appropriate units for their calculations.

- In 2 inferences, the subject had difficulty understanding the diagram that accompanied the problem statement. In particular, it was difficult to decide whether a certain line stood for a string or not.

The inferences that were inside the intended domain were:

---

[7] Frankly, this estimate seems high to us. If we actually tried to add the requisite 15 rules to Cascade and simulate these protocols, we would probably find that the coverage was closer to 60% or 70%.

- 4 inferences were coded for a case where the subject decided he needed to understand freefall better, and went off to read the relevant page of the textbook, then decided that his (incorrect) solution to the problem was right anyway.

- 2 inferences were coded for a case where the subject decided to convert a vertical acceleration to one parallel to an inclined plane, but apparently did not realize that projection could be applied directly, so he "converted" it to a force using $F = ma$, projected the force, then converted it back.

- 2 inferences involved a subject who let $g = -9.8$ for no apparent reason. This could have been an unintentional error, except that the subject noticed later that g was negative and did not correct it. The second inference occurred later, when she dropped a negative sign "for fun" as she put it, thus effectively canceling her earlier error and obtaining a correct answer.

- 1 inference was coded for a subject who invented something he called a "double force" that included both gravitational and frictional influences.

Although these lists would clearly be much longer if more protocols had been analyzed, and the small sample makes any statistical inferences unsound, taking the data at face value indicates that about a third of the unmatched behavior is outside the task domain, and the other two thirds is behavior that Cascade should model. Moreover, most of the behavior that Cascade should model is incorrect reasoning of a wide variety of types. More research is needed before we can conclude anything about the sources of these incorrect inferences.

Control choices

The preceding analyses assessed what was done, but ignored the order in which actions took place. This section concerns the overall control structure as well as the local choices of which rule to try first in achieving a goal. Both these factors control the order in which inferences take place.

Cascade uses a backwards chaining control structure. A goal is processed by selecting a rule, then posting any subgoals required by that rule. After the subgoals have been achieved, the rule's conclusion is asserted. Thus, a goal should show up twice in a protocol: when it is first posted and when it is completed. In our protocols, subjects did not usually talk about their goals when they posted them (see Appendix 1), although they often mentioned the conclusions that were made when a goal was achieved. This could be taken as a sign that they were not following a backchaining control structure. However, a control structure also restricts the order in which actions can take place. For instance, if A and B are subgoals of C, and D is not, then the order A.D.B cannot occur. Thus, the ordering in which goals are achieved is diagnostic of the control structure.

As part of the analysis in the preceding section, we fit a backwards chaining goal structure to the subjects' behavior in the four protocols analyzed. Of the approximately 151 goals, there were three cases where backwards chaining would not fit. In two, the subject performed goals prematurely (i.e., the A.D.B case just mentioned). During the third case, the subject explicitly decided which of two conjunctive (sibling) goals to do first. This kind of search control occurs in some means-ends analysis problem solvers (e.g., Prodigy: Minton et al., 1989) but not in Casca'e. In short, the available evidence indicates that Cascade's control structu e is not a bad first approximation to the subjects' overall approach.

The only search control decision made by Cascade is which rule to apply given that more than one matches the current goal. Two factors determine Cascade's choice of rule. If analogical search control can find an old goal that is isomorphic to the current goal, then the old goal's rule is chosen. If analogical search control does not apply, Cascade selects rules in the order in which they appear in a file. This file is set up to generate efficient behavior in general, and is not tuned for any particular subject.

The first analysis involves placing all goals in one of two classes: Either the first rule selected for achieving this goal was ultimately rejected and another rule was used in its place in the final solution, or the first rule selected was used in the final solution. This categorization was carried out for all Cascade goals and for all subject goals corresponding to Cascade goals. Of the 3461 cases where Cascade picked the correct rule

40

first, the subject failed to pick the correct rule first in only 81 (2.3%) cases. Thus, Cascade predicted the subject's choice of rule in almost all (97.7%) cases.

In order to determine how much of this success is due to analogical search control, we split the 3461 cases according to whether Cascade's choice was determined by analogical search control. Cascade selected the correct rule first *without* the involvement of analogical search control 2702 times. In 62 of these cases, the subjects did not choose the correct rule first. Thus Cascade's default rule ordering predicted the subjects' rule choices 97.7% of the time. Cascade picked the correct rule first *with* its analogical search control mechanism 759 times, and agreeing with the subject's choice in all but 19 cases, for a success rate of 97.5%.

