In M. Piatelli-Palmarini (Ed.) Proceedings of the Twelfth Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Erlbaum, 1990, pgs 717-724.

Explanation-based learning of correctness: Towards a model of the self-explanation effect

Kurt VanLehn, William Ball, Bernadette Kowalski
Depts. of Computer Science and Psychology
Carnegie-Mellon University

Abstract

Two major techniques in machine learning, *explanation-based learning* and *explanation completion*, are both superficially plausible models for Chi's self-explanation effect, wherein the amount of explanation given to examples while studying them correlates with the amount the subject learns from them. We attempted to simulate Chi's protocol data with the simpler of the two learning processes, explanation completion, in order to find out how much of the self-explanation effect it could account for. Although explanation completion did not turn out to be a good model of the data, we discovered a new learning technique, called *explanation-based learning of correctness*, that combines explanation-based learning and explanation completion and does a much better job of explaining the protocol data. The new learning process is based on the assumption that subjects use a certain kind of plausible reasoning.

Chi, Bassok, Lewis, Reimann and Glaser (1989) showed that self-explanation correlates with better learning. They analyzed protocols of 8 students who learned physics from a standard college textbook. The students first studied prerequisite material and took tests to show that they had learned it. They next read a textbook chapter on Newtonian dynamics, then studied examples of worked-out physics problems. Lastly, they solved problems on their own. All this work was done without feedback from an instructor. Protocols were taken during the example-studying phase and during the problem-solving phase. The percentage of problems answered correctly during the problem-solving phase was used to divide students into 4 Good students and 4 Poor students. As both groups of students performed equally well on pre-tests, it appears that the Good students learned more from the examples than the Poor students. In examining the causes of this, Chi et al. found two consistent behavioral differences between the groups. (1) The Poor students studied the examples by merely reading and paraphrasing them, in contrast to the the Good students, who explained each line of the example to themselves in considerable detail. (2) Although all students would utter sporadic comments indicating whether they understood an example's line, the Poor students tended to say that they *did* understand the line, while the Good students tended to say that they *did not* understand the line. Both correlations were large and statistically significant. The results have been replicated twice (Pirolli & Bielaczyc, 1989; Ferguson-Hessler & de Jong, 1990). Chi et al. labeled the Good student's example-studying process *self-explanation*. So the basic finding is that self-explanation correlates with better learning.

The research reported here is aimed at explaining the self-explanation finding by building computer simulations of the processes used by Good and Poor students and comparing the simulations' behavior with Chi's protocol data. We report here on the first version of the simulation program, Cascade1, indicate its strengths and weaknesses, and sketch the idea behind the next version, Cascade2. Before describing the program per se, it is worth reviewing some recent work in machine learning that appears to offer an account of the self-explanation effect.

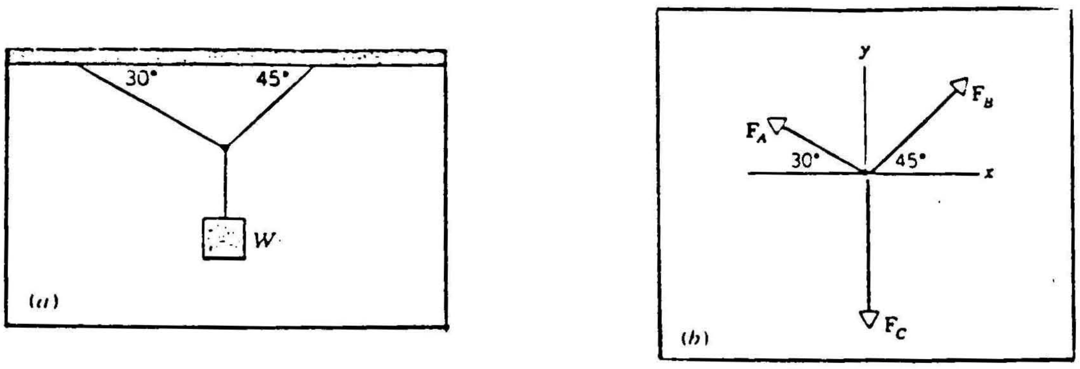
Explanation completion and explanation-based learning

