THE JOURNAL OF THE LEARNING SCIENCES, 8(1), 71–125

Rule-Learning Events in the Acquisition of a Complex Skill: An Evaluation of Cascade

Kurt VanLehn

Learning Research and Development Center
University of Pittsburgh

Acquiring a complex cognitive skill often involves learning principles of the task domain in the midst of solving problems or studying examples. Cascade is a model of such learning. It includes both rule-based reasoning and several kinds of analogical, case-based reasoning. Task domain principles are represented as rules, and Cascade learns new rules at *rule-learning events*, which are initiated by an impasse and utilize multiple kinds of reasoning. In this article, I evaluate Cascade's model of rule-learning events by analyzing ones gleaned from protocols of physics students solving problems and studying examples. As expected, Cascade's model is overly simple, but it appears feasible to extend it to cover all the observed learning events. The data themselves were surprising in that there are few learning events relative to the number that could have occurred, and those that did occur often involved forms of reasoning that are considerably shallower than expected. The data suggest ways that instruction can be improved to increase both the quantity and depth of learning events.

It always has been puzzling how a knowledge-based system could construct more knowledge while simultaneously using the knowledge it had. It is a bit like M. C. Escher's woodcut of hands drawing themselves: The knowledge is both constructor and constructee at the same time.

Early solutions to this puzzle assumed that learning was done by "hard-wired" knowledge-constructing operators that were domain-independent. Examples of such models include ACT–G (Anderson, 1982), Sierra (VanLehn, 1987), Sage (Langley, 1987), UPL (Ohlsson, 1987), and many others. The problem with this

Correspondence and requests for reprints should be sent to Kurt VanLehn, University of Pittsburgh, Learning Research and Development Center, 3939 O'Hara Street, Room 823, Pittsburgh, PA 15260.

solution to the puzzle was that it is not easy to get domain-independent mechanisms to construct the intricate rules that are needed for sophisticated, knowledge-rich problem solving.

More complicated mechanisms followed. For instance, a common idea was that knowledge was arranged in a base level and a metalevel. The metalevel could observe and edit the knowledge in the base level. Several models kept a complete trace of the system's base-level reasoning, which the metalevel would analyze to debug the system's knowledge in the event that a failure of some kind was detected (e.g., Neches, 1987; Ram & Cox, 1994; Ram, Narayanan, & Cox, 1995; Sussman, 1975). That is, learning was viewed as "self-debugging." Although the learning-as-self-debugging paradigm was (and still is) interesting as artificial intelligence (AI), its complexity often made it seem implausible as a model of human cognitive skill acquisition.

A breakthrough occurred in the mid-1980s with the simultaneous invention by several investigators of explanation-based learning (Russell & Norvig, 1995). Although explanation-based learning initially was developed to model speed-up learning, it gradually was recognized that it could construct completely new knowledge as well, where *new* means that the knowledge is not a deductive consequence of the existing knowledge. The key idea was to include in the knowledge base some *overly general knowledge* in the form of explanation patterns (Schank, 1986), causal attribution heuristics (Anderson, 1990; Lewis, 1988; Pazzani, Myer, & Flowers, 1986), determinations (Bergadano, Giordanna, & Ponsero, 1989; Widmer, 1989), overly general rules (VanLehn, Ball, & Kowalski, 1990; VanLehn, Jones, & Chi, 1992), and so on. However, overly general knowledge was only used when the "official" task domain knowledge failed.

Failure was defined differently in different systems, but a simple and common definition was to signal a failure and invoke learning whenever the reasoner reached an *impasse*. An impasse was defined as a goal that could not be achieved either because the official knowledge lacked any means of achieving it or the official knowledge had multiple means for achieving the goal but no way to decide which way to take.

When an impasse signals a need to modify the existing knowledge, the system tries to use its overly general knowledge to resolve the impasse (i.e., to achieve the goal that cannot be achieved with official knowledge alone). If the overly general knowledge succeeds, the line of reasoning taken is compressed and specialized to form an official piece of knowledge. This piece of knowledge is usually new in that it cannot be deductively derived from the existing official knowledge. Thus, instead of a whole host of domain-independent knowledge-construction operators, metalevels or debugging techniques, the system has just one domain-independent cycle, impasse–repair–reflect, in which the repair involves reasoning with overly general knowledge, and the reflection is an application of explanation-based learning.

The system's power comes from its overly general knowledge, which can be domain-specific. Moreover, new overly general knowledge can be acquired easily by syntactic manipulation of existing knowledge (Ram, 1990). There seems to be no limit to the power of this approach for continually increasing the system's knowledge.

This seems an excellent solution to the puzzle of how a knowledge-based system can learn itself. The impasse–repair–reflect mechanism is simple, but the learner is apparently arbitrarily competent due to the open-endedness of its overly general knowledge. The simplicity makes it appealing as a cognitive model and robust as an engineered system. Indeed, two impasse–repair–reflect cognitive models invented in the late 1980s (ACT and SOAR) have changed very little since then.

Although the computational problems seem to have been solved, there remained a nagging empirical problem: Human learners seldom seem to be aware of this impasse–repair–reflect cycle. In particular, self-observation suggested that we seldom say we are stuck when we supposedly reach impasses, we rarely talk about using overly general rules, and we never talk about compressing lines of reasoning into rules.

One response to this supposed empirical problem was to claim that the impasse–repair–reflect cycle is built into the cognitive architecture in such a way that participants have no conscious access to it, and thus cannot talk about it. For instance, SOAR's chunking is intended to model all forms of learning, even episodic learning, and participants have no conscious access to it (Newell, 1990). ACT's proceduralization also is used to account for a variety of memory phenomena, many of which appear to be beyond conscious access (Anderson, 1993).

A second response to the supposed empirical problem was to check the veracity of the self-observation. It may be that impasse–repair–reflect cycles occur infrequently, and when they do occur, our attention is drawn to other things and we do not notice that the cycle is operating. Several investigations (Lawler, 1991; Schoenfeld, Smith, & Arcavi, 1993; Siegler & Jenkins, 1989; VanLehn, 1991) combed through protocols of learning to find out if participants showed any signs of learning events, where a learning event is loosely defined as an episode in which the participant seems to acquire a new piece of knowledge. Sometimes the evidence for learning occurs at the time of the event itself, when participants say something that indicates that they have applied a piece of knowledge that they did not possess beforehand. In other cases, the evidence for learning is that participants shifted from nonapplication to application of a piece of knowledge, and the learning event occurred during the time of the shift. These microgenetic studies, as they are called in the developmental psychology literature, often found such evidence of learning events. Although sometimes the participants seemed fully cognizant of their learning processes, this was infrequent. Most learning events were characterized by pauses or garbled speech, rather then succinct statements corre-

sponding to each step of some learning process such as the impasse–repair–reflect cycle. Were the participants consciously aware of the learning event? It is difficult to say. More data would be helpful, and gathering such data was one of the purposes of this study.

This study extends the results of its predecessors in several ways. First, the task domain used here is much more complicated than finger counting, the Tower of Hanoi (VanLehn, 1991), and the others used in earlier studies. Second, in all the earlier studies, the participants already knew one method for achieving their task. What they learned was a more efficient method. In this study, the participants learned how to perform a task that they could not do at all prior to their learning events. Last, the large number of protocols and the rigorous modeling used in this study allowed a more thorough assessment of the variety of methods used by participants both to trigger learning events and to construct knowledge during them. In short, one purpose of the study is to probe deeper into learning events to gain a more complete understanding of their similarities and differences. This is a descriptive empirical task, rather than hypothesis testing or other more familiar empirical tasks, so the methods are taxonomic and heuristic.

The second purpose of the study is to evaluate Cascade, a computational model of complex cognitive skill acquisition based on the impasse–repair–reflect cycle. Cascade models students who are learning physics. Cascade begins with knowledge from the expository parts of a college textbook in introductory Newtonian mechanics. It then studies examples and solves problems. (Physics examples, such as the one in Figure 1, are problems that have been solved already.) As Cascade works, not only does it learn how to solve physics problems, it also learns principles of physics via an impasse–repair–reflect cycle. Learning principles is necessary because Cascade is given only partial knowledge of the expository parts of the textbook, representing the fact that students do not always learn everything that the textbook says. Moreover, textbooks often omit important principles from their exposition. Thus, Cascade needs to learn about the domain theory (physics) as well as how to use the theory to solve problems.

This article is a progress report on the empirical evaluation of Cascade. The evaluation began several years ago. VanLehn et al. (1992) showed that Cascade could model all the main effects of an important phenomenon, the self-explanation effect (Chi, Bassok, Lewis, Reimann, & Glaser, 1989). VanLehn and Jones (1993d) analyzed students' errors to show that the self-explanation effect is based on acquiring rules rather than cases or other large units of knowledge. VanLehn and Jones (1993b) showed that Cascade's behavior closely matched protocols of participants doing problem solving and example studying. In particular, 95% of Cascade's inferences were matched by participants' inferences, and 75% of the participants' inferences were matched by Cascade's inferences. VanLehn and Jones (1993a) showed that students who used analogical problem solving less were more effective learners, as predicted by Cascade. VanLehn (in press) analyzed Cascade's account of students' analogical reasoning.

This article evaluates Cascade's model of rule-learning events, which are defined to be episodes wherein Cascade constructs a new principle of physics via an impasse–repair–reflect cycle. Because Cascade already has been fit to approximately 40 hr of verbal protocol data, it was possible to locate episodes in the protocols corresponding to times when Cascade either had a rule-learning event or could have had one. Some of these episodes contained human learning events. Each human learning event was classified according to what triggered the event, what kind of reasoning occurred during it, and whether the physics principle constructed during it was used on subsequent occasions or forgotten. The evaluation simply asked how many of the human learning events could be modeled by Cascade.

We did not expect Cascade to be able to model every learning event. As mentioned earlier, Cascade and its contemporaries were invented as simple alternatives to the learning-as-self-debugging paradigm (e.g., Neches, 1987; Ram & Cox, 1994; Ram et al., 1995; Sussman, 1975). Although that paradigm was considered too complex to be a model of human learning, it was unconstrained enough that it certainly could model any conceivable learning event. The question addressed by this evaluation was essentially whether a simpler model, Cascade, could do almost as well, and when it fell short, would the full power of learning-by-self-debugging be necessary, or would some simpler alternatives work?

In addition to the planned outcomes (a descriptive study of learning events and an evaluation of Cascade), two unanticipated findings emerged. First, there were remarkably few learning events relative to the number of opportunities. Second, the kind of reasoning employed during learning events often was surprisingly shallow. These findings suggested that students are not learning as much as they could. The findings prompted some further protocol analyses aimed at understanding ways to increase the quantity and depth of students' learning events.

To summarize, one purpose of this study was to evaluate Cascade's model of learning events; thus, the first section describes Cascade in some detail. The second purpose of the study was to augment the existing descriptive accounts of learning events with a taxonomic description of learning events that occurred while students learned physics. The article's second section describes the empirical study and how the human learning events were culled from the protocols. The third and fourth sections are the core results of the study. The first one contrasts Cascade's and the participants' reasoning during learning events, and the second one contrasts the conditions under which Cascade and the participants initiate learning events. The last of the results sections examines the "shallowness" findings. A discussion section puts the results in perspective by reviewing earlier work on learning events and proposing a general model.

CASCADE

In this section, I describe Cascade and its main features by referring to the data and to common assumptions among other models of physics cognition. The section also briefly compares Cascade to other models in the literature.

Cascade Is a High-Level Model

Cognitive models of learning vary considerably in their generality and level. At one end of the spectrum are the cognitive architectures such as SOAR (Newell, 1990) and ACT–R (Anderson, 1993), which are low-level models with high generality. These models focus on how information is stored and retrieved as an automatic constituent of thinking. The models are intended to be extremely general. They are intended to apply to the learning of all kinds of knowledge. At the other end of the spectrum are high-level models of specific types of learning, such as Shrager's (1987, 1990; Shrager & Klahr, 1986) model of how participants learn a mental model of a programmable device via discovery. Such models elucidate the particular strategies that participants use to learn particular kinds of knowledge. Sometimes a model of high-level learning is constructed on top of a cognitive architecture, as was the case with a model implemented in SOAR of participants mastering an educational game (Conati & Fain Lehman, 1993).

Cascade is a high-level model of learning that is not built on top of a cognitive architecture. Instead of a cognitive architecture, it uses simple content-addressable memories of the type used in AI programs for decades. It would be interesting to reconstruct Cascade on top of a cognitive architecture. Until then, Cascade cannot account for certain phenomena, such as the fact that after repeatedly solving problems involving blocks sliding down inclined planes, participants simply "know" that exactly three forces act on such blocks. Accounting for such universal "speed up" phenomena is the responsibility of the cognitive architecture.

Cascade's generality has not yet been tested in that all work so far has concerned Newtonian mechanics. Nonetheless, physics is a difficult task domain because of the rich mixture of conceptual, mathematical, and procedural content, so elucidating students' learning of physics is both scientifically and educationally important. Moreover, physics is typical of many other task domains, such as mechanical engineering or statistics, that use mathematically expressed theories to analyze natural phenomena, so it is likely that Cascade would cover other important task domains besides physics.

Both Rules and Cases Are Necessary for Representing Physics Knowledge

Many models of physics cognition represent physics knowledge as rules (e.g., Bundy, Byrd, Luger, Mellish, & Palmer, 1979; Novak & Araya, 1980). Some of the rules correspond to well-known physics principles, such as Newton's first law. Others are rarely mentioned in textbooks, such as "The tension in a string is equal to the magnitude of any tension force exerted by that string."

On the other hand, some models of physics cognition represent knowledge as cases (e.g., Elio & Scharf, 1990; Reimann & Schult, 1993; Reimann, Wichmann, & Schult, 1993). These models define a case to be the solution to a whole physics problem. The initial case base is acquired by studying examples of simple problems being solved. Harder problems are solved by combining the cases of two or more simple problems.

In the protocols used to evaluate Cascade, students often mention both rules (e.g., Newton's law) and cases (e.g., the inclined plane example), so students clearly have both rules and cases in memory. However, it is difficult to tell from these verbal references what operational role the rules and cases play. Although both are mentioned, only one may be helping students solve problems or learn physics.

A closer look at the computational modeling literature suggests that competent physics problem solving requires both rules and either cases or something like cases. Modeling indicates that although it is possible to solve physics problems using only rules and weak methods such as means–ends analysis, such a model will search wildly. To even faintly resemble human problem solving, most rule-based models of physics problem solving have used some kind of physics-specific knowledge to prevent the huge combinatorial searches that result from using just rules and weak methods. In Mecho (Bundy et al., 1979), the search control knowledge took the form of metalevel, physics-specific schemas. In Issac (Novak & Araya, 1980), Newton (de Kleer, 1975), Able (Larkin, 1981), and other systems (Lamberts, 1990; McDermott & Larkin, 1978; Priest & Lindsay, 1992), it took the form of physics-specific rule interpreters. In Axe–T (Reimann et al., 1993; Reimann & Schult, 1993) and Eureka (Elio & Scharf, 1990), it took the form of cases. All these projects discovered that rules alone are not sufficient to solve physics problems efficiently. Because human students do not search as wildly as a rules-only weak-method problem solver, it is highly like that they learn more than just rules.