Initially, this seemed a disturbing result, because it appeared that analogical search control gives the model no predictive accuracy over the default rule ordering. This raises the question of whether Cascade would be better off without analogical search control. If it always used its default rule choice, would its overall prediction accuracy rise? Rather than simulate all 9 subjects with analogical search control turned off, we estimated what the fit would be. We gave Cascade all the knowledge necessary to explain and solve the examples and problems (so no impasses would be generated) and ran it on all the examples and problems. While running, it kept track of how many times it used analogical search control to choose a rule, and how many times that choice corresponded to the default rule choice. We found that analogical search control led to a choice different from the default rule approximately *12% of the time.* This implies that if Cascade had been run with search control turned off, then about 89 (12% of the 740) cases where analogical search control predicted the subjects' rule choice would now become cases where its predictions fail. In addition, hand analysis of the 19 cases where analogical search control failed to predict the subject's choices indicates that only 2 cases would be successfully predicted if analogical search control were turned off. Thus, if Cascade had only its default search control, it would mispredict 106 cases that it successfully predicted with analogical search control turned on, so its accuracy would drop to 95.6% as opposed to 97.7% with analogical search control enabled. Thus, analogical search control does help.

In order to further understand why analogical search control failed

41

to predict 19 rule choices. each was analyzed. All were generated by 2 subjects, so Cascade's analogical search control predicted the rule choices for 7 of the subjects with 100% accuracy. Moreover, it turns out that in 11 of the 19 cases. the first inference made by the subject could not be modeled by any Cascade rule. Such cases indicate an inaccurate model of the subject's knowledge, rather than an inaccurate model of the subject's search control.

In summary, our initial result appeared to suggest that analogical search control provided no closer match between Cascade's problem solving behavior and the subjects' than Cascade's normal problem solving did. However, for 7 of the 9 subjects, Cascade provides a clear improvement in matching the subjects when analogical search control is used. For the other two subjects, the failure to match appears to arise from missing prior knowledge rather than a defect in the learning mechanism. Moreover, if analogical search control is turned off, Cascade's prediction accuracy would drop.

Discussion of the fit between Cascade and individual subjects

There were two purposes in fitting Cascade to the individual subjects. The first was to find out what the subjects were doing. and the second was to find out how well Cascade could model that. We discuss these objectives together. first looking at example-studying behavior. then problem solving behavior.

The two major processes during example-studying were explanation of a line and acceptance of a line or a part of a line without explaining it. Cascade's model of self-explanation is to rederive the line via ordinary deduction. Cascade's backwards chaining control structure ensures that only inferences relevant to the top goal are made. This sufficed to model 63% of the subjects' 227 explantions (see Table 6). Cascade's model of accepting a line was simply to prune a whole subtree of an explanation by accepting a goal as achieved without trying to achieve it. This sufficed to account for 92% of the subjects' 422 cases of acceptance (see Table 5); actually. the lack of fit may be due to the goal structure implicit in Cascade's rules rather than the acceptance mechanism per se. Overall, Cascade accounts for about 75% of the subjects' behavior during example

42

studying.[8]

The self-explanations that Cascade does not model are mostly (73%) concerned with cognitive skills that we are not interested in modeling, such as algebraic equation solving. That left only 23 explanations that Cascade really should have modeled. These fell into two groups: incorrect explanations (14 cases) and more general comments (9 cases) including abstract, partial solution plans and observations such as, "The tension is important because it transmits force between the blocks of an Atwood's machine." The first group indicates that Cascade needs more buggy rules than it currently has. In particular, it needs to model misconceptions about acceleration and motion. The second group indicated that the subjects have an ability that Cascade lacks. They can stand back from the details and abstract an overall view of either the solution (i.e., they see an abstract plan or chronological pattern in the inferences) or the system (i.e., they form a mental model of the mechanical device). Although these are certainly interesting and important types of cognition, they appeared surprisingly rarely in this study (only 9 of 227 cases, or 4%). When the Cascade research began, we expected plan recognition to be the most important kind of self-explanation. We have since learned that it occurs rarely and may have little influence on subsequent problem solving.[9] Overall, it is good news that only 23 (14%) of the 166 interesting, task-domain relevant explanations uttered by subjects require extensions to Cascade in order to model them. Even in its present form, Cascade successfully models the bulk of the subjects' self-explanations.