Because the Good students seem to learn primarily while explaining examples, two types of machine learning are likely candidate models: explanation-based learning (Mitchell, Keller & Kedar-Cabelli, 1986; DeJong & Mooney, 1986) and explanation completion (VanLehn, 1987; Schank, 1986). Explanation-based learning (EBL) is a large class of machine-learning algorithms that work as follows. The machine has a domain theory that is sufficient to solve a wide class of problems. However, the theory is expressed in a way that makes solution of most problems computationally infeasible. Although the theory can in principle solve any problem in its domain given infinite computational resources, it can in practice solve only a small subset of the problems. EBL increases the performance of the system as measured by the number of problems it can feasibly solve, but EBL does not change the set of problems that it can solve in principle. Explanation completion (EC), on the other hand, is a class of learning algorithms that increase the set of problems that are solvable in principle. To put it in once-fashionable terms, EBL increases the performance of the system but not its competence, while EC increases its competence. They can be used together to increase both competence and performance.

Both EBL and EC require as input an example consisting of a problem and its solution. Figure 1 shows one of the examples used in the Chi study. The statement of the problem consists of the first paragraph and the accompanying diagram. They state the physical situation and the sought quantities. The solution consists of the force diagram, the equations and the rest of the text. The solution is not a complete description of the reasoning processes required to solve the problem. For instance, there is no explanation of the minus sign in the equation $F_{Ax}=-F_A\cos(30°)$. In problem-solving terms, an example's solution presents only some of the intermediate states along the solution path, not all of them.

Figure 1: A example from Chi's self-explanation study

Figure 1a shows an object of weight W hung by massless strings. Consider the knot at the junction of the three strings to be "the body". The body remains at rest under the action of the three forces shown in Fig. 1b. Suppose we are given the magnitude of one of these forces. How can we find the magnitude of the other forces?



$\mathbf{F_A}$, $\mathbf{F_B}$, and $\mathbf{F_C}$ are *all* the forces acting *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:

$$F_{Ax} + F_{Bx} = 0,$$
$$F_{Ay} + F_{By} + F_{Cy} = 0,$$

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 see that

$$F_{Ax} = -F_A \cos 30° = -0.866F_A,$$
$$F_{Ay} = F_A \sin 30° = 0.500F_A,$$

and

$$F_{Bx} = F_B \cos 45° = 0.707F_B,$$
$$F_{By} = F_B \sin 45° = 0.707F_B.$$

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

$$-0.866F_A + 0.707F_B = 0$$
$$0.500F_A + 0.707F_B - W = 0$$

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.

The first step in both EBL and EC is to explain the given example. Explaining an example is just like solving a problem, but is much easier computationally because the final state and some of the intermediate states are available.

EBL requires that an explanation be found for the example. The structure of the explanation depends on the kind of problem solving used for the task domain. It might be a solution path consisting of complete list of all intermediate states. It might be a solution tree, consisting of the goals, subgoals and primitive operators applied in generating the solution. It might be a causal network of instantiated schemata. The large variety of problem-solving techniques necessitates a large variety of explanation structures, and hence a large variety of EBL algorithms. Once an explanation structure has been found, the EBL algorithm analyzes it in order to find methods to speed up future problem solving on this kind of problem. There are a variety of methods for speeding up problem solving, so there are also a variety of analytic techniques used by EBL algorithms.

EC requires that only a *partial* explanation be found for the given example. For instance, the solver might be able to explain everything in Figure 1 except for the minus sign of $F_{Ax}=-F_A\cos(30°)$. The job of EC is to invent a new piece of knowledge that will complete the partial explanation. The new piece of knowledge should be plausible, and the various ways of defining "plausible" lead to various kinds of EC algorithms. Sierra (VanLehn, 1987) evaluated the plausibility of proposed completions syntactically, by measuring their size. This implemented a form of Occam's Razor, which suggests that the simplest hypothesis is the best, all other things being equal. Other EC programs have also used syntactic plausibility criteria successfully (e.g., Berwick, 1985; Hall, 1988). Schank (1986) pioneered the use of knowledge-based evaluations of plausibility. Completions are more plausible if they are consistent with known explanation patterns. Schank's task domain is common sense theories of the world, which explain, for instance, why a famous race horse might die at the height of his career. For instance, one of Schank's explanation patterns is "killed for the insurance money." Lewis (1988) and Anderson (1989) use explanation patterns taken from the causal attribution literature, because their domain theories concern the operation of physical devices. The variation among EC algorithms stems mainly from whether they use syntactic or knowledge-based definitions of plausibility, and from the domain-dependent nature of plausibility.