On the other hand, attempts to use case-based methods for solving physics problems ended up incorporating some rules as well (Elio & Scharf, 1990; Reimann et al., 1993; Reimann & Schult, 1993). A physics case contains the solution to a whole physics problem. Although cases alone are capable of solving problems that are similar to the ones used in training, extending a case to solve less similar problems requires smaller pieces of knowledge, namely, rules. Because human students clearly can solve such problems, they must be using rules as well as cases.

Cascade's Knowledge Representation

Cascade represents physics knowledge as rules and as case-based search control knowledge. The rules are used to represent physics principles. A means–ends anal-

ysis problem solver uses the rules to solve physics problems and to self-explain examples. Cases are used to make decisions about which rule to apply. Cascade's cases are represented as problem–goal–rule triples. Whenever the solver successfully achieves a goal, a triple is stored in a simple content-addressable memory. A problem–goal–rule triple represents that its rule was responsible for the successful achievement of a particular goal in a particular problem. Cascade's episodic memory (its experience) is just the collection of all triples stored in memory.[1] This experience is used by the solver whenever two or more rules apply to its current goal. The solver searches for a triple whose problem matches the current problem and whose goal matches the current goal. If such a triple is found, then the solver applies the rule mentioned by the triple. If not, the solver picks a rule arbitrarily (and remembers that it has done so because it may back up later and pick a different rule). The triples represent essentially the same content as a case-based memory for problem solving, although the information is organized differently. This mechanism is called *analogical search control* (VanLehn & Jones, 1993b). It is similar to mechanisms in Eureka (Jones, 1989), Gips (Jones & VanLehn, 1992, 1994), Icarus (Langley, McKusick, Allen, Iba, & Thompson, 1991) and Daedalus (Langley & Allen, 1993).

Physics concepts such as force or mass are utilized throughout Cascade's rules. For instance, a rule such as "An object that is slowing down is accelerated in a direction opposite its motion" mentions the concept of acceleration (and several other concepts as well). To fully understand a concept is simply to know all the rules that mention it.

Physics students have many misconceptions (e.g., Halloun & Hestenes, 1985; McCloskey, Caramazza, & Green, 1980). For instance, they often believe that whenever an object is moving, there must be a force propelling it along. Such misconceptions are represented in Cascade as rules (e.g., Ploetzner & VanLehn, 1997).

In short, the knowledge representation in Cascade consists of triples, which represent the student's episodic memory or experience, and rules, which represent the student's knowledge of physics principles, concepts, and misconceptions. As discussed subsequently, rules also are used to represent overly general knowledge that is not used during ordinary physics problem solving but that does play an important role during rule-learning events.

How Cascade Solves Problems and Studies Examples

Cascade can do two basic tasks: study an example or solve a problem. Its methods for solving problems are discussed first.

[1]Actually, the triples record only the solver's successful experiences. Rule selections that led to failures are not stored. Cascade models only the contents of episodic memory that it needs, and not the complete contents.

Originally, Cascade was just a means–ends analysis problem solver that used problem–goal–rule triples to help it select a rule when more than one rule applied to the current goal. In fitting Cascade to the protocols, it was discovered that students often would use analogy to examples to achieve a goal even when they could have used rules (VanLehn & Jones, 1993c). For instance, they sometimes would refer to an example even when their current goal was isomorphic to one that they had earlier achieved without referring to an example. To improve Cascade's fit to the protocols, it was revised so that the user could direct it to use analogy on some goals.

There are many different kinds of analogy, and Cascade uses a particularly simple one. It finds an example similar to the current problem, searches for a line in the example that matches the current goal, and copies the relevant parts of the line as a "solution" to the goal (often incorrect). This process is similar to Carbonell's (1983; 1986) transformational analogy, so it bears that name.

Cascade studies an example by reading each line of the example's solution and either self-explaining the line or glossing it. An example, such as the one shown in Figure 1, is represented to Cascade as a problem and solution, in which the solution consists of a list of lines. Each line is an assertion, often in the form of an equation. For each line, Cascade either can self-explain it or gloss it. The choice is determined by the user, who typically is trying to fit Cascade's performance to a protocol.[2] If Cascade is directed to self-explain a line, it treats the line as an assertion to be proved and tries to achieve this goal with the means–ends analysis problem solver. That is, Cascade models self-explanation as rederiving (proving) a solution line. If Cascade is directed to gloss a line, then it merely stores the line's assertion as if it had derived it. This allows the assertion to be referred to later.

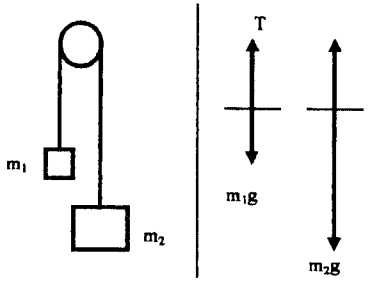
In summary, Cascade ends up having two ways to work on each of its two main tasks: It can study an example line either by self-explaining it or glossing it. It can achieve a problem-solving goal by applying either a rule or transformational analogy.

How Cascade Handles Impasses

Whenever means–ends analysis is running, during either problem solving or example explaining, it can reach an impasse. An impasse occurs when no rules apply to the solver's current goal, or the solver has tried all the rules that apply to the current goal and they all led to failure.

Cascade can respond to impasses in one of four ways:

[2]In VanLehn and Jones (1993b), for example, for each participant, we fit Cascade's behavior to the example-studying part of the participant's protocol. This caused Cascade to learn certain rules but not others. We then let Cascade solve problems. We measured the match between its behavior and the problem-solving part of the participant's protocol.



Problem:

Consider two unequal masses connected by a string that runs over a massless and frictionless pulley, as shown in the left pane above. Let m_2 be greater than m_1. Find the tension in the string and the acceleration of the masses.

Solution:

If the acceleration of m_1 is a, the acceleration of m_2 must be $-a$. The forces action on m_1 and m_2 are shown in the right pane.

The equation of motion for m1 is: $T-m_1g=m_1a$.

The equation for motion of m2 is: $T-m_2g=-m_2a$.

Combining, we obtain

$$a = \frac{m_2 - m_1}{m_1 + m_2} g \qquad T = \frac{2m_1m_2}{m_1 + m_2} g$$

FIGURE 1 A physics example.

- *Backing up:* The problem solver engages in chronological backup. That is, it resets its state to the last time it had to guess, and makes another guess instead. A guess is defined to be choosing a rule to apply to a goal when two or more rules are applicable and analogical search control does not provide any advice.
- *Transformational analogy:* This occurs only during problem solving, but it does not result in learning a new physics rule. It was described earlier.
- *Explanation-based learning of correctness:* This is the main process for learning physics rules. It is described in a subsequent section.
- *Analogical abduction:* This occurs only during example studying. It is another process for learning physics rules, and is also described subsequently.

This list of impasse-handling methods is not intended to be exhaustive. Cascade could, for instance, include Sierra's repair strategies (VanLehn, 1987).

This concludes the brief overview of Cascade. Further details about how it reaches impasses and how it learns rules will be supplied later, after presenting the data with which Cascade will be evaluated.

COLLECTING THE CORPUS OF LEARNING EVENTS

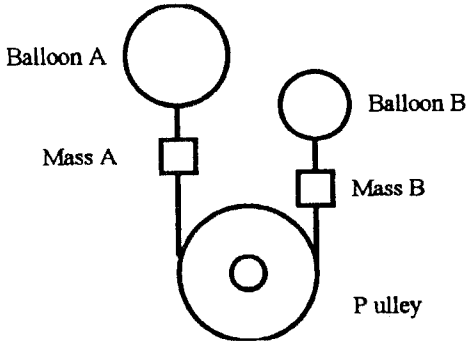
The protocols for these analyses came from the Chi et al. (1989) study of physics learning. The participants, 9 college students, reviewed basic mathematical and kinematics material until they could pass criterion tests on it. They studied chapter 5 from Halliday and Resnick (1981), which introduces Newtonian mechanics and gives a brief introduction to solving physics problems. The participants then talked aloud as they studied three examples from Halliday and Resnick and then solved 19 problems. Of the 19 problems, the first 12 were analogous to examples and the remaining 7 were not. Figure 1 shows one of the examples. Figure 2 shows a problem analogous to the example, and a problem that is not analogous to it. (Several of the problems, such as the ones in Figure 2, had multiple parts. Some parts were posed to Cascade as different problems, so it actually solved more than 19 problems.)

Participants worked for an average of 4.5 hr each, and the resulting collection of protocols is over 3,000 pages long. This makes it difficult to locate the rule-learning events. Therefore, the data analysis procedure had three steps: (a) determining which rules participants could have learned, (b) determining where in the protocols they could learn them, and (c) examining those locations in the protocols to see if a learning event occurred. The following subsections describe these steps.

Which Rules Could Be Learned?

The first step was to determine which rules were most likely to be missing from participants' knowledge as they began to study examples and solve problems. A rule could be missing either because it was not mentioned in the expository text that the students read, or it could be in the text but easily misunderstood or forgotten by the students.

Which rules are missing from the textbook? First, a cognitive task analysis was conducted of the task domain, and a set of rules developed that were computationally sufficient to correctly answer most problems in the domain. The result of this analysis was the first version of Cascade, which had 62 physics rules in its knowledge base.



The two balloons pictured above are pulling on one another via a massless, frictionless pulley. Suppose balloon A is pulling up on mass A with a net force of 500 N and balloon B is pulling up on mass B with a net force of 100 N (mass A = mass B = 10 kg). (a) What is the tension in the massless rope? (b) What is the acceleration of the balloons?

Non-analogous Problem

An elevator weighing 6000 lbs is pulled upward by a cable with an acceleration of 4.0 ft / s^2. (a) What is the tension in the cable? (b) What is the tension when the elevator is accelerating downward at 4.0 ft / s^2 but still moving upward?

FIGURE 2 Two physics problems.

The next step was to find out which of these rules was mentioned in the textbook. Each of Cascade's 62 rules was expressed in English. Two people not associated with the project judged whether each rule was mentioned in the textbook. There was 95% agreement between the two judges ($\kappa = .903$). Disagreements were settled by a third judge. Of the 62 rules, 33 rules were missing.

However, this figure is a bit inflated because Cascade used a fine-grained representation for direction. For instance, to represent that a gravitational force is directed straight down, Cascade had three rules. The rules asserted that the inclination is 90 degrees, the qualitative vertical direction is downwards and the qualitative horizontal direction is null (neither leftwards nor rightwards). For the purposes of this analysis, such geometric detail is superfluous, so the 33 rules were reduced to 10 basic rules, which are included in Table 1 as the first 10 items.

Which rules in the textbook were hard to learn? A second method for determining which rules might need learning was to analyze the participants' errors

(VanLehn & Jones, 1993d). Many errors came from a lack of knowledge of rules that were not mentioned in the textbook, but those rules are already listed in Table 1. Rules 11 and 12 were added to the list because their absence caused errors even though they were mentioned in the textbook.

The rules of Table 1 include all rules that caused errors and all rules that are missing from the text that the participants read. These rules seem most likely to have been missing at the time the students began studying examples and solving problems.

Which Learning Opportunities Are Learning Events?

The next step in the analysis was to locate all places in the protocols where the rules of Table 1 could be used. This was easily accomplished by examining traces of Cascade's reasoning. Each of these locations is called a *learning opportunity*.

It was next determined, for each learning opportunity, whether the participant used the rule, did not use the rule, learned the rule, and so forth. The results are displayed in Figure 3 (there is one chart for each of the rules in Table 1). Charts have a column for each time that the rule could be used during example studying or problem solving. Within a chart, the columns are ordered chronologically, with the leftmost column representing the first possible use of the chart's rule. There is a

TABLE 1
Rules Most Likely to be Missing

1. The component of a vector along an axis is negative if the angle between the vector and the axis is measured with respect to the negative axis.
2. A normal force is exerted by a surface on objects supported by it.
3. The coordinate axes can be rotated in order to simplify projection of vectors onto them.
4. Any object can be a body (i.e., the forces and accelerations on it will be analyzed with Newton's law). In particular, the knot made by tying several massless, inelastic strings together can be a body.
5. A frictional force is parallel to the object's motion but opposes it.
6. A compressed spring exerts a compression force on the objects touching its ends.
7. A compressed fluid exerts a pressure force on the surfaces of its container.
8. The acceleration of an object moving along an inclined plane is parallel to the surface of the plane.
9. In a pulley system, with objects tied to the ends of a string that runs over a pulley, if one object is accelerating towards the pulley, then the other is accelerating away from it.
10. When two objects are connected by a taut, inelastic string, the magnitudes of their accelerations are equal.
11. The tension in a string equals the magnitude of the tension force due to the string that is exerted on an object attached to the string.
12. Weight is the force exerted by gravity on an object of non-zero mass in a non-zero gravitational field.

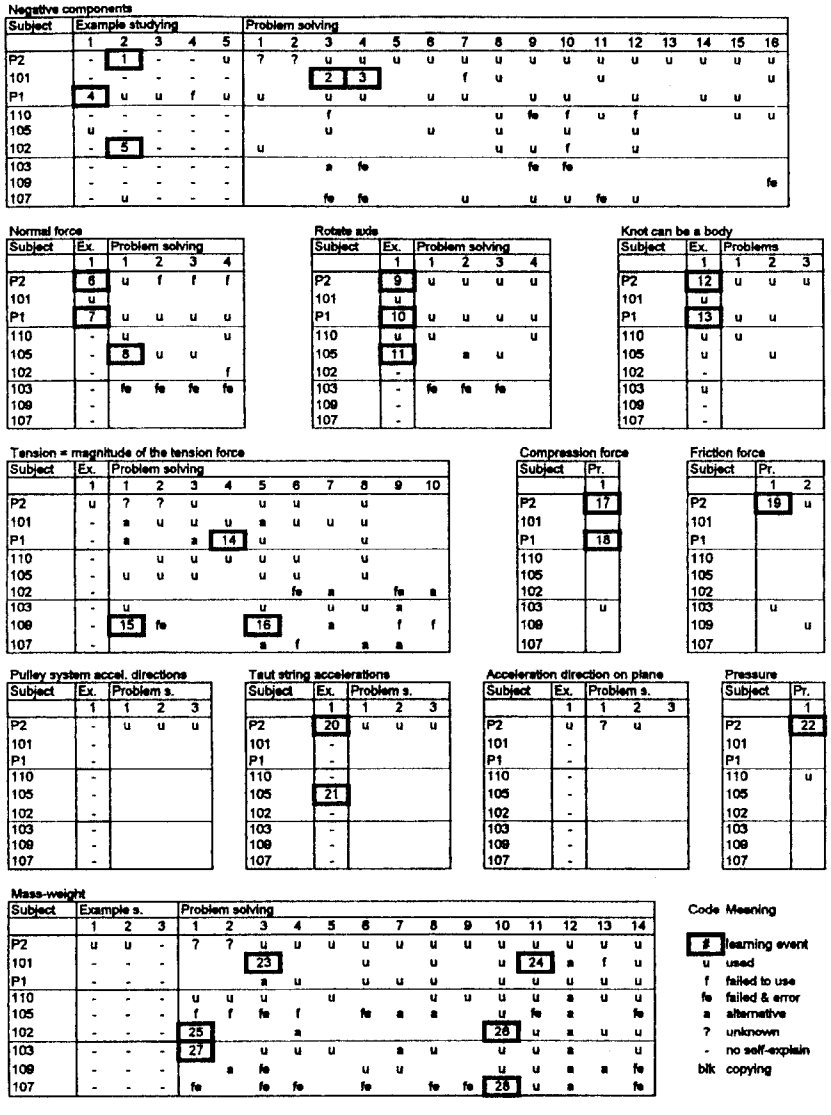


FIGURE 3 Learning opportunities.

row for each participant. Each cell contains a code that describes what happened at that learning opportunity. The following are the codes used in the cells:

- A dark box with a number inside it indicates a learning event. Events are numbered to facilitate referring to them. The same numbering is used in the Appendix, which describes each event in detail.
- u: The participant used the rule.
- fe: The participant failed to use the rule and this caused an error.
- f: The participant failed to use the rule, but this did not cause an error, usually because some other error compensated for the first error, thus producing a correct answer to the problem.
- A hyphen indicates that the student did not self-explain during the time when the rule could be used, and thus avoided the opportunity to apply the rule.
- A blank cell indicates that the student used transformational analogy at the time the rule could be used and thus avoided the opportunity to apply the rule.
- a: The participant used some alternative method, other than transformational analogy, that caused him or her to avoid the place where the rule could be applied.
- ?: The protocol was missing, so it is not known what happened at this learning opportunity.