---

[8] The coverage figure for example studying was calculated as follows: Table 6 shows that 63% of the subjects' self-explanations are modeled by Cascade. Table 5 shows that 92% of the acceptances are modeled by Cascade. However, these tables use different units. From Table 5, we can estimate that about 40% of the subjects' behavior was acceptances, so we can use that figure to form a weighted average of the two coverages, and thus calculate that about 75% of the subjects' example studying behavior is matched by Cascade.

[9] Although it seems pointless with only 3 cases of abstract planning in the example studying protocols, we could analyze the problem solving protocols to see if these subjects' rule choices during the relevant sections of their protocol are better explained by the abstract plans they found during example studying than by the existing search control mechanisms of Cascade. Because analogical search control probably makes the same predictions about rule choices as an abstract plan, we doubt that this analysis would yield unequivocal results.

43

The two most common problem solving processes were backwards chaining rule-based inference and transformational analogy. We were surprised by the prevalence of transformational analogy during problem solving, although it was certainly due in part to the fact that 12 of the 21 problems in the study were isomorphic (or nearly so) to one of the three examples (i.e., there was a set of "string" problems, "incline" problems, and "pulley" problems). Although only around 6% of subjects' problem solving involved transformational analogy (see Table 7), it often had a profound affect on the direction of the subjects' search. Cascade has a simple model of transformational analogy, but it was not powerful enough to handle all the cases. Subjects sometimes find analogical mappings that Cascade cannot. They sometimes mix transformational and regular problem solving (47 cases). They sometimes prefer to use transformational analogy even when they do not have to use it (35 cases). Many of these problematic cases occur when subjects need to draw a free-body diagram and refer to the example's free-body diagram for help. They may be using well-honed skills for visual analogizing. This would explain why Cascade's transformational analogy, which is oriented towards analogical transfer of equations, is so incomplete.

On the whole, it appears that most of the example-studying and problem-solving behavior can be explained as deduction, simple acceptance of example statements, and transformational analogy. Although these three processes cover only 75% of the example studying behavior and 60-90% of the problem solving behavior, the behavior they do not cover mostly involves mathematical manipulations and other types of cognition that are outside the domain of study.

We were surprised to find that the overall control structure and local control choices were also modeled rather accurately by Cascade (see the preceding section). However, we did not stress this aspect of Cascade either during it development nor during its evaluation, so there is probably much room for improvement in both areas.

GENERAL DISCUSSION

We have completed three steps of a four-step research program. The first step was to find a computationally sufficient account for the knowledge acquisition that occured in the Chi et al. study. The major techni-

cal hurdle was finding a way to constrain search during problem solving so that impasses would occur at the right places. This was achieved by adding analogical search control, a form of symbol-level learning. There was no way to tell in advance of running Cascade whether analogical search control was sufficient. Fortunately, it was, and Cascade was able to learn all the rules that it needed to learn. Moreover, the problem of getting impasses to occur in the right places is faced by all impasse-driven machine learning systems, so this result is relevant to many machine learning systems. A minor hurdle was finding a way to transfer knowledge from the knot-is-a-body example line to problem solving. After trying several methods, we discovered a new machine learning technique, which we called analogical abduction.

The second step in the research program was to demonstrate that Cascade could explain the main findings of the Chi et al. study. As a model of the self-explanation effect, Cascade was qualitatively adequate. It could self-explain example lines as well as just accept them. It could solve problems with and without referring to examples, and its analogical references can both dive into the middle of the example to pick out a single fact (analogical search control) or read the example from the beginning searching for a useful equation (transformational analogy). In order to go beyond qualitative similarity, simulations were conducted that modeled an idealized good student and an idealized poor student. All four of the main findings from the self-explanation study were reproduced in the contrast between the two simulations. A particularly suprising result was that most of the learning occurred during problem solving even though the particular learning strategy we manipulated, self-explanation, operated only during example studying. Examination of Cascade's processing showed that the acceleration of its learning during problem solving was caused by (1) analogical search control obtaining more guidance from the experience (derivation) left behind by self-explaining the examples, and (2) problem solving having prerequisite knowledge, obtained during example studying, that allowed it to reach impasses where learning could appropriately take place.