Both EBL and EC are consistent with the main findings of the self-explanation studies. Both learning techniques require explanation of the examples, so the Poor students, who did not explain the examples, could not engage in these learning processes and hence should learn less. This explains the first finding, which is that the amount of explanation given examples correlates with the amount of learning. The second finding is that the Good students tend to say, "I don't understand that," whereas the Poor Students tend to say, "Yes, that makes sense." If EBL improves problem solving, then the problem solving must be non-optimal as the student tries to explain the example. Thus, there must be some investigation of false paths during example explaining. Assuming that the learning process is EBL explains why the Good students often make negative self-monitoring statements. EC can also explain this finding, for it requires that explanations be only partial. Failing to explain a part of the example is what causes the Good student to say, "I don't understand that." In summary, it seems that both correlations seen in the self-explanation studies are adequately explained by both EC and EBL. A finer analysis of the data is necessary in order to tell whether the source of the self-explanation effect is EBL, EC, both or neither.

The analysis presented here shows that neither EBL nor EC is adequate for explaining the self-explanation effect. Instead, we propose a new learning technique, *explanation-based learning of correctness* (EBLC), that seems to be a better model of the self-explanation effect than EBL and/or EC. The following sections discuss the inadequacies of EBL and EC, then present EBLC and show how it overcomes these inadequacies.

Explanation-based learning as an account of self-explanation

If EBL is to account for the self-explanation effect, then students would have to acquire a complete domain theory before studying the examples. There are two reasons to believe that this could not have occurred. First, Chi et al. (1989) tested students' knowledge of Newton's laws just before they started studying the examples. Good and Poor students knew only 5.5 of the 12 components that constitute a complete understanding of the laws, according to expert physicists. Moreover, Good students gained 3.0 components during the example studying, whereas the Poor students gained only 0.25 components. These findings are not consistent with the assumption that the students had a complete understanding of the domain before studying the examples.

The second reason for doubting that students acquired a complete domain theory prior to studying the examples is that the examples present important information that is just not presented anywhere earlier in the text. For instance, the concept of normal force is presented for the first time in the examples. Later we shall present an analysis showing that only 11 of the 28 physics rules needed for solving the Chi problems are presented in the chapter. The other 17 are presented for the first time in the examples. Basically, the chapter devotes most of its prose to the experimental evidence for Newton's laws and their history. Problem solving is taught by example.

In short, the hypothesis that EBL alone can explain the self-explanation effect is inconsistent with the details of the experimental and textual evidence. Thus, the cause of the effect must be either EC, a combination of EC and EBL, or something new.

Explanation completion in Cascade

The next step in the argument is to assess whether EC can explain the self-explanation effect. To do so, we constructed Cascade1, a simple problem solver that learns via explanation completion. As will be seen later, the particular form of EC used in Cascade1 forms the basis of explanation-based learning of correctness (EBLC), which is the learning technique that seems most likely to be the cause of the self-explanation effect. Thus, we will discuss Cascade1 in some detail.

Because Cascade1 only tests EC and not EBL, it does not need the models of memory, elementary processes, and so forth that a cognitive performance model requires. Therefore we chose a problem-solving framework that is easy to work with even though it is not plausible as a cognitive architecture. Cascade1 represents knowledge as Horn clauses and uses backwards chaining as it sole problem-solving method. Thus, Cascade1 started out as a version of pure Prolog. (Descriptions of this type of problem solver can be found in most recent AI textbooks, e.g., Charniak & McDermott, 1986.) As discussed later, extra mechanisms were added in order to make EC work correctly.