Because Cascade's rule-learning events are an interruption of means–ends analysis problem solving that focuses on remedying some just-detected flaw in the domain knowledge, a learning event was coded in Figure 3 if a participant used a rule but paused a long time before applying it, mentioned being stuck or confused, referred to the textbook, or showed any other sign of processing that went beyond that participant's usual example studying or problem solving behavior. Similar criteria were used for coding learning events in earlier studies (Siegler & Jenkins, 1989; VanLehn, 1991).

RULE CONSTRUCTION DURING LEARNING EVENTS

In this section and the next, I analyze the observed learning events to evaluate Cascade's model of rule-learning events. Each section begins with a description of Cascade's model, then categorizes the observed learning events, indicting which fit the model and which did not. In this section, I discuss the reasoning that occurred during learning events. In the next section, I discuss the reasoning that initiated the learning events.

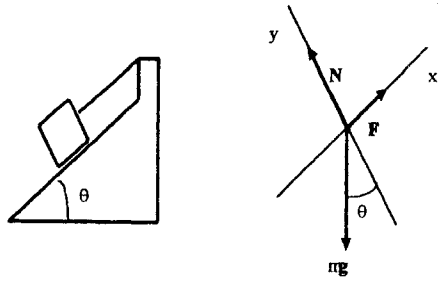
How Cascade Constructed Rules

Cascade has two methods for constructing rules: explanation-based learning of correctness and analogical abduction. These methods are discussed in the following two sections. Explanation-based learning of correctness is discussed first because it was responsible for constructing most of the rules that Cascade learned.

Explanation-based learning of correctness. Explanation-based learning of correctness (VanLehn et al., 1992) presupposes that students distinguish "official" physics knowledge from overly general and commonsense knowledge of physical phenomena, and that when they solve textbook physics problems, they use only their official physics knowledge. There is some evidence for this assumption. For instance, students often harbor misconceptions that only appear when they are asked to reason using common sense; when asked to reason formally, they use different rules and get different answers (e.g., di Sessa, 1993; Halloun & Hestenes, 1985). Part of what students need to learn is which commonsense rules are also official rules of physics. For instance, when you speed up while traveling in a straight line, both common sense and official physics rules say that you accelerate; however, when you travel in a circle at a constant speed, common sense says that you do not accelerate, but official physics says that you do.

Cascade's commonsense physics knowledge includes both descriptions of physical phenomena, such as the fact that a block supported by a surface pushes down on the surface, and overly general rules that link commonsense reasoning with official physical reasoning. The overly general rules do not always produce correct conclusions. For instance, the overly general rule "A push corresponds to an official physics force" would incorrectly conclude that the push one feels when rounding a corner in a car (often called centrifugal force) corresponds to an official physical force. When Cascade is reasoning normally, it does not use such rules because they are not marked as official physics rules. Cascade's official physics rules include official mathematical rules, and its commonsense rules include mathematical overgeneralizations, such as "Any operation applied to a negative quantity produces a negative quantity."

Although commonsense rules are not used during ordinary problem solving, they are used during explanation-based learning of correctness. Cascade's learning events occur in response to an impasse, and an impasse occurs when the current goal cannot be achieved with the official physics rules known to Cascade. Explanation-based learning of correctness begins by turning off the flag that restricts the solver to using official physics rules. Cascade then tries to achieve the current goal using all the rules it knows. If it succeeds, then it uses explanation-based learning (Russell & Norvig, 1995) to construct a new official physics rule. Explanation-based learning compresses the reasoning Cascade made between the time of the impasse and the achievement of the goal by eliminating all



We wish to analyze the motion of a block on a smooth inclined plane.

(a) Static case. The figure on the left shows a block of mass m kept at rest on a smooth plane, inclined at an angle θ with the horizontal, by means of a string attached to the vertical wall. The forces acting on the block, which we choose as "the body," are shown in the figure on the right. **F** is the force exerted on the block by the string; **mg** is the force exerted on the block by the earth, that is, its weight; and **N** is the force exerted on the block by the inclined surface. **N**, called the normal force, is perpendicular (that is, normal) to the surface of contact because there is no frictional force between the surfaces. If there were a frictional force, **N** would have a component parallel to the incline. Because we wish to analyze the motion of the block, we choose ALL the forces action ON the block. Note that the block will exert forces on the 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.

Suppose that θ and m are given. How do we find **F** and **N**? since the block is unaccelerated, we obtain

$$\mathbf{F} + \mathbf{N} + m\mathbf{g} = 0.$$

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. With this choice of coordinates, only one force, **mg**, must be resolved into components in solving the problem. The two scalar equations obtained by resolving **mg** along the x- and y-axes are

$$F = mg \sin(\theta) = 0 \text{ and } N = mg \cos(\theta) = 0$$

from which F and N can be obtained if θ and m are given. Note that these equations reduce to the expected results for the special case of θ=0 and θ=90.

FIGURE 4 The inclined plane example (parts b and c omitted).

the intermediate results and turning inessential constants into variables. The new rule's conclusion is just the goal itself, with inessential constants turned into variables. The new rule's conditions are ones that match the problem state at the time of the impasse and are minimally necessary to support the rule's conclusion.

For instance, suppose Cascade is self-explaining the example of Figure 4 and it cannot explain the normal force exerted on a block by the inclined plane that supports it. An impasse occurs. Cascade turns off the official physics flag and reasons as follows:

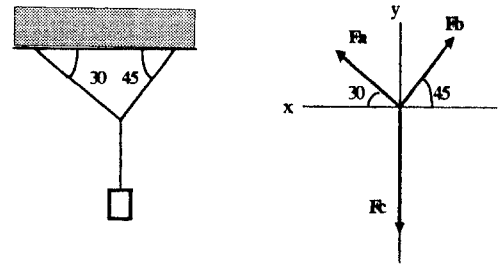
Since the block is supported by the plane, it follows (from a commonsense rule) that the block pushes down on the plane. Therefore (by an overly general rule), the push is an official physics force that acts on the plane and is due to the block. Therefore (by Newton's third law), there is a reaction force to it that acts on the block and is due to the plane.

This reaction force matches the observed normal force and thus explains it. Because explaining the normal force was the current goal, Cascade uses explanation-based learning to compress that line of reasoning into a new rule. The rule's condition is that the block be supported by the plane. The rule's conclusion is just the force that was explained. The identity of the block and the plane did not play a role in the derivation, so the constants that represent them are turned into variables. Thus, the new rule is, "If ?X is supported by ?Y, then there is a force on ?X due to ?Y." The "?" indicates a variable. Cascade assumes that the rule is correct (it is, actually) and marks it as an official physics rule.

Explanation-based learning of correctness bears that name because it uses explanation-based learning, which is usually used to account for speed-up learning, to construct new correct rules out of knowledge that is overly general and therefore incorrect. The trick is that although the overly general knowledge is incorrect, specializations of it can sometimes be correct. Similar accounts appear in other learning theories with the role of overly general rules being played by causal attribution heuristics (Anderson, 1990; Lewis, 1988; Pazzani, 1990), explanation patterns (Schank, 1986), determinations (Bergadano et al., 1989; Widmar, 1989) and constraints (Ohlsson, 1993, 1996).

Analogical abduction. Cascade's second method of constructing rules during learning events is based on analogy. The method only applies when the goal that Cascade is trying to achieve is a fully specified proposition, that is, it contains no variables. Fully specified, variable-free goals occur frequently during example studying and rarely during problem solving, so analogical abduction is used only during example studying. Because the goal is fully specified, Cascade simply assumes that the proposition is true. This resolves the impasse. However, Cascade also builds a rule that sanctions doing this again whenever a subsequent problem is similar to this one. Suppose that the problem in which the learning event occurs is named A, and the proposition that was assumed true is P, then the rule says, "If the current problem is analogous to problem A, and in particular, there is a mapping M, such that M(A) is the current problem, then M(P) is true." This rule construction process is named *analogical abduction* (VanLehn et al., 1992).

As an illustration of analogical abduction, consider the example of Figure 5. An early line in the example says, "Let the body be the knot." This is essentially a strategic move, in that it indicates that the example will be applying Newton's law to the knot formed by the junction of the three strings. However, Cascade cannot find



Problem: The figure on the left shows an object of weight W hung by strings. Consider the knot at the junction of the strings to be "the body." The body remains at rest under the action of the 3 forces shown in the figure on the right. Suppose we are given the magnitude of one of the forces. How can we find the magnitude of the other forces?

Solution: $\mathbf{F}_a$, $\mathbf{F}_b$ and $\mathbf{F}_c$ are all the forces action on the body. Since the body is unaccelerated, $\mathbf{F}_a+\mathbf{F}_b+\mathbf{F}_c=0$. Choosing the x- and y-axes as shown, we can write this vector equation as three scalar equations:

$$F_{ax} + F_{bx} = 0 \text{ and } F_{ay} + F_{by} + F_{cy} = 0$$

using Eq. 5-2. The third scalar equation for the z-axis is simply

$$F_{az} = F_{bz} = F_{cz} = 0$$

that is, the vectors all lie in the x-y plane, so that they have no z-components.

From the figure, we can see that

$F_{ax} = -F_a \cos(30) = -0.866\ F_a$ and $F_{ay} = F_a \sin(30) = 0.500\ F_a$, and

$F_{bx} = F_b \cos(45) = 0.707\ F_b$ and $F_{by} = F_b \sin(45) = 0.707\ F_b$.

Also,

$$F_{cy} = -F_c = -W,$$

because the string C merely serves to transmit the force on one end to the junction at its other end. Substituting these results into our original equations, we obtain

$$-0.866\ F_a + 0.707\ F_b = 0 \text{ and } 0.500\ F_a + 0.707\ F_b - W = 0.$$

If we are given the magnitude of any one of these three forces, we can solve these equations for the other two. For example, if W=100 N, we obtain F_a = 73.3 N and F_b = 89.6 N.

FIGURE 5 The Three Strings Example.

an explanation for this line, not even with its commonsense rules. Thus, it assumes the proposition is correct and builds a rule that says, "If there is mapping M, such that applying the mapping to the Three Strings Example generates a problem equal to the current problem, then let the body be the object in the current problem that corresponds to the knot of the Three Strings Example." In other words, if this problem is similar to the Three Strings Problem, assume its knot should also be the body analyzed by Newton's law.

TABLE 2
Types of Reasoning During Rule Learning Events

1. During example studying
   A. Just accepting (analogical abduction?) [7, 10, 11, 12, 13]
   B. Explanation-based learning of correctness [1, 4, 5, 6, 9, 20, 21]
2. During problem solving
   A. From the instructional material
      1. Reading the textbook [23, 25, 26, 27, 28]
      2. Analogy to an example [2, 3, 8, 15, 16, 22]
   B. Reasoning
      1. Counterfactual [17, 18, 19]
      2. Explanation-based learning of correctness [14]
   3. Algebraic manipulation [24]

Cascade has only these two methods of rule construction, and it applies analogical abduction only if explanation-based learning of correctness fails. As the next section shows, the participants used other methods as well, and they often used something like analogical abduction in preference to other methods.

How the participants constructed rules. In this section, I describe the kinds of reasoning that occurred during the learning events gleaned from the protocols. One coder categorized each of the 28 learning events according to the type of reasoning that participants seemed to be doing during the learning event. The coder, who was familiar with Cascade, invented categories as he went. A second coder was given those categories and coded each of the events independently. The codes agreed in all but one case, for an intercoder reliability of $\kappa = .956$. Table 2 shows the results. The numbers in square brackets identify rule-learning events. In some cases (e.g., rule event 16), the participant seemed to use one line of reasoning to construct the rule and a second to confirm it. For simplicity, the categorization is based on the first line of reasoning in such cases.

The top level of the hierarchy divides rule-learning events by when they occurred, during example studying or during problem solving. During example studying, a common type of reasoning was to just accept what the example asserted without trying to explain it any further. Rule-learning event 12 is a good illustration (see the Appendix). It may be that analogical abduction occurs during these learning events because the participants often explicitly referred back to the example (i.e., they flip the pages of the textbook and reread part of the example) at just the times when the rule formed by analogical abduction would predict that they should do so. Of course, some of the participants do not refer explicitly to the example, but they could be referring to a mentally held version of it instead. Alternatively, as argued in VanLehn and Jones (1993b), the participant may have learned nothing at all during these acceptance events, and just used transformational analogy during problem solving. There is no way to determine using these data whether analogical abduction is in fact the process that occurs when participants just accept an example's line after an impasse.

The next category is explanation-based learning of correctness. Learning events 4 and 6 are good illustrations (see the Appendix). However, it is not easy to tell if a rule-learning event should be classified as an instance of explanation-based learning of correctness because participants rarely mention the rules that they use while reasoning. They prefer to mention only the intermediate results as they reason. Because they do not mention their rules, one cannot easily tell whether the rule that produced a certain intermediate result is overly general or appropriately specific. Consider, for instance, rule-learning event 21. While studying the pulley example (Figure 1), the participant had a slight problem understanding why the two blocks have the same acceleration:

> [Reads: If the acceleration of m1 is a, the acceleration of m2 must be –a.] (Slight pause) O—, Okay, that makes sense because they're connected to a pulley. And if one's moving one speed, then the other one has to be moving at the same speed.