The third step in the research program was to demonstate that Cascade can simulate real student cognition at the 5 to 10 second unit of analysis. We fit Cascade to subject protocols by forcing it to explain exactly the same lines as the subject and to do transformational analogy

at exactly the same points as the subject. We also gave it initial knowledge that approximated the subject's knowledge just prior to explaining the examples. Setting these parameters, plus occasionally "nudging" the system, sufficed to fit Cascade to cover most of the subjects' behavior. Such fitting was carried out for all 9 subjects, and the resulting match between system and subject behavior was evaluated. As one might expect, given that Cascade was designed to model the subjects, almost everything it did was also done by the subjects. Tables 5 and 7 show that over 95% of Cascade's goals occured in the subjects' behavior and were achieved the same way by both simulation and subject. On the other hand, when the subjects' behavior is analyzed in terms of goals and inferences, about 75% of their example-studying behavior and between 60% and 90% of their problem-solving behavior is matched by Cascade goals and inferences. Most of the unmatched behavior concerns mathematical manipulations and other skills that Cascade was not intended to model. It was found that the main inadequacy in Cascade is its simple model of transformational analogy. Subjects were quite clever at forming useful analogies with the examples, and especially their free-body diagrams.

The fourth step in the research program is to use the fitted models of individual subjects to find out more about their learning. Unfortunately, we discovered during the current fitting that some of our assumptions about initial knowledge were incorrect, so we will have to rerun the fitting exercise with new assumptions before conducting these analyses. Nonetheless, a few speculative remarks can be made on the basis of the existing analyses.

We were surprised that there were so few clear-cut cases of impasse-driven learning in the protocols. Tables 5 and 7 show that the subjects had clear signs of impasses on only 18 occassions when Cascade did EBLC. Although analyses to be conducted later will tell us exactly why there were so few clear cases of learning, it appears that it is due to overuse of the free-body diagrams. Most of the rules learned by the idealized good student simulation are used during the initial stage of solving a problem, when a situation is analyzed and the forces and accelerations are found. The examples cover this phase by merely presenting the free-body diagram and perhaps adding a few lines of explanation for any forces that they consider unobvious. As a consequence, most subjects simply accepted the free-body diagrams without trying to explain them,

46

and thus missed the opportunity to learn. During problem solving, the diagrams were again overused. This time they subjects tended to use transformational analogy to adapt an example's free-body diagram instead of figuring one out from their knowledge of physics. Thus, they would miss the chance to do impasse-driven learning. This problem was exacerbated by the fact that most of the 25 problems were deliberately constructed so as to have free-body diagrams that were similar to ones in the examples. Subjects who used transformational analogy for these problems tended to get them right. This meant that subjects could learn very little and yet still get high scores. For instance, one subject who did not know about normal forces nonetheless got all of the "normal force" problems right. In short, it appears that overuse of the diagrams, exacerbated by the design of the study's problem set, reduces the number of cases of impasse-driven learning.

On the other hand, it could also be that subjects had instances of impasse-driven learning, but they showed no overt signs of them. Most of the cases of overt impasse-driven learning came from just two subjects who were the most vocal of the subjects. It is likely that the other subjects had episodes of impasse-driven learning but did not report them. We had hoped to detect these silent impasses by seeing changes in the subject's behavior. That is, we had hoped to see one or more occasions where the to-be-learned rule could be applied but was not, followed by a solid string of occasions where the rule was applied. The learning event would be somewhere in the vicinity of the transition from non-usage to usage. This type of analysis succeeded in locating silent learning events in Tower of Hanoi protocols (VanLehn, 1991a) and finger counting protocols (Siegler & Jenkins, 1989; Jones & VanLehn, 1991). Unfortunately, most subjects in this study either always used a rule or always avoided it. In some sense, they had to do this. Subjects in the two earlier studies learned new alternative strategies to solve a problem that they could already solve. Thus, if they did not use one of the to-be-learned rules, they always had their old rule to use instead. This was not the case in the present study. If a rule was missing for a new kind of force, for example, then the subject's only alternative to learning a new rule was to use a transformational analogy. Once they have successfully used transformational analogy for this appearance of the new force, they would tend to use it for all other appearances. In this fashion, they would miss

47

the opportunity to learn new rules. In short, if a subject was to learn a rule, they tended to learn it on the first occasion that it was possible to learn it. If they used an alternative to the rule, then they tended to keep using that alternative through the end of the study. Thus, we found few transitions from not using a rule to using a rule, and we have little solid evidence for silent impasses.

In short, it appears that there is less learning in the protocols than we had hoped, although this might be partly an artifact of our inability to detect silent impasses. Nonetheless, there is much to be learned by examining the instances of learning that did occur. For instance, it would be good to find out if analogical search control really did play a role in guiding the subjects to an appropriate impasse.