It turned out to be quite simple to use this problem solving architecture to explain examples, rather than solve problems. During problem solving, the problem's givens are encoded as facts and the sought values are specified as a goal. By deriving (proving) the goal from the facts, Cascade produces values for the sought quantities. For instance, for the problem in Figure 1, the top level goal would be:

F=magnitude(force(block,string,tension)), and
N=magnitude(force(block,plane,normal))

In the process of proving these statements, specific values for the variable F and N would be calculated. The simplest way to model the explanation of an example is to ask Cascade1 to prove that the final answer is correct. Thus, the following would be used for the example of Figure 1:

mass(block)*g*sin(angle(plane,horizontal))=magnitude(force(block,string,tension)), and
mass(block)*g*cos(angle(plane,horizontal))=magnitude(force(block,plane,normal))

Specific expressions have been substituted for F and N in the top level goal. But this formalization of example explaining is too simple, for it omits all the intermediate state information given by the example. Therefore we represent the example as a sequence of subproblems, such as generating the free-body diagram, generating the vector equation, choosing coordinate axes and generating the scalar equations. Explaining an example is modeled by having Cascade1 prove correct the example's answers to each of these subproblems.

We invented a very simple type of EC for Cascade1. Whenever the backwards chainer fails to prove the current goal (which could be a subgoal of some higher goal) using the regular rules, it tries to prove it using special rules that correspond to Schank's explanation patterns. We found that only two of these special rules were necessary. The first rule, *Generalized Distribution,* is an overgeneralization of the arithmetic principle of distribution. For instance, it would sanction both $a(b+c)=ab+ac$ and $log(b+c)=log(b)+log(c)$ even though the latter is incorrect. The second rule, *Property Value Conservation,* comes from a general property of mathematical calculations, which is that they tend to move symbols around but not destroy them. The particular case addressed by the rule involves symbols that are the values of an object's properties. The rule states that if initially P(x)=V, where x is some object, P is a property and V is a value, and later it is observed that Q(x)=V, then it is likely that P(x)=Q(x). That is, the explanation for Q(x) having value V in the example is that Q(x) always has the same value as P(x) and that P(x)=V in the example.

Adding this mechanism of EC to the Horn clause architecture was very simple. It required only two changes. In order to make the rules general, we had to allow pattern-matching variables in predicate positions and function positions. That is, in order to get the overgeneralized distribution rule to match both log(ab) and sin(ab), we had to use a pattern-matching variable in a function position, where pure Prolog would only allow a pattern-matching constant. Secondly, in order to have the application of these rules actually change the rules that represent the subjects' physics knowledge, we had to have them write a new rule into the rule base. This was accomplished by adding a version of Prolog's "assert" predicate. Testing this predicate causes its argument, a new rule, to be added to the rule base.

One mechanism that we did not need was any kind of meta-level demon that would watch for failures, wrest control from the normal interpreter just before it was about to backup, and cause the special rules to be run instead. Instead, we just added the special rules at the end of the list of rules. Because Cascade1 tries the rules in order, the special rules would be tried only after all the normal rules had been tried. If they too failed, only then would Cascade1 back up.

It rapidly became clear that search had to be very carefully controlled if this simple approach to EC was to work. Every time the normal rules fail, Cascade1 tries to learn. Sometimes it would learn new clauses even when we didn't want it to. The Property Value Conservation rule was particularly nasty at this. If Cascade1 went down a bad search path, there was a good chance that the rule would postulate a spurious connection between two property values, causing the search to succeed when it shouldn't have.

Although we made several changes in order to control the search, the most effective and interesting one was to add caching to Cascade1. Whenever Cascade1 proved a goal, it would be added to the front of the list of facts (normally, the list of facts contains only the information given in the problem). Caching does not change the competence of Cascade1, but it does change the order in which it investigates search paths. When trying to prove a goal, it first checks the fact list, starting at the front. Putting new statements at the front of the fact list gives Cascade1 a recency effect. Statements that it has been thinking about recently will be considered first when trying to prove a new goal. In this task domain, recency seems to help Cascade1 take the correct search path first. This in turn reduced the amount of mislearning.