The participant seemed to deduce the equality of the two accelerations from a series of correct inferences: The blocks are connected, so displacement of one causes an equal displacement of the other, thus their speeds are the same, and hence the magnitudes of their accelerations are the same. There exist correct rules of physics that could produce each step of this line of reasoning, but they have not been presented in the textbook, and they would require calculus to derive from the basic definitions of velocity and acceleration. Thus, it is unlikely that the participant was employing correct rules. Instead, the participant was probably reasoning sloppily: The two blocks are connected, so their motions are the same, and thus their speeds are the same; acceleration can be derived from speed somehow, so the accelerations are probably the same as well. This line of reasoning is the application of several commonsense rules and the overly general rule, "If the ?X properties of two objects are equal, and the ?Y property can be derived from the ?X property somehow, then the ?Y properties of the two objects are probably equal as well." When the protocol did not contain enough information to disambiguate the application of correct rules from the application of commonsense rules, and the correct rules were not mentioned in the text and were not likely to be part of a participant's prior knowledge, it was assumed that the participant used commonsense rules.

Learning events that occurred during problem solving were distinguished depending on whether the participant referred to the instructional material (category II.A) or not (category II.B). If they referred to the textbook (category II.A.1), they invariably sought the mass–weight rule. This is not surprising, as that rule has a section to itself in the textbook and the participants probably recalled that the section exists, so when they finally realized that they need to know the exact relation

between mass and weight, they knew where to look. When participants referred to an example to obtain a new rule (category II.A.2), they often did not self-explain the line of the example that they imported. In half the cases (events 2, 3, and 16), the participants tried to understand what they were copying before they copied it. In the other cases, the participant simply accepted the example's assertion without explaining it more deeply, adapted the assertion to their current problem, and formed a new rule. Learning events 2 and 8 illustrate each kind of analogical reasoning.

Category II.B, labeled "Reasoning," contains instances of participants inferring new rules without referring to the textbook or the examples. During learning events in the first category, counterfactual reasoning, participants figured out that a force was missing from their diagrams because the existing forces clearly did not sum to zero and yet the object was at rest. They then used counterfactual reasoning[3] and Newton's laws to infer the direction and possible source of the missing force. Learning event 18 is a good illustration of counterfactual reasoning. Category II.B.2 contains an instance of explanation-based learning of correctness. It is similar to those that occurred during example studying. Category II.B.3 contains a remarkable case wherein a participant used algebraic equation solving to convince himself of a qualitative rule, that weight is the force due to gravity.

Discussion

Let us consider example studying and problem solving separately. Cascade did quite well at modeling the rule-construction methods participants use during example studying. Cascade's two rule-construction methods were consistent with all 12 of the example studying learning events, although as usual with protocol data, participants seldom revealed their reasoning in enough detail that one can be completely sure of the degree of match to the model.

Strictly speaking, Cascade is consistent with only one of the learning events that occurred during problem solving, namely the one that was categorized as explanation-based learning of correctness. However, Cascade could be extended to handle all 5 of the events under the reasoning (II.B) category. For instance, if Cascade could reason counterfactually to resolve an impasse, and explanation-based learning could be extended to deal with counterfactual lines of reasoning, then Cascade could probably model the learning events in category II.B.1. However, Cascade's main weakness as a model is its lack of coverage of the learning events wherein participants referred to the textbook or the examples. Cascade's transformational analogy could probably be extended to cover these. Because transformational analogy can already resolve the impasse, extension is required

[3]To reason counterfactually, one supposes that one of the facts in a given situation is false and sees what that entails.

only for constructing a rule based on the analogy. There are many different ways to generalize an analogy and form a rule. It is not clear how participants do it, or whether they do it for all impasses or just some impasses.

Cascade prefers explanation-based learning of correctness over analogical abduction, but this preference was not strong in the students' learning. Among the problems and examples studied by the participants, Cascade uses analogical abduction only during the learning events involving the knot-is-a-body rule (events 12 and 13), whereas the participants may have used it on three others as well.

In fact, the participants were surprisingly reluctant to use reasoning of any kind. As just discussed, during example studying they preferred to trust the example without verifying its assertions in 3 of the 10 cases in which they had a choice. During problem solving, participants preferred in 11 of 16 cases to seek a new rule in the instructional material rather than construct a new rule via reasoning. This unexpected preference for shallow reasoning will be discussed at greater length later.

THE INITIATION OF LEARNING EVENTS

In this section, I discuss how Cascade and the students initiated learning events. This is an important issue because it is difficult to learn a correct rule if the learning event does not occur in an appropriate context.

How Cascade Initiated Learning Events

Like all impasse-driven learning systems, Cascade's learning events are always initiated by impasses. However, impasses are not in one-to-one correspondence with flaws in the knowledge base. Some missing rules do not cause impasses, and some impasses are not caused by missing rules. Cascade has several methods for dealing with these problems. However, because none of them appears to have been used by the participants, it is convenient to delay discussion of them until after the student data have been presented.

How the Participants Initiated Learning Events

To determine how participants' learning events were initiated, one coder categorized the 28 events, inventing categories as he went. A second coder was given the categories, and coded the events independently. The coders agree in all but three cases, for an intercoder reliability of $\kappa = .853$. Table 3 shows the results. The lowest-level classifications are followed by a bracketed list of numbers identifying the rule-learning events in that category.

Of the 12 learning events that occurred during example studying, all were triggered by impasses that occurred when the participants were trying to explain an

TABLE 3
Ways That Rule Learning Events Were Triggered

1. During example studying
   A. Impasses while trying to explain an example line [1, 4, 5, 6, 7, 9, 10, 11, 12, 13, 20, 21]
2. During problem solving
   A. Impasses
      1. Impasse because no rule exists for the current goal [15, 23, 24, 25, 26, 27]
      2. Impasse during self-explanation of the problem statement [14, 17, 18, 19]
      3. Impasse while self-explaning before doing analogy [2, 3, 16]
      4. Impasse because an incorrect rule is blocked [28]
   B. Checking
      1. Checking one's work against the example [8]
      2. Counterintuitive answer [22]

example line, and are thus consistent with Cascade. Learning event 4 is a particularly clear case.

Of the 16 learning events that occurred during problem solving, only 6 were triggered by impasses that Cascade could model, namely, those in category II.A.1. In these learning events, students were trying to achieve a goal and had no rule that would do so. Learning event 23 is a clear case.

The other impasses in category II.A could in principle be modeled by Cascade, but only if its model of problem solving were extended. To model category II.A.2, Cascade should self-explain the statements of problems rather than just accept them as it does now. Similarly, Cascade could model the learning events in category II.A.3 if it were extended so that it sometimes tried to self-explain an example line before using transformational analogy to copy it over to the current problem. Category II.A.4 contains a learning event in which the participant could have applied an incorrect rule (a misconception) but did not do so apparently because that would mean answering a four-part question with the same answer for each part, which the participant apparently considered unreasonable. Instead of applying the incorrect rule, the participant reached an impasse and learned a correct rule. Cascade would probably have to be significantly extended to handle this kind of reasoning.

In both learning events of category II.B, the participants checked their work. During event 8, the participant checked his force diagram against the example's force diagram. During event 22, the participant checked her answer to the problem using commonsense physics. Both kinds of checking are beyond Cascade's present abilities.

Discussion

Once again, Cascade seems to be able to model the learning events that occur during example studying well, but its coverage of those that occur during problem solving is spotty. In particular, the participants' use of checking to trigger learning events indicates that, contrary to Cascade's assumption, not all learning events are triggered by impasses. Indeed, one of these learning events (22) seems to be part of a debugging process. Debugging a computer program consists of detecting the existence of a bug by noticing an unexpected event (e.g., an incorrect answer or an illegal operation), locating the bug that is causing the unexpected event and fixing the bug. In the case of learning event 22, the participant detected the existence of a bug by noting that her answer did not make sense, located the bug by comparing her solution to the example's solution, and fixed the bug by postulating a force analogous to the one used in the example. In retrospect, it seems that impasse-driven learning events are just a special case of debugging in which detecting and locating the bug is particularly simple. The impasse is the unexpected event that signals the existence of a bug, so the learner does not have to use more sophisticated strategies such as units checking or commonsense reasoning to detect errors. The learner assumes that the reason that the impasse occurred on a particular goal is that a rule is missing that should have applied to that goal at the time of the impasse. This simple assumption replaces the elaborate searches that are sometimes needed to find the bug that is causing an unexpected event. In order for Cascade to model all the learning events in the data, it would have to be changed significantly to adopt the learning-as-debugging paradigm. In particular, it would need more elaborate strategies for detecting errors and other unexpected events (such as units checking) and for locating the bugs. These routines would supplement impasses as triggers for learning events.

As mentioned previously, Cascade has some special methods for dealing with the fact that not all impasses are caused by missing rules and not all missing rules cause impasses. If the impasse-driven learning paradigm is generalized to become learning-as-debugging, some of these problems may be overcome. The remainder of this section discusses the problems, Cascade's solutions, and the participants' solutions.

In any problem solver that searches, some impasses should not trigger a learning event because they are caused by unlucky search control decisions and not by missing rules. Cascade, for instance, sometimes must guess which rule to apply to a goal because multiple rules are applicable and analogical search control does not tell it which one to choose. If Cascade makes an unlucky choice, it will go down a false solution path and perhaps reach an impasse. Such an impasse should be repaired by backing up and guessing again and not by learning a new domain rule. A problem solver that searches should initiate a learning event only after it has determined that the impasse is not caused by an unlucky search control decision.

Cascade solves this problem by always backing up from impasses when it first encounters them. This means that Cascade makes two attempts to solve a problem or explain an example. On the first attempt, it backs up from every im-

passe. If that attempt fails to solve the problem (or explain the example), then there is no combination of search control decisions that will succeed, so there must be some missing or incorrect rules. Thus, Cascade makes a second attempt, allowing learning to occur at impasses (for a complete description of the control structure, see VanLehn et al., 1992, p. 17). In general, all impasse-driven learning systems have the same difficulty when they solve problems that require a search. The system cannot distinguish between impasses caused by searching from those caused by flawed knowledge. Cascade's method, running the search to exhaustion before concluding that some knowledge is missing, should work for all such systems.

Cascade's method was hardly visible in the protocols. The closest approximation occurred in learning event 4. Before trying to construct a rule, the participant first checked for mathematical errors in the line of reasoning preceding the impasse. This is similar to checking it for poor search control decisions. On the whole, however, the learning events did not seem to be preceded by checking for unlucky search control decisions.

Instead of checking their solution paths at the time of an impasse, it could be that students maintain a running estimate of their confidence that they are on a correct solution path, and only learn when their confidence is high. Their confidence would decrease if they guessed (i.e., multiple rules applied, and they chose one arbitrarily) or if they used "fresh" rules, namely rules that they constructed themselves but had not yet used successfully many times. Some data support this suggestion. Because low confidence prevents learning that would occur at classic "no rule applies" impasses, the confidence heuristic explains why this study found so few learning events triggered by no-rule-applies impasses during problem solving. Second, the heuristic predicts that most such learning events should occur early in the problem because fewer rule applications and search control decisions will have been made then. In fact, of the six no-rule-applies impasses (category II.A.1), four occurred early in the problem solving (numbers 15, 24, 26, and 27). In short, although Cascade's method of thoroughly checking for unlucky search control decisions before engaging in learning worked well for it, the participants appear to use some other method, such as the confidence heuristic mentioned previously, to decide whether an impasse is worth learning from.

Another general problem with impasse-driven learning systems is that the missing knowledge might cause an impasse at the wrong place. That is, the missing rule should have been applied to a certain goal, but instead another rule applied and changed the problem solver's state in such a way that an impasse occurs later. If impasse-driven learning handles this late impasse, it will probably construct an incorrect rule. Cascade was able to prevent most cases of this by adjusting the knowledge representation used for rules (see VanLehn & Jones, 1993b, p. 307). However, it is clear that special techniques for handling late impasses are unnecessary if the impasse-driven learning paradigm is generalized to learning-as-debugging. Late impasses, incorrect answers and some errors all appear well after the to-be-learned rule should have applied. Figuring out what kind of bug in the knowledge would cause the observed anomaly is exactly like locating a bug in a program once one has observed that the program is malfunctioning.

Another general problem for impasse-driven learning systems is that some flaws in the knowledge base may not cause any impasse at all. For instance, this occurred whenever Cascade tried to achieve the rather common goal, "Find all the forces acting on the body." Because this goal has a universal quantifier ("Find ALL ... "), it is successfully achieved no matter how many forces Cascade finds. Thus, if Cascade is missing a force law (e.g., the tension force rule mentioned previously), then a force will be missing from the set of forces generated in service of this goal. However, that may not cause an impasse. Indeed, the whole problem may be solved without impasse. The answer will be wrong, but Cascade does not know that, so it has no evidence at all that there is a flaw in its knowledge. The problem with universally quantified goals was solved in Cascade via what in retrospect appears to be a hack (see VanLehn & Jones, 1993b, p. 304). The students, as it turns out, had a much better solution.

By examining the learning events in which students learned forces during problem solving, we can infer the students' method for detecting missing rules that do not cause impasses. As Figure 3 shows, there were five learning events wherein a force was learned during problem solving. In three of these (17, 18, and 19), the problem asked the student to calculate the magnitude of that force. The students self-explained the presupposition that the force existed, and this caused the three learning events. Thus, because these learning events did not occur while the participants were working under a universally quantified find-all-forces goal, they do not help us determine how the students find out about missing rules that do not cause impasses. Thus, the burden of the analysis falls on just two learning events, 8 and 22. In both cases, the participants were oblivious to the missing forces. They only discovered that a force was missing when they checked their work against the example (event 8) or common sense (event 22). Checking both final and intermediate results of a program is just what a good programmer does to detect bugs.

In short, it appears that students are not just impasse-driven learners like Cascade but instead can perform all phases of self-debugging. They can check their work for errors, localize the bugs that cause those errors, and then fix the bugs. It is not yet clear how they decide which impasses (or errors, for that matter) to treat as evidence of a flaw in their knowledge, and which are due to unlucky search control decision, slips (unintentional errors), or other causes that do not require changes in the domain knowledge to correct or prevent them. The confidence-based heuristic mentioned previously is but one possibility.

OTHER LESSONS LEARNED FROM THE LEARNING EVENTS ANALYSIS

Perhaps the most impressive thing about Figure 3 is how little students learn. Not only are there few learning events, but even after a learning event, the supposedly learned rules are sometimes never used. This section reanalyzes the data to find out why the students learned so little and what can be done to increase their learning.

Lost Opportunities for Learning Events

Common sense suggests that opportunities to learn a particular kind of knowledge are lost by failing to use that knowledge during example studying and problem solving. For instance, if law students are supposed to learn legal case-based reasoning, but they are actually constructing arguments using legal rules (laws), then they will learn neither the techniques of case-based reasoning nor the target legal cases. In physics, the target knowledge is a set of rule-like principles, such as Newton's laws, the kinematics equations, and the force laws. If students solve problems and study examples without using these rules, perhaps by using case-based reasoning instead, then they lose opportunities to learn the rules. This fairly obvious hypothesis will be confirmed by an analysis of the rule-learning events. The analysis also quantifies the magnitude of the effects.