Cascade is based on the assumption that the self-explanation effect is due solely to a difference in example-studying habits rather than a difference in prior knowledge. Surprisingly, this assumption held up even during the fitting of individual protocols. However, we expect some challenges to arise during the next set of analyses. It may be that the subjects' policies about using transformational analogy are just as important as self-explanation in determining whether learning will take place. We suspect that effective learning requires both that the subject explain an example and that they try not to refer to it during problem solving for purposes of obtaining a free-body diagram or an equation. On the other hand, referring to the example for advice on which rule to choose (analogical search control) should be encouraged. Methodologically, the complexity of this speculative prescription shows the advantage of simulation-based analyses of behavior. A prescription based on just the Chi et. al study would be simpler and perhaps not as effective.

Cascade shows promise as a general model of cognitive skill acquisition, but it needs considerable work beyond fixing its model of transformational analogy. In order to be a more complete account of the phenomena at hand, it needs a model of analogical retrieval and of the difference between physical and mental references to the examples. We believe the existing mechanisms can also handle some well-known phenomena of skill acquisition, such as practice and transfer effects, but this needs to be demonstrated. The major limitation on the generality of Cascade 3 is its use of monotonic reasoning. With the help of Rolf Ploetzner, we are currently incorporating a version of the situation

48

calculus which will greatly enhance the types of reasoning Cascade can model. and thus the number of task domains that it can model. We are encouraged to extend Cascade to become a more complete, more general model of learning by its similarity to other theories of cognitive skill acquisition (e.g., Anderson, 1990: Schank, 1986). It is considerably simpler than those theories and probably more thoroughly implemented and tested. We hope that its simplicity and empirical adequacy remain intact as it is extended.

Finally, these results shed some light on the possibility of using machines to acquire knowledge for expert systems from ordinary human instructional material. Any AI expert would suspect that machines would have a hard time learning from human materials because they lack common sense. It turns out that common sense was important for Cascade's learning. but it was not particularly hard to provide it. Common sense was encoded in the non-domain knowledge given to Cascade as part of its initial knowledge base. Most of the non-domain knowledge concerned geometric reasoning, common-sense physical reasoning about pushes and pulls, and most significantly, overly general rules. This knowledge was used during EBLC to form new physics-specific domain knowledge. Hence, common sense knowledge was crucial because it heavily constrained learning. On the other hand, it was not particularly hard to figure out what that knowledge should be. Whenever Cascade would reach an impasse that it could not resolve with its existing common sense knowledge, it was usually quite simple to specify that knowledge. After all, it is common sense.

REFERENCES

Anderson. J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson. J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review. 94*(2). 192–210.

Anderson. J.R. (1990) *Adaptive Control of Thought.* Hillsdale. NJ: Lawrence Erlbaum.

Anderson. J. R.. Farrell. R. G., & Saurers. R. (1984). Learning to program in LISP. *Cognitive Science, 8,* 87–129.

Anderson, J. R., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Combridge, England: Cambridge University Press.

Bielaczyc, K., & Recker, M. M. (1991). Learning to learn: The implications of strategy instruction in computer programming. In L. Birnbaum (Ed.), *The International Conference on the Learning Sciences* (pp. 39-44). Charlottesville, VA: Association for the Advancement of Computing in Education.

Bundy, A., Byrd, L., Luger, G., Mellish, C. & Palmer, M. (1979) Solving mechanics problems using meta-level inference. In B. Buchanan (Ed.), *Sixth International Joint Conference on Artificial Intelligence* (pp. 1017-1027). Los Altos, CA: Morgan Kaufmann.

Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.) *Machine Learning. An AI Approach: Vol. 2*. Los Altos, CA: Morgan-Kaufman.

Charney, D., Reder, L., & Kushit, G. (1990). Goal setting and procedure selection in acquiring computer skills: A comparison of tutorials, problem-solving, and learner exploration. *Cognitive Science, 7, 323-342*.

Chi, M. T. H., de Leeuw, N., Chiu, M., & LaVancher, C. (1991). *The use of self-explanations as a learning tool*. Manuscript submitted for publication.

Chi, M.T.H., Bassok, M., Lewis, M.W., Reimann, P. & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science. 13, 145-182*.

Chi, M.T. H. & VanLehn, K. (1991) The content of physics self-explanations. *Journal of the Learning Sciences. 1(1), 69-106*.

Chi, M. T. H., VanLehn, K., & Reiner, M. (1988). *How are impasses resolved while learning to solve problems*. Paper presented at the 29th meeting of the Psychonomics Society, Chicago.