Computational sufficiency

Cascade1's EC technique is so simple that there is some doubt as to whether it is computationally sufficient to learn all the rules that a subject might learn while explaining examples. In order to test its computational sufficiency, we put Cascade1 to the following test.

First, we developed a set of rules sufficient to correctly solve all the problems in the Chi experiment. We tried to make this rule set a faithful representation of the Good students' knowledge by comparing its behavior to the behavior of the Good students on the final problems in the Chi experiment. There were 111 rules in this set.

Next, we classified each rule according to where it could have first been learned, using the following classifications. The numbers in parentheses are the number of rules in each classification.

- Text (11). Mentioned in the textbook prior to the examples.
- Examples (17). Used for the first time in the examples.
- Math (50). Learned before the physics study.
- Common sense (17). Learned before the physics study.

In addition, 16 rules were classified as non-knowledge, since they merely changed the format of various data structures without modifying their content. They are artifacts of the formalization.

Next, we determined whether Cascade1 could learn each of the 17 rules that are used for the first time in the examples. These rules are listed in table 1. The testing procedure was simply to delete the rule from the rule set, have Cascade1 explain the examples, and see if it regenerated the rule.

Table 1: Rules to be learned while explaining examples

1. If all the forces on a body are collinear, then the acceleration of the body must be along that line.
2. If a body is sliding along an inclined plane, then the acceleration of the body is parallel to the plane.
3. If a body is supported only by an inclined plane, the acceleration has a downwards sense.
4. A normal force is perpendicular to the surface causing it.
5. A normal force points up.
6. If a string has parts, then a tension force on a body caused by the string is parallel to the part of the string attached to the body.
7. If a string has no parts, then a tension force on a body attached to the string is parallel to the string.
8. If a string pulls up on a body, then the tension force on the body has an upwards sense.
9. If a string pulls down on a body, then the tension force on the body has a downwards sense.
10. The magnitude of a tension force caused by a string is equal to the string's tension.
11. If two blocks hang from a string that runs over a pulley, then their accelerations are equal in magnitude.
12. If two blocks hang from a string that runs over a pulley, then their accelerations have opposite senses: one up and the other down.
13. A knot can be a body.
14. Projection(L=R,axis) is Projection(L,axis)=Projection(R,axis). Projection onto an axis distributes over equations.
15. If S is a scalar and V is a vector, then Projection(S*V,axis) is S*Projection(V,axis).
16. When all forces on a body are collinear, the free-body diagram has only one axis.
17. When two forces on a body are perpendicular, the free-body diagram should have an axis aligned with each of them.

Generalized distribution was able to learn rule 14, and a small modification of it would suffice to learn rule 15. These are the only two rules that have the right form for learning by Generalized distribution. Many rules have roughly the right format for them to be learned by Property Value Conservation, but Property

Value Conservation was able to learn only rules 6, 7 and 10. It's inability to learn rules 4, 8 and 9 are due to its inability to calculate one value given another. For instance, rule 4 needs to assign a value to the inclination of a normal force, but the value it assigns is not simply the value of an existing property. Rather, the normal force's inclination is 90 degrees plus the plane's inclination. Although Property Value Conservation is not powerful enough to learn these rules, it is clear how to create a new explanation pattern that could. The basic mechanism of EC is not threatened by the inability of Cascade1 to learn these three rules.

Rules 1, 2, 11, 12 and 17 present·a more difficult problem. Their conditions are too complicated for Property Value Conservation to regenerate. Property Value Conservation would produce a subset of the rule's conditions, thus creating an overly general rule. Rules 3, 5, 13 and 16 have this same problem and another one as well. They assign a value to a property that does not come from some pre-existing property's value. They violate the essential feature of Property Value Conservation, which is that value symbols are conserved, by creating a value out of thin air, as it were. Altogether, these 9 rules indicate that there are serious problems with Cascade1's learning.