During example studying, students lost the opportunity to learn by glossing example lines instead of self-explaining them. That is, they simply read the line and perhaps paraphrased it. Figure 3 can be used to calculate how much learning was lost to glossing. Of the 12 target rules, 9 occurred during example studying, so there were $9 \times 9 = 81$ rule instances to be learned, one for each of the 9 students. Of these 81 rule instances, learning events occurred for only 11. Of the remaining 70 rule instances, there were 11 in which the student seemed to know the rule already, and 59 where the student glossed all the learning opportunities during the example. Now suppose that during those 59 episodes, the students had self-explained the example instead. How many learning events would occur? That depends on how many of the 59 rule instances were known prior to studying the example. Suppose we assume that half of them are known already and half need to be learned. This is a reasonable assumption, given that 11 rule instances were observed to be known already and 11 were observed to be learned. Thus, of the 59 rule instances, approximately 29 would yield learning events if the students self-explained instead of glossing. That is, the number of learning events would quadruple, going from 11 to 40, if students always self-explained examples. Although the literature has indicated that self-explanation is good (Bielaczyc, Pirolli, & Brown, 1995; Chi, de Leeuw, Chiu, & LaVancher, 1994; Chi et al., 1989; Ferguson-Hessler & Jong, 1990; Lovett, 1992; Pirolli & Bielaczyc, 1989; Renkl, in press), this analysis indicates the rather extraordinary potential of this studying strategy.

During problem solving, students often copied portions of an example rather than reasoned from the rules (principles) of the domain. For instance, if the diagram of the problem looked similar to that of an example, students would often just copy the force diagram of the example. Sometimes they would copy the force diagram from memory, a behavior that could be considered case-based reasoning. Such copying corresponds to Cascade's transformational analogy. Because students do not use rules while copying, they lose opportunities to learn the target rules. To calculate the number of lost opportunities, Figure 3 can be analyzed again. When problem solving began, there were 11 rule instances that appear to have been known prior to example studying and 11 that had been learned during example studying. Thus, of the $9 \times 12 = 108$ rule instances, $108 - 22 = 86$ rule instances remained to be learned during problem solving. Of these 86 rule instances, only 14 were learned at learning events, 13 appear to have been known prior to problem solving (but were not accessed during example studying and were thus not included in the 22), and 52 were not learned because the student was copying the example at every occasion when the rule could be used. The remaining 7 rule instances were not learned even though the students occasionally tried in vain to use the rule. If students had never copied and the same 14:13:7 ratio applied, then the number of rule instances learned would rise from 14 to 35. That is, the amount of learning would more than double if students always used rule-based reasoning instead of copying from the examples.

Similar calculations show that if students both self-explained the examples and used rule-based reasoning exclusively to solve the problems, the probability of having a learning event for a target rule if it is not known already would rise from .30 to .94. This quantifies the importance of using good studying strategies.

Some kind of feedback during problem solving would be necessary to raise the probability of learning events all the way to 1.0. In this study, the experimenter did not tell students when they made an error. Consequently, on 7 of the 108 learning opportunities, the students just make errors (e.g., omitting a force) instead of learning even though they were not copying the example at the time. Such cases would probably have been eliminated if the students were given some kind of feedback that pointed out their error. Because these are the only cases of nonlearning left when glossing and copying are eliminated, it appears that combining feedback with the use of appropriate studying strategies would virtually guarantee that whenever a student did not know a rule, they would have a learning event wherein they could learn that rule.

It is not clear why students so often avoided self-explanation and rule-based reasoning. Glossing and copying examples can get students through the problem solving quickly with relatively few errors; self-explanation and rule-based reasoning take more effort and produce mostly long-term gains (learning physics) and few short-term gains. Cost–benefit analyses of the self-explanation strategy was done by Pirolli and Recker (1994) and Recker and Pirolli (1995). Dweck (1986)

reviewed work showing that varying the reward structure can cause participants to switch from a performance orientation (short-term gains) to a learning orientation (long-term gains). More research is needed on this important issue.

However, even if students use strategies that maximize the frequency of learning events, they could construct an incorrect rule, or they could later forget whatever rule they had derived. The next section examines conditions under which robust learning of correct rules occurs during learning events.

Deep Versus Shallow Rule Construction

It seems plausible that deeper reasoning during a learning event would make it more likely that the student will construct a rule that is correct and easily remembered. This suggests classifying learning events according to how much reasoning occurred, then seeing if deeper reasoning is associated with more successful learning.

Two methods were used to make the deep–shallow distinction. For the learning events that occurred during example studying, learning events involving explanation-based learning of correctness (category I.B in Table 2) were placed in the deep reasoning category, whereas learning events in which the student just accepted the example line (category I.A) were placed in the shallow reasoning category. For the learning events that occurred during problem solving, the existing categories (i.e., whether the instructional material was consulted) seem independent of the depth distinction, so a new coding was done. Two coders rank-ordered the learning events from deep to shallow. They agreed on which learning events comprised the top five, but their rankings of the other learning events differed dramatically. Thus, the top five learning events (17, 18, 19, 22, and 24) were coded as deep, and the remaining 11 were coded as shallow.

Was deep rule construction better? Presumably, shallow reasoning is more likely to allow students to learn incorrect rules. To check this, the learning events were coded according to the correctness of the rule that the participant appeared to learn. There were 20 clear cases of correct rules, 5 clear cases of incorrect rules, and 3 cases in which it was unclear whether the rule was correct (see the coding in the Appendix). Table 4 shows which of the clear cases came from which kind

TABLE 4
Is Deeper Reasoning More Often Correct?

	Correct	*Incorrect*	*Total*
Deep reasoning	9	0	9
Shallow reasoning	11	5	16
Total	20	5	25

TABLE 5
Are Rules Produced by Deeper Reasoning Used on Subsequent Opportunities?

	Used	*Not Used*	*Total*
Deep reasoning	6	2	8
Shallow reasoning	12	3	15
Total	18	5	23

Table 6
Do Good Solvers Prefer Deeper Reasoning?

	Deep	*Shallow*	*Total*
Good solvers	10	8	18
Poor solvers	1	6	7
Total	11	14	25

of reasoning. As expected, deeper reasoning was more often associated with correct rules, although the effect was only marginally reliable ($p = .06$, Chi-squared).

Presumably, a rule that is constructed via deeper reasoning should be more easily recalled and reused than one constructed via shallow reasoning. Not only should the deeper reasoning enhance reconstructive recall, it could loosen the rule from its context and generalize it. Using Figure 3, learning events were coded as producing rules that were either remembered and used at a subsequent opportunity or not used at subsequent opportunities. In five cases, there were no further opportunities to use the rule, so it was unclear whether the rule was remembered (see the coding in the Appendix). Table 5 shows which of the clear cases came from which kind of reasoning. The expected trend appears to be absent ($p = .78$, Chi-squared).

Good students preferred deep reasoning more than poor did. Earlier studies have found that students who learn the most are doing self-explanation during example studying and minimizing copying during problem solving (Chi et al., 1989; VanLehn, in press). Because they use studying strategies that increase learning, it may be that they also use rule construction strategies that increase learning. Thus, it would be interesting to find out if the more effective learners preferred to use deep reasoning during their learning events. Using the median split of Chi et al. (1991), students were classified as effective learners (good solvers) and ineffective learners (poor solvers) on the basis of their errors during problem solving.[4] Table 6 shows how many learning events involved deep versus shallow reasoning for each

[4]P1, P2, P101, and P110 were classified as good solvers. P102, P103, P107, and P109 were classified as poor solvers. P105 was excluded. Adding P105 as either good or poor did not affect the results.

of the two groups of students. As hypothesized, the effective learners preferred to use deep reasoning, although the effect was only marginally reliable (p = .06, Chi-squared).

There are two plausible interpretations of this result. One is that deep reasoning *caused* better learning and hence reduced error rates. A second interpretation is that some students have a learning orientation, whereas others have a performance orientation (Dweck, 1986; Dweck & Leggett, 1988). Those who are motivated to learn the task domain prefer both productive studying strategies and deep reasoning. Those who are motivated to achieve fast, accurate performance at the possible expense of learning the task domain prefer glossing the examples, copying the example solutions to solve problems, and employing shallow reasoning during learning events. These data support both interpretations equally. Indeed, the two interpretations could both be true.

A Unified Explanation

Figure 6 displays most of the explanations discussed previously plus a few new ones. It is intended both to summarize and unify the explanations and to provide a focused set of hypotheses to guide future work.

Arrow 1 represents that less copying (transformational analogy) during problem solving causes students to use rule-based knowledge more, which causes them to have more impasses and learning events as they detect flaws in their knowledge. Arrow 2 represents that self-explaining more of the examples increases the number of learning events. Arrow 3 represents a speculation, as it could not be tested with these data, that students will have more impasse-driven learning events if they check their work more frequently while problem solving or if they receive feedback on their work. Checking and feedback should also reduce the number of incorrect rules learned, as indicated by Arrow 5. Arrow 6 indicates that deeper reasoning should prevent mislearning, as shown earlier. Arrow 7 indicates that deeper reasoning should increase the students' ability to use the rule after its construction. This trend did not reach significance, so it may not exist.

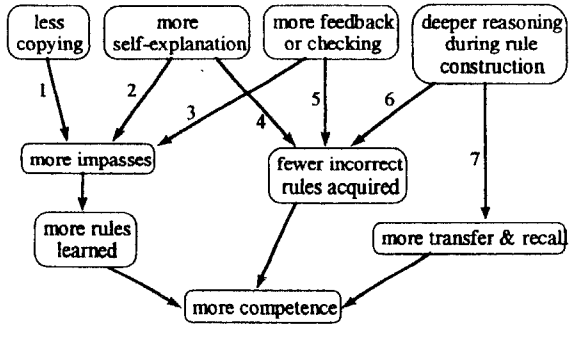


FIGURE 6 Four studying strategies and their effects on learning.

TABLE 7
Example Studying Promotes Correct Rule Acquisition More Than Problem Solving

	Correct	*Incorrect*	*Total*
Example studying	12	0	12
Problem solving	8	5	13
Total	20	5	25

Arrow 4 represents that increasing self-explanation should decrease mislearning, an influence that has not yet been discussed but is present nonetheless. The underlying idea is that when students are rederiving an example line, they have the line itself to guide them and keep them from making false assumptions. When learning events occur during problem solving, there is no such guidance, which increases the chance that the student will invent an incorrect rule. The data support this claim. Table 7 shows that the proportion of learning events where incorrect rules were acquired is higher during problem solving than during example studying (p = .02). Thus, as Arrow 4 indicates, the more self-explanation, the more rules learned during example studying, and the fewer rules remain to be learned during problem solving, and the better the chance of learning correct rules. Moreover, learning more rules during example studying increases the likelihood that the solver will stay on a correct solution path during problem solving, which in turn increases the chance that impasses during problem solving are worth learning from. This effect has been demonstrated with Cascade (see VanLehn et al., 1992, p. 25) but has not yet been tested empirically.

The four learning strategies shown at the top of Figure 6 may all ultimately cause more learning and more competence. Thus, they are appropriate goals for instruction. Not coincidentally, these are exactly the goals of the Andes tutoring system (Conati, Gertner, VanLehn, & Druzdzel, 1997; VanLehn, 1996).

DISCUSSION

Several issues raised in the literature on learning events are addressed by this study. They are discussed subsequently, followed by a brief summary of the overall claims.

Awareness of Learning Events

A seminal issue is whether learning takes place in discrete episodes (learning events) of which the participant is fully aware, or whether it occurs deep in some in-

accessible part of the cognitive architecture. This study partially addresses this issue in that it succeeded in finding learning events. However, the thorny issue of whether participants are aware of them as learning events was not analyzed here, in part because Cascade does not model the distinction between conscious and unconscious processing. Nonetheless, those interested in the awareness issue may find the Appendix useful because it often reports the participants' utterances at the time of the learning event.

Is All Learning Triggered by Impasses?

Another important issue is whether all learning events are trigger by impasses. This study provides clear evidence that some learning events are not triggered by impasses. In learning events 22 and 8, for instance, the participants deliberately checked their work by comparing it to examples, detected errors, and learned new rules in the process of correcting their errors. Although some learning events were not triggered by impasses, they were triggered by errors, albeit indirectly. Thus, the data are consistent with the more general hypothesis that all learning events are triggered by some kind of failure, such as impasses or errors. The data are thus consistent with the learning-by-self-debugging paradigm, which generally initiates learning only when some kind of failure is detected.

However, even that more general hypothesis might be too constraining because other studies have found learning events that apparently were triggered by neither impasses nor errors. Siegler and Jenkins (1989) conducted a longitudinal study of preschool children as they solved simple orally presented addition problems (e.g., "What is 5 + 3?"). All the participants spontaneously invented more efficient strategies. Their discoveries occurred on problems that were no harder than ones the children had solved earlier. In most cases, the children had solved exactly these same problems earlier. Moreover, none of the children said they were stuck at the time of the rule learning event. Thus, it is unlikely that the students reached an impasse. Rather, they seem to somehow have "just noticed" that they could use their fingers more efficiently. Jones and VanLehn (1994) demonstrated that a computational model can notice the same things that the Siegler and Jenkins' participants did and make the same sequence of strategy discoveries. Similar impasse-free learning events occurred when a participant learned how to solve the Tower of Hanoi puzzle (VanLehn, 1991). The participant just noticed a more efficient way to solve the puzzle. Ruiz and Newell (1989) developed a model of strategy acquisition for the Tower of Hanoi in which "noticing" plays an important role.

Although noticing, impasses, and errors are involuntary, students also can set out to deliberately form a new understanding of the task domain. A clear case occurred in a Tower of Hanoi protocol (VanLehn, 1991). Early in the protocol, the participant had successfully solved the puzzle but was unsatisfied because the experimenter's instructions were to find a "good solution procedure." The participant set about quite deliberately to find an improved procedure by first solving the trivial one-disk puzzle, then the two-disk puzzle, then the three-disk puzzle, and so on. After solving the two-disk puzzle, she paused to reflect on her solution and discovered a new rule. The participant was clearly not at an impasse, and she was not noticing something involuntarily. She was deliberately looking for a new strategy or at least a fragment of one.

Although the Tower of Hanoi participant alternated between finding a solution to the puzzle and reflecting on the solution, it seems possible for participants to interweave reflection and problem solving thoroughly, so that they devote a little attention after each action to noticing whether there is an opportunity for learning. For instance, when I begin a repetitive task, such as cutting 15 bookshelves, during the first few repetitions I deliberately look for shortcuts and guard against mistakes. A student adopting such a reflective mind-set while doing homework probably would learn much more than someone just focused on getting the job done. Perhaps these are the cognitive mechanisms behind the "reflective practitioners" lauded by Schon (1983) and the "learning oriented" students of Dweck (1986).