Dietterich, T.G. (1986) Learning at the knowledge level. *Machine Learning. 1, 287-316*.

Ferguson-Hessler, M.G.M. & de Jong, T. (1990). Studying physics texts: Differences in study processes between good and poor solvers. *Cognition and Instruction. 7, 41-54*.

Jones, R. M. (1989) *A model of retrieval in problem solving*. Doctoral

dissertation. Information and Computer Science, University of California, Irvine.

Jones, R. M., & VanLehn, K. (1991). Strategy shifts without impasses: A computational model of the sum-to-min transition. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 358-363). Chicago: Lawrence Erlbaum.

Larkin, J. (1983) The role of problem representation in physics. In D. Gentner & A. Collins (Eds.) *Mental Models*, Hillsdale, NJ: Lawrence Erlbaum.

LeFevre, J., & Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, *3*, 1-30.

Lewis, C. (1988) Why and how to learn why: Analysis-based generalization of procedures. *Cognitive Science*, 12, 211-256.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, *40*, 63-118.

Newell, A. (1990) *Unified theories of cognition*, Cambridge, MA: Harvard University Press.

Novak, G. S., Jr., & Araya, A. (1980). Research on expert problem solving in physics. In T. Dietterich & W. Swartout (Eds.), *Proceedings, Eighth National Conference on Artificial Intelligence* (pp. 465-470). Los Altos, CA: Morgan Kaufmann.

Pirolli, P. (1987). A model of purpose-driven analogy and skill acquisition in programming. In *Proceedings of the Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.

Pirolli, P. & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum.

Reif, F. (1987). Interpretation of scientific or mathematical concepts: Cognitive issues and instructional implications. *Cognitive Science*, *11*, 395-416.

Reimann, P. (in press). Modeling active, hypothesis-driven learning from examples. In E. De Corte, M. Linn, H. Mandl, & L. Verschaffel (Eds.), *Computer-based learning environments and problem solving*. Berlin: Springer.

Schank, R.C. (1986) *Explanation Patterns: Understanding Mechani-*

cally and Creatively. Hillsdale, NJ: Lawrence Erlbaum.

Siegler, R. S., & Jenkins, E. (1989). *How children discover new strategies*. Hillsdale, NJ. Lawrence Erlbaum.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*, 59-89.

Sweller, J., & Levine, M. (1982). Effects of goal specificity on means-ends analysis and learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 8*(5), 463-474.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions*. Cambridge, MA: MIT Press.

VanLehn, K. (1991a). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science, 15*, 1-47.

VanLehn, K. (1991b). Two pseudo-students: Applications of machine learning to formative evaluation. In R. Lewis & S. Otsuki (Eds.), *Advanced research on computers in education*. New York: North-Holland.

VanLehn, K., Ball, W. & Kowalski, B. (1990). Explanation-based learning of correctness: Towards a model of the self-explanation effect. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum.

VanLehn, K., Jones, R.M. & Chi, M.T.H. (in press). A model of the self-explanation effect. *Journal of the Learning Sciences*

VanLehn, K. & Jones, R.M. (in press). Integration of analogical search control and explanation- based learning of correctness. In S. Minton & P. Langley (Eds.) *Planning, Scheduling and Learning*. Los Altos, CA: Morgan Kaufman.

Ward, M., & Sweller, J. (1990). Structuring effective worked examples. *Cognition and Instruction, 7*, 1-39.

APPENDIX

This appendix contains an example of a subject's protocol encoded at the level of Cascade-like subgoals. The protocol appears in the left column, and the encoding appears in the right. An "(R)" in the left column indicates that the subject is reading the problem aloud. The coding procedure consisted of pretending that the subject was a version of Cascade, and generating the problem-solving trace that this version

of Cascade would have to generate to lead to the utterances found in the protocol. In order for the subject's actions to fit into Cascade's control structure. it was sometimes necessary to hypothesize problem-solving goals that are not verbalized.

The Cascade model contains four types of goals:

- value(X): find a value for the quantity X.

- solve(X=Y): find a solution to the equation X=Y.

- retrieve(example): find an example similar to the current problem.

- equation(X): find an equation that can be used to compute a value for X.

Each goal appears with an "S:" when the goal is posted and an "F:" when a solution to the goal has been found. In addition. the model sometimes must explicitly "backtrack" over some subgoals to account for backtracking behavior by the subject.