Upon reflection, it seems that the culprit is the explanation-pattern approach to EC. An explanation pattern, especially as implemented in Cascade1, is just another rule, albeit an overly general, possibly incorrect one. If explanation patterns are viewed as part of the domain theory, then this approach to learning is really a form of EBL. Seen this way, the underlying problem in Cascade1 becomes apparent. It expects to learn from the application of a single rule. That is like insisting that EBL only learn from a one-step explanation. Clearly, this is not powerful enough. For instance, one way to learn rule 2 is to reason as follows:

> (1) If a body slides along a surface, then its velocity is parallel to the surface. (2) If a body slides along a flat surface, then its velocity is straight. (3) An inclined plane is a flat surface. (4) If the velocity of an object is straight, then its acceleration is parallel to its velocity. (5) If A is parallel to B and B is parallel to C, then A is parallel to C. From these 5 statements, it follows that: if a body slides along an inclined plane, then its acceleration is parallel to the plane.

This multi-step explanation requires quite a bit of common sense and mathematical knowledge, so it is unlikely that a single explanation pattern would exist for it. However, because common sense, and naive physics in particular, can be notoriously inaccurate, such multi-step explanation retain an important property of explanation patterns, which is that they are not guaranteed to be correct. Indeed, our intuition is that most students who explain rule 3 would leave out the stipulations involving flatness and straightness. This would produce the same correct conclusion even though it uses incorrect rules.

Thus, the right way to view this form of learning is that the machine first tries to use correct knowledge, but when that fails to explain something, it uses a mixture of correct and highly general, possibly incorrect knowledge. If this succeeds, it stores the explanation away for use in later problem solving. The usual EBL processes can be used to replace some of the constants appearing in the explanation with variables, thus generalizing the explanation into a rule. Presumably, if this new rule is often used with success in later problem solving, the machine's estimate of its correctness will gradually increase.

This learning process is not exactly EBL. In EBL, the net effect is an increase in the speed or efficiency of the computation. Although this may also occur in this learning process, the more important effect is an increase in the number of problems that can be answered with high probability of a correct answer. As far as we know, this is an entirely new type of machine learning, so we have given it a name: *explanation-based learning of correctness* or EBLC. It clearly deserves more exploration. We plan to do so in the context of the next version of Cascade.

In summary, we have found that Cascade1's approach to EC, which was based on one-step application of explanation patterns, was not computationally sufficient to learn all the rules that subjects might learn from the examples. However, we discovered a new learning process, EBLC, which is both a kind of EC and a kind of EBL. This new learning technique seems able to learn all the rules, and plans are being made to test it.

Protocol evidence for the explanation completion

This section analyzes protocol data as a second argument to show that explanation completion is not a good account for the self-explanation effect. Along the way, more evidence is accumulated about the nature of self-explanation that will shape the final definition of EBLC, which is presented in the section following this one.

In both Cascade1's EC as well as any other kind of EC, the missing rules in the knowledge base determine which parts of an example cannot be explained. Thus, we can test the hypothesis that EC underlies the self-explanation effect without knowing the precise details of the EC process. We need only examine the parts of the protocol where an EC process would have to occur, given our assumptions about which rules are missing from the subject's knowledge. This section discusses such an analysis.

We analyzed the protocol of one subject, S101, who was a particularly articulate Good student, as he studied examples and solved problems. For each of the 17 rules in table 1, we expected to find some sign of rule acquisition in at least one of the places where that rule could potentially be used. Protocol analyses by ourselves and others (VanLehn, 1989a; VanLehn, 1989b; Siegler & Jenkins, 1989) found that 90% of the places where a rule was acquired (as determined by a shift in the subject's later behavior), the protocol was marked either by a long pause, on the order of 20 seconds or more, or by extended comments about the discovering activity. Using these criteria, we located each place in the protocol where one of the 17 rules could be used and carefully analyzed the text in that vicinity. We also examined the rest of the protocol looking for obvious signs of rules being learned.

We found 5 clear cases of rules being discovered in the S101 protocol. However, only one of these 5 rule acquisition events corresponded to any of the 17 rule acquisition events predicted by the EC hypothesis. Of the remaining 4 cases, one resulted in the acquisition of an incorrect, buggy rule and 3 involved acquisition of qualitative physics rules. Because the Cascade1 simulation modeled only the acquisition of quantitative, correct knowledge, it did not predict 4 of the 5 observed learning events. By tuning the Cascade1 simulation to S101's prior knowledge and simulating his thirst for a qualitative explanation as well as a quantitative one, it may be possible for Cascade1 to account for all 5 of S101's learning events, so we do not view this aspect of the mismatch between data and predictions as particularly informative. In particular, it doesn't tell us much about the ability of EC to model S101's learning.