Summary

The results bring both good news and bad news for Cascade and similar impasse-driven learning models. The good news is that there are indeed learning events, and many of them can be modeled by the simple impasse–repair–reflect cycle. Participants seldom speak coherently about their learning events, but there is evidence for them even in verbal protocols, which are notorious in their incompleteness.

The bad news is that the impasse–repair–reflect cycle is too simple to model all the observed learning, and even the learning-as-self-debugging paradigm might be too simple. Human students appear to be capable of doing everything that good programmers do in developing their programs, except that students are developing their own knowledge. That is, good students can debug their knowledge and can even plan and execute tests that will detect flaws in their knowledge. Moreover, the literature suggests that students can notice opportunities for optimizing their knowledge and perhaps even use their knowledge in a kind of reflective, single-step mode that simultaneously checks the knowledge and executes it. Additionally, the analyses in this article show that participants probably bring the full range of their knowledge to bear on deciding when to learn (e.g., only learn from an impasse if confident in the correctness of the solution path) as well as what to learn. None of this seems beyond the power of cognitive modeling, although it is more complex than any existing model, even recent models from the learning-as-self-debugging paradigm (e.g., Ram & Cox, 1994; Ram et al., 1995). As a

target for future modeling efforts, the learning events have been included as an Appendix.

If one does build a more powerful model of learning events, it would be beneficial to implement it on top of an impasse-based memory model, such as SOAR or ACT–R. This would go a long way toward clarifying the distinction between low-level and high-level impasse-driven learning.

Last, it was unexpected and a bit discouraging to find that students had so few learning events and that they often preferred to reason shallowly during them. Fortunately, the data suggested several strategies that should remedy this situation: self-explaining examples, avoiding copying during problem solving, obtaining feedback during problem solving, and reasoning more deeply during learning events.

ACKNOWLEDGMENTS

This research was supported by the Cognitive Science Division of Office of Naval Research under Grants N00014–92–J–1945 and N00014–94–1–0674. Preparation of the manuscript was completed while the author was a Fellow at the Center for Advanced Study in the Behavioral Sciences, supported by Spencer Foundation Grant 199400132. I am grateful to Charles Murray, Stephanie Siler, Peter Pirolli, Mimi Recker, and Ashwin Ram for their comments on the manuscript.

REFERENCES

Anderson, J. R. (1982). The acquisition of cognitive skill. *Psychological Review, 89,* 369–406.

Anderson, J. R. (1990). *The adaptive character of thought.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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

Bergadano, F., Giordanna, A., & Ponsero, S. (1989). Deduction in top-down inductive learning. In A. M. Segre (Ed.), *Proceedings of the Sixth International Workshop on Machine Learning.* Los Altos, CA: Morgan Kaufman.

Bielaczyc, K., Pirolli, P., & Brown, A. L. (1995). Training in self-explanation and self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem-solving. *Cognition and Instruction, 13,* 221–252.

Bundy, A., Byrd, L., Luger, G., Mellish, C., & Palmer, M. (1979). Solving mechanics problems using meta-level inference. In *Proceedings of the Sixth International Joint Conference on AI* (pp. 1017–1027). San Mateo, CA: Morgan Kaufmann.

Carbonell, J. G. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 137–162). Palo Alto, CA: Tioga.

Carbonell, J. G. (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 artificial intelligence approach* (Vol. 2, pp. 371–392). San Mateo, CA: Morgan Kaufman.

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, 15,* 145–182.

Chi, M.T.H., de Leeuw, N. D., Chiu, M. H., & LaVancher, C. (1994). Eliciting self-explanations improves understanding. *Cognitive Science, 18,* 439–477.

Chi, M.T.H., & VanLehn, K. (1991). The content of physics self-explanations. *The Journal of the Learning Sciences, 1,* 69–105.

Conati, C., & Fain Lehman, J. (1993). EFH–Soar: Modeling education in highly interactive microworlds. *Lecture notes in artificial intelligence.* New York: Springer-Verlag.

Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. (1997). On-line student modeling for coached problem solving using Bayesian networks. In A. Jameson, C. Paris, & C. Tasso (Eds.), *User modeling: Proceedings of the Sixth International Conference, UM97.* New York: Spring Wien.

de Kleer, J. (1975). *Qualitative and quantitative knowledge in classical mechanics* (Tech. Rep. No. AI–TR–352). Cambridge, MA: MIT AI Laboratory.

di Sessa, A. A. (1993). Towards an epistemology of physics. *Cognition and Instruction, 10,* 105–225.

Dweck, C. S. (1986). Motivational processes affecting learning. *American Psychologist, 41,* 1040–1048.

Dweck, C. S., & Leggett, E. L. (1988). A social-cognitive approach to motivation and personality. *Psychological Review, 95,* 256–273.

Elio, R., & Scharf, P. B. (1990). Modeling novice-to-expert shifts in problem-solving strategy and knowledge organization. *Cognitive Science, 14,* 579–639.

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

Halliday, D., & Resnick, R. (1981). *Fundamentals of physics* (2nd ed.). New York: Wiley.

Halloun, I. A., & Hestenes, D. (1985). Common sense concepts about motion. *American Journal of Physics, 53,* 1056–1065.

Jones, R. (1989). *A model of retrieval in problem solving.* Unpublished doctoral dissertation, University of California, Irvine.

Jones, R. M., & VanLehn, K. (1992). A fine-grained model of skill acquisition. In J. Kruschke (Ed.), *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 873–878). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Jones, R. M., & VanLehn, K. (1994). Acquisition of children's addition strategies: A model of impasse-free, knowledge-level learning. *Machine Learning, 16,* 11–36.

Lamberts, K. (1990). A hybrid model of learning to solve physics problems. *European Journal of Cognitive Psychology, 3,* 151–170.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 99–161). Cambridge, MA: MIT Press.

Langley, P., & Allen, J. (1993). A unified framework for planning and learning. In S. Minton (Ed.), *Machine learning methods for planning.* San Mateo, CA: Morgan Kaufmann.

Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., & Thompson, K. (1991). A design for the Icarus architecture. *SIGART Bulletin, 2,* 104–109.

Larkin, J. H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition* (pp. 311–334). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Lawler, R. W. (1991). The progressive construction of mind. *Cognitive Science, 5,* 1–30.

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

Lovett, M. C. (1992). Learning by problem solving versus by examples: The benefits of generating and receiving information. In J. Kruschke (Ed.), *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 956–961). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

McCloskey, M., Caramazza, A., & Green, B. (1980). Curvilinear motion in the absence of external forces: Naive beliefs about the motion of objects. *Science, 210,* 1139–1141.

McDermott, J., & Larkin, J. H. (1978). Re-representing textbook physics problems. In *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence* (pp. 156–164). Toronto, Canada.

Neches, R. (1987). Learning through incremental refinement of procedures. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development* (pp. 163–219). Cambridge, MA: MIT Press.

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

Novak, G. S., & Araya, A. A. (1980). Research on expert problem solving in physics. In *Proceedings of the First National Conference on Artificial Intelligence* (pp. 178–180). Menlo Park, CA: AAAI.

Ohlsson, S. (1987). Transfer of training in procedural learning: A matter of conjectures and refutations? In L. Bolc (Ed.), *Computational models of learning* (pp. 55–88). New York: Springer-Verlag.

Ohlsson, S. (1993). The interaction between knowledge and practice in the acquisition of cognitive skills. In A. Merowitz & S. Chipman (Eds.), *Cognitive models of complex learning* (pp. 147–208). Dordrecht, The Netherlands: Kluwer.

Ohlsson, S. (1996). Learning from performance errors. *Psychological Review, 103*, 241–262.

Pazzani, M. J. (1990). *Creating a memory of causal relationships: An integration of empirical and explanation-based learning methods*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Pazzani, M., Myer, M., & Flowers, M. (1986). The role of prior causal theories in generalization. In T. Kehler, S. Rosenschein, R. Filman, & P. F. Patel-Schneider (Eds.), *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 545–550). Menlo Park, CA: Morgan Kaufman.

Pirolli, P., & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 459–457). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Pirolli, P., & Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction, 12*, 235–275.

Ploetzner, R., & VanLehn, K. (1997). The acquisition of informal physics knowledge during formal physics training. *Cognition and Instruction, 15*, 169–206.

Priest, A. G., & Lindsay, R. O. (1992). New light on novice–expert differences in physics problem solving. *British Journal of Psychology, 83*, 389–405.

Ram, A. (1990). Incremental learning of explanation patterns and their indices. In B. Porter & R. Mooney (Eds.), *Machine learning: Proceedings of the Seventh International Conference* (pp. 313–320). Los Altos, CA: Morgan Kaufman.

Ram, A., & Cox, M. (1994). Introspective reasoning using meta-explanations for multistrategy learning. In R. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach* (Vol. 4, pp. 349–377). San Mateo, CA: Morgan Kaufmann.

Ram, A., Narayanan, S., & Cox, M. T. (1995). Learning to troubleshoot: Multistrategy learning of diagnostic knowledge for a real-world problem-solving task. *Cognitive Science, 19*, 289–340.

Recker, M. M., & Pirolli, P. (1995). Modeling individual differences in students' learning strategies. *The Journal of the Learning Sciences, 4*, 1–38.

Reimann, P., & Schult, T. J. (1993). Understanding and using worked-out examples: A computational model. In G. S. a. K. F. Wender (Ed.), *The cognitive psychology of knowledge* (pp. 177–201). Amsterdam: Elsevier.

Reimann, P., Wichmann, S., & Schult, T., J. (1993). A learning strategy model for worked-out examples. In P. Brna, S. Ohlsson, & H. Pain (Eds.), *Artificial intelligence in education: Proceedings of AI–ED93* (pp. 290–297). Charlottesville, VA: Association for the Advancement of Computing in Education.

Renkl, A. (in press). Learning from worked-examples: A study on individual differences. *Cognitive Science*.

Ruiz, D., & Newell, A. (1989). Tower noticing triggers strategy change in the Tower of Hanoi: A Soar model. In G. Ohlsson & E. Smith (Eds.), *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 522–529). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Los Altos, CA: Morgan Kaufman.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Schoenfeld, A. H., Smith, J. P., & Arcavi, A. (1993). Learning: The microgenetic analysis of one student's evolving understanding of a complex subject matter domain. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 4, pp. 55–177). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Schon, D. A. (1983). *The reflective practitioner: How professionals think in action*. New York: Basic Books.

Shrager, J. (1987). Theory change via view applications in instructionless learning. *Machine Learning, 2*, 247–276.

Shrager, J. (1990). Commonsense perception and the psychology of theory formation. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 437–470). San Mateo, CA: Morgan Kaufmann.

Shrager, J., & Klahr, D. (1986). Instructionless learning about a complex device: The paradigm and observations. *International Journal of Man–Machine Studies, 25*, 153–189.

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

Sussman, G. J. (1975). *A computer model of skill acquisition*. New York: Elsevier.

VanLehn, K. (1987). Learning one subprocedure per lesson. *Artificial Intelligence, 31*, 1–40.

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

VanLehn, K. (1996). Conceptual and meta learning during coached problem solving. In C. Frasson, G. Gauthier, & A. Lesgold (Eds.), *ITS96: Proceedings of the Third International Conference on Intelligent Tutoring Systems* (pp. 29–47). New York: Springer-Verlag.

VanLehn, K. (in press). Analogy events: How examples are used during problem solving. *Cognitive Science*.

VanLehn, K., Ball, W., & Kowalski, B. (1990). Explanation-based learning of correctness: Towards a model of the self-explanation effect. In M. Piattelli-Palmarini (Ed.), *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 717–724) . Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

VanLehn, K., & Jones, R. M. (1993a). Better learners use analogical problem solving sparingly. In P. E. Utgoff (Ed.), *Machine learning: Proceedings of the Tenth Annual Conference* (pp. 338–345). San Mateo, CA: Morgan Kaufmann.

VanLehn, K., & Jones, R. M. (1993b). Integration of analogical search control and explanation-based learning of correctness. In S. Minton (Ed.), *Machine learning methods for planning* (pp. 273–315). Los Altos, CA: Morgan Kaufmann.

VanLehn, K., & Jones, R. M. (1993c). Learning by explaining examples to oneself: A computational model. In S. Chipman & A. Meyrowitz (Eds.), *Cognitive models of complex learning* (pp. 25–82). Dordrecht, The Netherlands: Kluwer.

VanLehn, K., & Jones, R. M. (1993d). What mediates the self-explanation effect? Knowledge gaps, schemas or analogies? In M. Polson (Ed.), *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 1034–1039). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

VanLehn, K., Jones, R. M., & Chi, M.T.H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*, 1–59.

Widmer, G. (1989). A tight integration of deductive and inductive learning. In A. M. Segre (Ed.), *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 11–13). Los Altos, CA: Morgan Kaufman.

APPENDIX
The Learning Events Themselves

This appendix describes each of the observed learning events along with the codes given them. The triggering code indicates the condition that probably initiated the learning event. The reasoning code is a coarse categorization of the types of reasoning that occurred during the learning event. The outcome code indicates whether the acquired rule is correct and whether it was used later (memorable) or not (forgettable). The "correct" code must be understood as meaning "nothing obviously wrong," as opposed to "deeply and completely understood." When protocols are being quoted, text in square brackets indicates actions taken by the participant or comments by the experimenter (E). The numbers in the headings correspond to those in Figure 3.

1. P2 Learns the Negative Component Rule

Participant P2 initially was unfamiliar with the rule that the component of a vector is negative if the angle used to specify it is attached to the negative portion of an axis. As she self-explained the equations of the Three-Strings Example (Figure 5), she failed to notice the negative sign at the first application of this rule, which occurred in the derivation of the equation $F_{ax} = -F_a\, cos(30°) = -0.866\ F_a$. However, when she reached the next application of the rule, on the equation $F_{cy} = -F_c = -W$, she said:

> Hmm, why is minus? Uh hun. Because these axes are starting here so this is minus. Then it's minus *W*, minus *W*. *W* bound with object weight. Weight is a force, OK? OK. And this is minus *W*. So this is the other direction. What about the *X*'s. It should also be minus. Yah, that was a minus. [E: What are you thinking about now?] Minus *FA*. For the *X*'s, I want to check if it was minus ... one of the forces was minus the other. *FAX* equals minus.