The coding process allowed us to fit the hypothetical Cascade model as closely as possible to the subject's behavior. In doing this, we were able to identify specific locations at which the current implementation of Cascade would fail to generate the subject's behavior. At these locations. we would have to manually "nudge" the system onto the correct reasoning path. Events of this type are marked with event numbers in the right-hand margin.

This particular protocol concerns a "poor" subject solving a combination pulley-incline problem. The subject initially retrieves the pulley example and copies the equations for tension and acceleration from that example. For the most part. the subject attempts to directly apply these equations to the current problem (a strategy that will lead to an incorrect solution). The subject computes the tension of the string and the acceleration of the free-hanging block in this manner. However. to compute the acceleration of the block on the incline. the subject assumes his result for acceleration from the copied equations is actually the projection of an acceleration vector that points down the incline. The subject apparently does not have a rule for computing the projection of an acceleration vector. so he converts it to a force vector by multiplying the acceleration by the mass of the block ($F = ma$). He then computes the projection

of the force vector and converts the result back to an acceleration with $F = ma$ again.

Okay.	
(R) A block of mass m1 equals 3.0 slugs	
on a sss...smooth	
incline plane of angle 30 degrees is	
connected by a cord	
over a small frictionless pulley to a	
second block of mass	
m2 equals 2.0 slugs hanging vertically.	
	Soughts: accel(m1), accel(m2),
	tension(cord)
(R) What is the acceleration of each	S: value(accel(m1)), value(accel(m2))
body?	
Okay.	
That would just...	
Ahhh...	
Okay, the acceleration...	
That would be M...with...that would be	S: retrieve(example)
like the pulley	F: retrieve(example): px
again...this equation. (mumbles)	S: equation(m1), equation(m2)
This one.	S: equation(m1)
	F: equation(m1): T-m1g=m1a
	S: equation(m2)
	F: equation(m2): T-m2g=m2a
	F: equation(m1), equation(m2):
	T-m1g=m1a, T-m2g=m2a
Okay.	
Ummm...	
Okay.	
Wait.	
If...	
Find the tension of the whole...	
Okay, I want to find the tension of the	
whole thing again,	
and then I can find the acceleration	
for each one, and use	
that to find force for ehh...for the	
second...well I'll use	
it.	
Okay, if I use that to find the	+ S: solve(accel(m1)= (1)
component of the force like	Fn(proj(accel(m1))))),
for the F...	solve(accel(m2)=
	proj(accel(m2)))
The Y component for this m1 block on	S: value(Fn(proj(accel(m1)))))),
the slant, and find the	value(proj(accel(m2)))
whole force, then I'll find the whole	S: value(T) [value(tension(cord))]
acceleration for that.	
Okay.	
So that would be T equals 2 times M...	S: equation(T)
Two times mass one is 3.0 times mass	F: equation(T): ((2m1m2)/(m1+m2))g
two is 2.0 over mass one	S: solve(T=((2m1m2)/(m1+m2))g)
plus two, is 3.0 plus 2.0...time	S: value(m1)
slugs...	F: value(m1):3.0
	S: value(m2)
	F: value(m2):2.0
	S: value(g)
	+ S: value(system(problem)) (2)
What system is slugs?	+ S: value(system(slugs)) (3)
Where is that table again?	
Here.	
Slugs is feet and all.	F: value(system(slugs)):
So that would be...	british
	F: value(system(problem)):
	british

Gravity would be 32 feet per second
squared.

Okay, so T equals 2 times 3 is 6.

Time...12 ( ) is 36 times 2 is 12.

12...time 32 equal...divided by 5.

Equals 76.8.

Okay.
And the acceleration...
For m1 the accel...
I'll do m2 first.
Acceleration for m2 is the lighter
one, would be...
Ahh...T minus m1Gm2V equals m2A...
76.8 minus 2 times 32

equals 2A.
2A equals 76.8 minus 64 would be 12.8.

A would be... 6.4.

And for...
m1...

T minus m1G equals negative m1A

Will be 76.8 minus 3 times 32 equals

```
F: value(g): 32

S: solve(T=((2*3.0*2.0)/
           (3.0+2.0))*32)
  S: value((2*3*2)/(3+2))*32)
    S: value((2*3*2)*32)
      S: value(2*3*2)
        S: value(2*3)
        F: value(2*3): 6
        S: value(6*2)
        F: value(6*2): 12
      F: value(2*3*2): 12
      S: value(12*32)
      F: value(12*32): 384
    F: value((2*3*2)*32): 384
    S: value(384/(3+2))
      S: value(3+2)
      F: value(3+2): 5
      S: value(384/5)
      F: value(384/5): 76.8
    F: value(384/(3+2)): 76.8
  F: value((2*3*2)/(3+2))*32): 76.8
  F: solve(T=((2*3.0*2.0)/
          (3.0+2.0))*32): T=76.8
 F: solve(T=((2m1m2)/(m1+m2))g):
              T=76.8
F: value(T): 76.8
S: value(Fn(proj(accel(m1))))
Backtrack                      (4)
S: value(proj(accel(m2)))

 S: solve(T-m2g=m2a)
      [a=proj(accel(m2))]
  S: value(T)
  F: value(T): 76.8
  S: value(m2)
  F: value(m2): 2
  S: value(g)
  F: value(g): 32
  S: solve(76.8-2*32=2*a)
   S: value(76.8-2*32)
     S: value(2*32)
     F: value(2*32): 64
     S: value(76.8-64)
     F: value(76.8-64): 12.8
   S: value(76.8-2*32): 12.8
   S: solve(12.8=2*a)
    S: solve(a=12.8/2)
     S: value(12.8/2)
     F: value(12.8/2): 6.4
    F: solve(a=12.8/2): a=6.4
   F: solve(12.8=2*a): a=6.4
  F: solve(76.8-2*32=2*a): a=6.4
 F: solve(T-m2g=m2a): a=6.4
F: value(proj(accel(m2))): 6.4

S: value(Fn(proj(accel(m1))))
 S: value(proj(accel(m1)))
  S: solve(T-m1g=-m1a)
           [a=proj(accel(m1))]
   S: value(T)
   F: value(T): 76.8
   S: value(m1)
   F: value(m1): 3
   S: value(g)
```

negative 3A...
That's 76.8 minus ( )...
76.8 minus 96 equals negative 19.2.

Negative 3A equals negative 19.2.

So A equals 6.4 again.

It's meters per...
No it isn't.
It's feet per second squared.
This is feet per second squared.
Okay, but that's just the vertical
part.
Cause it's not a slant.
So you can find...
Force equals ma..

And so it would be Fy equals mAy

Fy equals mass is 3 slug times 6.4

equals 19.2 pounds.

Okay, then F would equal Fy over the
sine of 30 degrees.
fif...
Equal 19.2 over the sine of 30
degrees equals 19.2 divided by
30 degrees sine equals 38.4 pounds.

Okay, then.
Acceleration...
F equals ma...
38.4 pounds equals...mass is 3 slugs
times A.

A equals 12.8 feet per second squared.

```
      F: value(g): 32
     S: solve(76.8-3*32=-3*a)
      S: value(76.8-3*32)
       S: value(3*32)
       F: value(3*32): 96
       S: value(76.8-96)
       F: value(76.8-96): -19.2
      F: value(76.8-3*32): -19.2
      S: solve(-19.2=-3*a)
       S: solve(a=-19.2/-3)
        S: value(-19.2/-3)
        F: value(-19.2/-3): 6.4
       F: solve(a=-19.2/-3): a=6.4
      F: solve(-19.2=-3*a): a=6.4
     F: solve(76.8-3*32=-3*a): a=6.4
    F: solve(T-m1g=-m1a): a=6.4
   F: value(proj(accel(m1))): 6.4










   S: solve(F=ma) [a=accel(m1)]    (5)
    S: solve(a=F/m)
     S: value(F/m)
      S: value(F)
       S: solve(F=Fy/sin 30)
        S: value(Fy/sin 30)
         S: value(Fy)
          S: solve(Fy=m*ay)
              [ay=proj(accel(m1))]
           S: value(m*ay)
            S: value(m)
            F: value(m): 3
            S: value(ay)
            F: value(ay): 6.4
            S: value(3*6.4)
            F: value(3*6.4): 19.2
           F: value(m*ay): 19.2
          F: solve(Fy=m*ay): F=19.2
          F: value(Fy): 19.2




         S: value(19.2/sin 30)

         F: value(19.2/sin 30): 38.4
        F: value(Fy/sin 30): 38.4
       F: solve(F=Fy/sin 30): F=38.4
      F: value(F): 38.4




       S: value(m)
       F: value(m): 3
       S: value(38.4/3)
       F: value(38.4/3): 12.8
      F: value(F/m): 12.8
     F: solve(a=F/m): a=12.8
    F: solve(F=ma): a=12.8
   F: value(Fn(proj(accel(m1)))): 12.8
  F: value(Fn(proj(accel(m1))))),
```