The explanation of the minus sign is not immediately obvious to her, so the triggering code is **impasse while trying to explain an example line.** She constructed a correct, albeit abbreviated, explanation. Interestingly, P2 went on to check that F_{ax}, the component of the other force that is near a negative axis, was also negative. She later checked signs while explaining the two subsequent examples. This suggests that the participant actually hypothesized and verified the rule during this impasse, rather than recalling it, so the reasoning code is **explanation-based learning of correctness.** The participant then used the rule correctly on every problem where it was applicable. Thus, the outcome code is **correct and memorable.**

2. P101 Learns an Incorrect Negative Component Rule

The participant failed to notice the negative signs in component equations throughout the examples, and he avoided the sign rule by using analogy during the first two problems. On the third problem, he reexamined the three-strings example more carefully, and this time he noticed the negative signs on the component equations, for example, $F_{ax} = -F_a\, cos(30°) = -0.866\ F_a$. He said, "I'm trying to figure out why these are negative." He did not find an explanation, so he decided just to use the equations verbatim, although he commented, "This is called copying too much from the book. I hate that." The triggering code for this learning event is **impasse while self-explaining before doing analogy.** The reasoning code is **analogy to an example.** Shortly thereafter, the participant self-explained another negative sign, and again he did not seem to realize that the source of the negative sign was the negative portion of the axis. Thus, he had at best a very specific and perhaps even incorrect understanding of the negative projection rule. It is not clear whether he retained it because on the next problem he learned a different rule for negative projections. Thus, the outcome code is **incorrect and unclear.**

3. P101 Learns Another Incorrect Negative Projection Rule

On the next problem, P101 again used analogy to obtain a negative expression, $a_x = -g\ sin\ \theta = -.422$. Here he said:

> I don't know why it's negative ... except I think they said before, looking back over here, that, oh, okay. It doesn't matter. Acceleration is not relative. That's positive. [Crosses out negative sign on .422.] 'Cause they were saying this is the positive *X* direction and that's the negative *X* direction, so it really doesn't matter, the fact that it has a negative or positive, to the real world.

The participant was not immediately able explain the example's negative sign, which he had copied to the current problem, so the triggering code is **impasse while self-explaining before doing analogy.** The participant got information from the example, so reasoning code is **analogy to an example.**

However, the participant still did not learn a completely correct rule. He inferred a connection between the direction along the axis and the sign of component expressions, but seemed to believe that one can adjust the signs arbitrarily to suit the directions. Evidence for this occurs on a subsequent problem, where he said, "You get *A* equals minus, I don't care if it's negative or not. That's not important." This shows that he did retain his rule, so the outcome code is **incorrect and memorable.**

4. P1 Learns the Negative Component Rule

While self-explaining the Three-Strings Example (Figure 5), participant P1 explained the equation $F_{ax} = -F_a\,cos(30°) = -0.866\,F_a$. She reached an impasse trying to explain the negative sign as she says, "Ohhh, how did they get that negative in there?" The participant's first attempt to resolve the impasse was to check her work, looking for sign errors. Finding none, she tries to explain the negative sign by looking for a trigonometric identity because of "the trig functions being negative and positive for no apparent reason." The participant finds no trigonometric identity that accounts for the sign, so her third idea is that $cos(30°)$ might evaluate to a negative number, but she realizes the inadequacy of this explanation before looking up the value of $cos(30°)$. The participant finally succeeds in explaining the negative sign by noticing that the x-component of the vector is in the negative direction:

> Hmmm, negative cosine 30, why would they say ahhh, ummm ... The ohh, okay maybe it's just because the *A* component is the *X* component of force *A* is negative. So they just. Well okay I'll, I'll try that for awhile. Let's see if that works, cause that makes sense. [E: What makes sense?] The reason the negative is there is because the *X* component is in the negative direction on the *X*-axis.

Although the participant's broken speech in the vicinity of the discovery does not reveal exactly what her thinking processes are, it is plausible that she applied an overly general rule of mathematics, which is that operations often preserve negative signs. Thus, the negative sign associated with the negative *X*-axis is preserved by the projection operation and emerges as a negative sign in the equation. The reasoning code is thus **explanation-based learning of correctness** and the triggering code is **impasse while explaining an example line**.

A few minutes later, the participant used the contrapositive of the rule to explain the equations $F_{bx} = F_b cos(45°) = 0.707F_b$ and $F_{by} = F_b sin(45°) = 0.707F_b$. She said, "It's not negative because they're both in the positive direction." Almost immediately, she again used the rule to explain the equation $F_{cy} = -F_c = -W$. However, she complained, "I'm still not sure where they get negative?" This is more evidence that the participant used some kind of heuristic, uncertain reasoning (such as explanation-based learning of correctness) to infer her rule, even though she was applying the rule successfully. At any rate, the participant had clearly learned a new, correct rule, and she continued to use it on most of the subsequent examples and problems, so the outcome code is **correct and memorable**.

5. P102 Learns the Negative Component Rule

P102 was explaining the Three-Strings Problem (Figure 5) when she noticed the negative sign on the equation $F_{cy} = -F_c = -W$. She paused after first encountering the negative sign, read a little further in the hopes of clarifying the confusion, then paused again. (Hence, the triggering code is **impasse while trying to explain an example line**.) Then she said, "Negative *W*. It's because it's going in a negative direction it points, they give it a negative value; it's below the *Y*-axis. I mean the *X*-axis." Because this participant said very little, it is possible that she simply had trouble recalling the negative sign rule. However, that rule was not mentioned in the text. Moreover, this participant was generally rather taciturn, which would also explain the lack of verbal evidence of reasoning. Thus, it seems likely that the participant inferred the negative sign rule in the same way that participant P1 did. Thus, the reasoning code is **explanation-based learning of correctness.** The rule is used often later, so the outcome code is **correct and memorable**.

6. P2 Learns the Normal Force Rule

P2 studied an example in which a block was held still on an inclined plane by a rope attached to a wall (Figure 4). When P2 read the example's explanation of the normal force, which was indicated by a vector labeled *N* on the force diagram, she said:

> [Reads: And *N* is the force exerted on the block by the inclined surface...] Hmm... Why? Why is it like this? I don't know. [E: Why is what like this?] Why this is a direction... Why the direction should be this way? How it is pushing, the block is pushing the incline plane here so the incline, so the opposite direction should be here? Where is the opposite to the... to the force to the weight? Ah... It can't be on the same body, it's on the earth. OK, I think I understood. I'm not sure, but I think I understood. OK.

P2 was clearly unable to explain the vector's direction, so the triggering code is **impasse while trying to explain an example line**. Her common sense told her that the block is pushing on the plane, and she used an overly general rule to sanction treating this push as a proper physical force. She then used Newton's third law to explain that the normal force was in the opposite direction because it is acting on the block, whereas the pushing force as acting on the plane (which she called earth). Thus, the reasoning code is **explanation-based learning of correctness**.

However, the rule she invented was only used once, on the first isomorph to this example. (Her protocol is missing, but a normal force appears on her force diagram.) On three subsequent problems, she did not include a normal force, even though she solved the problems correctly. This appears to be a case of a rule being learned, then forgotten. Thus, the outcome code is **correct and forgettable**.

7. P1 Learns the Normal Force Rule

As P1 self-explained the inclined plane example (Figure 4), she reached the example lines that explained the normal force that appeared on the force diagram (right

half of Figure 4). She paused after reading for so long that the experimenter eventually asked her what she was thinking. Her response indicated that she was unfamiliar with the normal force concept. However, she did not try to justify the existence of the normal force. Thus, the triggering code is **impasse while trying to explain an example line** and the reasoning code is **just accepting** the example's assertion.

On the first problem where normal forces are relevant, the participant deliberately did not refer to the inclined plane example, but as she drew the normal force on the force diagram, she said, "And the *Y* direction is the normal force.... 'Cause I remember exactly how this was done in the example problems. And I wouldn't want to go against the example." Clearly, the participant learned enough about the normal force during the example to use that knowledge during the subsequent problem, but the participant's self-deprecating comment suggests she had a rote visual or case-like memory rather than a well-understood rule. Nonetheless, she used the normal force rule correctly every time it was applicable. Thus, the outcome code is **correct and memorable**.

8. P105 Learns the Normal Force Rule

P105 did not self-explain much of the inclined plane example (Figure 4), and he apparently did not learn the normal force rule. On the first isomorphic problem to the inclined plane example, the participant drew, without looking at the example, a force diagram that left out the normal force. Then the participant checked his diagram by referring to the example while saying:

> I'm ahh ... looking at example 6. Ummm... (pause) Let's see. [E: Please talk.] Ummm. Okay. Sss ... Says [Reads: *N* is the force exerted on the block by the inclined surface.] Okay, so that would be perpendicular to it. [Draws normal force.] That's the force I wasn't thinking of.

The participant learned a rule from this because he correctly drew a normal force on the next two incline problems without referring to the example. The triggering code is **checking one's work against the example**, the reasoning code is **analogy to an example**, and the outcome code is **correct and memorable**.

9. P2 Learns the Rotate Axes Rule

P2 self-explained every line of the Three-Strings Example. When the example described the force diagram (see Figure 5, right half), she said:

> [Reads: Choosing the *X*- and *Y*-axis as shown] (pause) OK. [E: What are you thinking about there?] How did they pick it? So it's convenient because you save one of the ... You put one of the axes on one of the forces ... line. On the *W* force. And it's convenient because that's how we are used to look at the picture. OK.

The pause is a sign of an impasse, so the triggering code is **impasse while explaining an example line**. She devised two explanations for why the axes are positioned as shown. The first explanation corresponds to the rule that rotates the axes so that the maximal number of forces lie along axes. It is unlikely that the participant knew this rule already because it is not in the textbook and most math courses never rotate the axes. However, this participant was fairly good at trigonometry, so she probably realized that aligning the axes with the vectors reduces the number of trigonometric functions required for expressing the components of the vectors. This was confirmed in the next example (see Figure 3), where she explicitly stated the reasons for rotating the axes:

> [Reads: It is convenient to choose the *X*-axis of our reference frame to be along the incline.] Why is it convenient? So we can save one force. What force we save? We save the *F* force. [E: We save an *F* force?] I save it so, that I don't have to calculate it by, with angles. And I save one. I don't have 2 components for it, I have only 1 component. So, we pick the axis so it will lie with one of the forces in line, at least with one of the forces. OK. So.

Later P2 realized that she actually saves two forces. On all subsequent inclined plane problems, she rotated the axes. Clearly, she learned the rule, so the outcome code is **correct and memorable**. The reasoning code is **explanation-based learning of correctness** because the rules used to determine that a force would be saved by aligning it with the axis seem to be heuristic rules of mathematics.

10. P1 Learns the Rotate Axes Rule

As P1 self-explained the inclined plane example (Figure 4), she came to the example line that explained the rotated axes in the force diagram (right half of Figure 4) and said:

> [Reads: 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.] So they're going to, *X*-axis along the incline, and they, they don't show it, but ... .

The participant uses rotated axes on all other relevant occasions, so she seems to have learned a general rule, but it is not clear whether she appreciates why rotating the axes is convenient. Thus, the reasoning code for this episode is **just accepting**

the example's assertion, the triggering code is **impasse while trying to explain an example line**, and the outcome code is **correct and memorable**.

11. P105 Learns the Rotate Axes Rule

When first looking at the force diagram of the inclined plane example (right half of Figure 4), P105 said, "I don't see why they put the *X*-axis parallel to the incline," so the triggering code is **impasse while trying to explain an example line**. Later, he pauses after reading the example's explanation of the rotated axes, but makes no comment. Apparently, he learned the rule from reading the example because he rotated the axes on a subsequent problem without referring to the example. The triggering code is **just accepting** the example's assertion, and the outcome code is **correct and memorable**.

12. P2 Learns the Knot Rule

As P2 studied the Three-Strings Example (Figure 5), she said:

> [Reads: Consider the knot at the junction of the 3 strings to be the body.] Why the knot is the body? [E: What are you thinking?] Why the knot is the body? [Reads: Consider the knot at the junction of the 3 strings to be the body.] Why this should be the body? I thought *W* was the body. OK, let's see later.

P2 was unable to explain the example's choice of body, so the triggering code is **impasse while trying to explain an example line**. However, she never seemed to find an explanation, and never mentioned the knot again. Yet she had no trouble using a knot as a body during the four subsequent problems, so she clearly learned something from this episode. The reasoning code is **just accepting** the example's assertion, and the outcome code is **correct and memorable**.

13. P1 Learns the Knot Rule

While studying the Three-Strings Example (Figure 5), P1 said:

> [Reads: Consider the knot at the junction of the three string to be the body.] That's strange. The body. [E: And just feel free to write on the diagrams.] Okay. [Reads: The body remains at rest under the action of the three forces shown in figure 5–6*b*.] Okay this is, well I'll just note this as strange to begin with. Because they are considering a knot, a body and I would automatically assume that they would be interested in weight than they apparently they're not here.

The participant was unable to explain the choice of the knot as the body, so the triggering code is **impasse while trying to explain an example line**. Nonetheless, she did use knots as bodies in several subsequent problems. Thus, the reasoning code is **just accepting** the example's assertion, and the outcome code is **correct and memorable**.

14. P1 Learns the Tension Rule

P1 initially did not differentiate tension forces, which are vectors, from tensions in strings and ropes, which are scalar properties of objects. She did not notice the dissonance when an example uses "tension in the string" to refer to the magnitude of a tension force. During several problems, P1 often referred to "tension force" but never to "tension in the string." When the problem statement gave "tension in the string," she wrote it on the force diagram as the magnitude of the tension force. However, she finally noticed the difference while trying to solve the first problem that had "tension in the string" as a *sought* quantity:

> One thing that's different in this problem than the others, is that it talks about the tension in rope *A*. And ... rather than just force. But ... I don't ... I think that's the ... I think tension and force are the same thing in rope *A*. So ... I'm just going to proceed accordingly.

The long pauses are evidence of an impasse as she tried to find a rule that will deliver a value for the sought quantity, so the triggering code is **impasse during self-explanation of the problem statement**.

When a subsequent problem also sought tension in the string, she had no difficulties, so apparently, she had learned a new rule here (the outcome code is **correct and memorable**). She probably relied on the similarity of the two quantities' names, which is an application of an overly general rule ("Two quantities with similar names and related objects are probably equal."). Thus, the reasoning code is **explanation-based learning of correctness**.

15. P109 Learns the Wrong Tension Rule

P109 had a very algebraic style of problem solving. She focused on finding equations that had variables with the right type of quantities. Her first problem, which was isomorphic to the Three-Strings Example (Figure 5), said "The tension in string 1 is 18 *N*." The participant had copied several equations from the example, but the example uses subscripted *F* variables instead of *T* variables, and always referred to the forces as "forces due to the string" rather than "tension forces." Because the given information was tension, but the equations contained only forces due to strings, the participant got stuck (so the triggering code is **impasse due to no**

rule exists for the current goal). The participant began to leaf through the textbook, and chanced on an example of a block hanging from a string. One of the equations in the example was $F = T + W$. P109 ignored the fact that this equation only holds for this example, and used it to convert tension to force. On three subsequent problems, the participant again applied the equation $F = T + W$. This learning event resulted in the acquisition of a wildly incorrect, albeit general rule. Thus, the outcome code is **incorrect and memorable**, and the reasoning code is **analogy to an example**.

16. P109 Almost Learns the Tension Rule

Although P109 usually used the equation $F = T + W$ to convert tension to force, she did something different on the second problem that was isomorphic to the inclined plane problem (Figure 4). In that problem, the tension was sought instead of given. Perhaps that caused her to not use the $F = T + W$ equation. At any rate, she tried to apply the inclined plane example to this problem, but it only discussed "the force F exerted on the block by the string" and never tension. She said:

> [Reads: F is the force exerted on the block by the string.] So that would also be the tension on the string. So, I think so. Looking at page 6, yeah, I don't feel there are any other mention of tension, and there has to be tension in this problem, so. Sounds like it.

The participant mapped tension in this problem to F in the example, which is correct and could eventually lead to learning the correct rule for tension. Thus, the episode's triggering code is **impasse while self-explaining before doing analogy**, and the reasoning code is **analogy to example**. Unfortunately, she never made this inference again, and used the equation $F = T + W$ instead. Thus, the outcome code is **correct and forgettable**.

17. P2 Learns the Compression Force Rule

P2 was solving the following problem: "A block is being held in place on an inclined, frictionless surface ($\theta = 40°$) by a massless spring. If the mass of the block is 10 kg, what force must the spring be exerting on the block?" The diagram for this problem looked just like the one for the inclined plane example (Figure 4) except that there was a spring below the block instead of a string above it. Thus, the block compressed the spring, so the spring exerted a compression force on the block.

After reading the problem, the participant paused and said, "Ummm" The text did not mention compressed springs, so the participant probably had no prior familiarity with compression forces, and the problem did not state that the spring was compressed. On the other hand, the problem did ask about a force exerted by the spring on the block, so P2 probably realized that some force exists. The participant's pause suggests an impasse, probably because she wanted to draw the force but did not know what direction it should go or how big it should be. Thus, the triggering code is **impasse during self-explanation of the problem statement**. Her next utterance was:

> So, the block wants to fall down. This is the force of the block in this direction. [Draws an arrow from the block parallel to the incline, pointing downward.]

The force drawn by the participant corresponded to the net force acting on the block if the spring were removed. Although the participant could have used formal physics rules to infer this, the phrase "wants to fall down" suggests some kind of informal physics reasoning. The participant's next utterance was:

> And the spring is giving exactly the opposite one. This direction. [Draws an arrow from the spring parallel to the incline, pointing upward.]

The force drawn by the participant could be produced via the target rule, "A compressed spring exerts a force on the objects compressing it," along with a commonsense inference that the spring is compressed. However, the phrase "exactly the opposite" and the counterfactual context of the preceding utterance makes it more likely that the participant used a rule such as, "If a body is at rest, but removing an object that touches it would allow it to move in response to a certain force F, then the object must exert a force on the body that is equal and opposite to F." This rule is a highly specific, qualitative version of Newton's first law. It is also a version of di Sessa's (1993) balancing p-prim. In summary, the reasoning code is ***counterfactual reasoning***.

Despite the admirable line of reasoning, the participant probably did not learn the target rule. Her line of reasoning did not mention the fact that the spring was compressed, for instance. However, there were no other opportunities to use compression forces in subsequent problems, so it is not clear whether she formed a general rule, and what kind of rule it is. Thus, the outcome code is **unclear and unclear**.

18. P1 Learns the Compression Force Rule

After P1 read the spring problem (described in the preceding learning event), she said: "Now this doesn't seem like a possible physical situation. It seems that the spring is in the wrong place. I don't see how the spring can be holding it where it is. Obviously, no spring that I know of." The participant reached an impasse trying to

explain the force mentioned in the problem statement, and hence the triggering code is **impasse during self-explanation of the problem statement**.

The participant went on for many lines comparing this problem and the inclined plane example, complaining all the while about the physical impossibility of a compressed spring (e.g., "What kind of spring wants to expand rather than contract?"). Finally, she said:

> I'm going to consider the situation without ... the spring in it. The net force would be ... Well ... I didn't draw these proportionally. It would be moving. ... Let's see. ... Vector composition. The block would move down so ... I'm going to draw the forces pointing up ... even though springs don't work that way.

Like P2, she reasoned counterfactually that if the force were not there, the block would move. Analogy alone was not sufficient to convince her to use the force. Thus, the reasoning code is **counterfactual reasoning**. As with P2, this counterfactual line of reasoning probably did not produce the target rule, but there was no chance for P1 to apply whatever she learns from this episode, so the outcome code is **unclear and unclear**.

19. P2 Learns Friction Force

P2 read the following problem: "A fireman weighing 160 pounds slides down a vertical pole with an average acceleration of 10 ft/s^2. What is the average vertical force he exerts on the pole?". She then said:

> Why should he exert any vertical force? Weighing ... The pole is vertical. And he's sliding down. I don't think there is any vertical force. [Omit 13 lines where E explains that fire stations have fire poles which the firemen slide down to respond quickly.] And ... why should they exert any force on the pole? [E: Ahhh] Any vertical force on the ... on the pole? (pause) Otherwise it will be free fall. So ... so they hop ... They are holding the ... I see. They are holding the pole so that not to fall a free fall. So there is something left. So this is the force. Okay. I'll try this hypothesis.

Although frictional forces were mentioned briefly in two places in the textbook, the text reserves its major discussion of friction for a later chapter. Thus, it is likely that the participant did not know about frictional forces in general. She certainly cannot see the friction here. There was a long pause, which warrants the triggering code of **impasse during self-explanation of the problem statement**. She then hypothesized that there must be a vertical force to counteract gravitational force and prevent free fall. This is an interesting example of counterfactual reasoning with Newton's second law, so the reasoning code is **counterfactual reasoning**.

There is only one other problem involving frictional force, which begins by mentioning it explicitly, "Determine the frictional force of the air on a body of mass." After reading this, P2 immediately recognized the similarity of this problem to the fire pole problem and said, "So it's again ... to find the difference between the weights and the actual fall." Apparently, she learned something fairly general from the fire pole problem, so the outcome code is **correct and memorable**.

20. P2 Learns the Taut-String Acceleration Rule

While studying the pulley example (Figure 1), P2 had a bit of trouble understanding the line that introduced the equality of the acceleration of the two blocks:

> [Reads: The acceleration of M2 must be $-A$.] Huh. Of course. [E: Why is that? What did you say?] The acceleration here must be $-A$. I said of course. [E: OK.] Not that I will think about it myself, but yes, I agree.

The initial "Huh" plus the fact that the participant said she would not think of drawing that conclusion herself justifies assigning this episode a triggering code of **impasse while trying to explain an example line**.

It is difficult to tell how the participant justified this inference to herself. Given her last comment, it is doubtful that she simply recalled it, especially given that it was not mentioned in the textbook. It is likely that she was doing the same reasoning as P105, but much more rapidly and without commenting on it. That is, she probably reasoned that because the two blocks were connected by a taut string, their displacements must be the same, so their speeds must be the same, so their accelerations must be the same. Thus, the reasoning code is **explanation-based learning of correctness**. Because she used what she learned on every possible subsequent occasion, the outcome code is **correct and memorable**.

21. P105 Learns the Taut-String Acceleration Rule

While studying the pulley example (Figure 1), P105 had a problem understanding why the two blocks have the same acceleration:

> [Reads: If the acceleration of m1 is A, the acceleration of m2 must be $-A$. Slight pause.] Oh—Okay, that makes sense because they're connected to a pulley. And if one's moving one speed, then the other one has to be moving at the same speed.

The triggering code is **impasse while trying to explain an example line**. The reasoning code is **explanation-based learning of correctness** because the participant

seems to deduce the equality of the acceleration from a series of correct commonsense inferences. It is not clear whether the participant formed a general rule because he used analogy on all pulley system problems, so the outcome code is **correct and unclear**.

22. P2 Learns the Pressure Force Rule

P2 solved a problem that was isomorphic to the pulley example (Figure 1), but involved a U-shaped tube filled with a massless, incompressible fluid that support two blocks resting on massless, frictionless seals. Each block had two forces on it, a gravitational force pulling it downward and a pressure force pushing it upward. The participant first solved the whole problem without the pressure forces, but got a counterintuitive answer, that both blocks accelerated downward, so she announced that she could not solve the problem. Thus, the triggering code is **counterintuitive answer**. The experimenter suggested that she look at the examples, so the participant referred to the pulley example (Figure 1) and posited upwards forces that were analogous to the tension forces of the pulley problem. There were no other opportunities to use pressure forces in subsequent problems, so it is not clear whether she formed a general rule for pressure forces or not, so the outcome code is **unclear and unclear**. The reasoning code is clearly **analogy to an example**.

23. P101 Learns an Incorrect Mass–Weight Rule

As P101 solved an isomorph of the Three-Strings Example (Figure 5), he calculated a magnitude for the gravitational force due to a block, but the problem sought the block's mass, so he said:

> It's got a force of a –17.19. That's in Newtons, and that should be its mass. I'm going to check if it is. Don't ask me how I'm going to check it is. [E: OK. On page 2.] Use the mass of the block. It's in kilograms. It doesn't matter, mass and weight are the same, ... aren't they? No. I think they are. Yeah, a Newton is the weight of an object.

The participant could not find a way to convert weight to mass, so the triggering code is **impasse because no rule exists for the current goal**. To resolve the impasse, the participant scanned the textbook and said:

> And if we want to find out what its mass is ... this is interesting. [Reads: Newton is 1 kg accelerated one meter per second squared.] Right. So, if it's 19, 9.8 m/s, that'll just leave 'bout a kilogram mass, won't it? Okay. 17.19 = x, one kilogram being accelerated, 9.8.

The participant wrote the right equation, 17.19 = x 9.8, after reading a portion of the textbook about units. Thus, the reasoning code is **reading the textbook**.

However, the participant appears to think that mass and weight are the same thing expressed in different units. After two more problems solved by analogies that avoid the mass–weight rule, he said "What I first got to do is get its mass in Newtons." He got the right answer again, but it seems he was using a "units conversion" concept. The outcome code is thus **incorrect and memorable**.

24. P101 Almost Learns the Mass–Weight Rule

After using his units conversion rule many times, P101 wanted to calculate the force of gravity on a fireman in a problem that states the fireman's weight. The participant said,

> Okay. Um, we'd have to consider ... the force of gravity. Okay. Let's find out what the force of gravity is exerting on them and then we can figure out what, what his, what he's exerting on it. Now, let me remember, weight is equal to, what's force equal to weight? [Reads: Force is equal to weight over gravity times acceleration.] [E: And you're looking on page 68 for that.]

The participant did not realize that weight is the force of gravity, so he is looking for some way to calculate that force given the weight. He eventually decides to substitute g for a in the equation $F = (W/g)a$, and this simplifies to just $F = W$:

> Oh, I'm going to get force is equal to weight divided by gravity times gravity which is going to equal to weight. Right? Is that right? Okay. Um, so I'm going to get 160 pounds. That's the force. Yeah. It kind of makes sense 'cause they, they weigh you in pounds, don't they. That's force. Okay. So, average acceleration. The force the, the gravity is exerting on him, yeah, yeah, that makes sense, is 160.

He appears to have learned the correct rule, that weight is a force of gravity, and not just another unit for mass. The triggering code is clearly **impasse because no rule exists for the current goal**. The participant used several different lines of reasoning. Reading the text did not seem to help directly, as it only supplied the equation $F = (W/g)a$. The main line of reasoning appears to have been the algebraic manipulations that eventuated in $F = W$. This conclusion was corroborated by a units argument. Thus, the reasoning code is **reasoning via algebraic manipulation**. The outcome code is **correct and forgettable** because it turns out that he did not use this rule again, although he did use his units conversion rule once.

25. P102 Learns an Incorrect Mass–Weight Rule

On the first problem isomorphic to the Three-Strings Example (Figure 5), P102 followed the example and derived a number for F_c, which is indicated by the example

as equal to $-W$. The problem sought a mass, and the participant realized that this is not a mass, so she says:

> So the mass of the block [hums and turns page] ... [inaudible] the force equals mass times acceleration and because it's just hanging there, it's probably acceleration of gravity. So 17.17 Newtons equals mass times ... umm ... 9.8 meters per second squared.

The participant did not find the equation $W = mg$ as she paged through the text, but $F = ma$ instead. She is clearly guessing about what to substitute for a in $F = ma$, so the rule she invented probably is not correct, as it would incorporate parts of the context (e.g., "because it's just hanging there"). Thus, the triggering code is **impasse because no rule exists for the current goal**, the reasoning code is **reading the textbook**, and the outcome codes is **incorrect and forgettable** because she does not use this rule again.

26. P102 Learns the Mass–Weight Rule

The participant avoided all other possible uses of the mass–weight rule up to the space traveler problem ("A space traveler whose mass is 75 kg leaves the earth. Compute her weight (a) on the earth, (b) on Mars, where $g = 3.8\ m/s^2$, etc."). As she started working on this problem, she said: "Ummm, wants weight, so. ... [turns page] Go back to the weight section, weight and mass, weight equals mass times gravity." The participant learned the rule by locating the equation $W = mg$ in the weight section of the book. Thus, the triggering code is **impasse because no rule exists for the current goal**, the reasoning code is **reading the textbook**, and the outcome code is **correct and memorable**.

27. P103 Learns the Mass–Weight Rule

On an early problem, P103 reached an impasse because she did not know the mass–weight rule: "They give you a force and you want to find the mass. So, weigh ... [laugh] Ummm ... the equation is ... I guess it ... I guess you use the $F = ma$ ... but there is no acceleration." She starts scanning the book for equations, and soon finds $W = mg$: "That's weight equals the mass times the gravity." She used this equation frequently in many subsequent problems. The triggering code for this learning event is **impasse because no rule exists for current goal**, the reasoning code is **reading the textbook**, and the outcome code is **correct and memorable**.

28. P107 Almost Learns the Mass–Weight Rule

As P107 started to solve the space traveler problem ("A space traveler whose mass is 75 kg leaves the earth. Compute her weight (a) on the earth, (b) on Mars, where $g = 3.8\ m/s^2$, etc."), he said,

> So ... Hmmmm ... This was ... [E: Ummmhummm] For part A ... I'm going to go back and look at the formula just to make sure. [pages ?] weight and mass. Weight is equal to mass times ... [E: You found the formula on page 68] ... G. Right.

Prior to this problem, the participant made six mass–weight errors. He simply treated mass and weight as the same thing. However, doing that on this problem would mean writing 75 kg as the answer to all parts of the problem. Because this violated the participant's norms about exercises, the participant's usual mistaken inference was blocked. Thus, the participant was at an impasse, and the triggering code is **impasse because an incorrect rule is blocked**. The participant simply looked up the equation in the textbook, so the reasoning code is **reading the textbook**.

The learning event was not strong enough to have a lasting effect. On the next problem, he first stated "Mass is 160 pounds," then later noticed his mistake: "Oh. I forgot the pounds to kilograms. [pages] Mmmm ... Mass. Okay. One pound is equal to .4536 kilograms." He used his new knowledge, perhaps, to notice the error but used a table of units to correct it instead of the mass–weight rule. Apparently, he did not learn much from the learning event on the previous problem. On a later problem, he returned to his former habit of treating mass and weight as interchangeable. Thus, the outcome code is **correct and forgettable**.