The more informative result is that only one of the predicted rule acquisition events showed up in the protocol data. At all the other places, the subject just used the rule with no more commenting nor pausing than he typically exhibited during other parts of the protocol. Although the informality of this analysis makes the finding quite tentative, it is consistent with the hypothesis that the first use of 16 of the 17 rules evoked no more cognitive processing than ordinary rule application would.

It could be that there was no sign of learning these 16 rules because the rules were already known. Perhaps those 16 rules are actually presented by the text before the examples, and we overlooked them. We reanalyzed the chapter looking specifically for those rules, and could find no signs of them. Another possibility is that the subject learned these 16 rules in his high school physics course. However, it would be unlikely for him to recall so many of them, especially when the scores from the test on the components of Newton's laws indicated grave deficiencies in Good student knowledge prior to studying examples. If the subject did not learn the rules before the experiment and did not learn them while reading the text that precedes the examples, then he must have learned the rules during the examples using a learning process that does not generate long pauses or unusual comments in the protocols.

Conclusions

The findings from both the computational sufficiency analysis and the protocol analysis can be explained if self-explanation is modeled by explanation-based learning of correctness (EBLC). EBLC is based on the following three assumptions. (1) Rules are not dichotomized as correct, proper components of the domain theory versus incorrect, heuristic, overly general explanation patterns. Instead, the correctness is a continuous quantity. (2) Learners prefers to use more correct rules before less correct ones. If they are forced to use rules with very low correctness values, then they might complain, pause or show other signs that would normally be interpreted as a problem solving impasse. (3) Whenever a rule is used in a successful explanation, a more specific version of the rule is created and stored in memory. The correctness value of the new rule is higher than the old one because it participated in a successful explanation.

EBLC blurs the distinction between EBL and EC by blurring the distinction between a complete explanation and a partial one. According to EBLC, students estimate an explanation as more or less correct, depending on which rules are used in forming it. When students get feedback from the textbook or a teacher on the actual correctness of an explanation, they update their estimates of the correctness of the rules used. The simplicity and rationality of this model is rather compelling.

Moreover, it makes sense of the protocol findings. S101 did not pause or complain when applying 16 of the 17 rules that should have been first generated during the explanation of examples. We can account for this by assuming that he didn't apply the rules per se, but used general, common sense reasoning instead. However, these common sense rules were apparently correct enough that the subject did not pause or complain.

Stepping back to look at the overall argument, we have achieved our original goal of determining whether self-explanation was primarily due to EBL, EC or something else. The most plausible hypothesis is that self-explanation is due to EBLC, which is both a type of EBL and EC. Moreover, from a computer-science perspective, we have come to the novel (to us at least) conclusion that when a learner uses *plausible reasoning* rather than logically correct reasoning, then EBL and EC are actually the same processes. This conclusion may shed new light on other parts of cognitive science. For instance, it blurs the distinction between competence or knowledge-level learning (such as EC) and performance or symbol-level learning (such as EBL).

Acknowledgments

This research was supported by the Office of Naval Research's Cognitive Sciences Program, contract N00014-88-K-0086 and by the Office of Naval Research's Computer Sciences Division, contract N00014-86-K-0678.

References

Anderson, J.R. (1989). A theory of the origins of human knowledge. *Artificial Intelligence, 40*(1-3), 331-352.

Berwick, R. (1985). *The Acquisition of Syntactic Knowledge.* Cambridge, MA: MIT Press.

Charniak, E. & McDermott, D. (1986). *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley.

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

DeJong, G. & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*(2), 145-176.

Ferguson-Hessler, M.G.M. & de Jong, T. (1990). Studying physics tests: Differences in study processes between good and poor performers. *Cognition and Instruction, 7*(1), 41-54.

Hall, R. (1988). Learning by failing to explain: Using partial explanations to learn in incomplete and intractable domains. *Machine Learning, 3*(1), 45-78.

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

Mitchell, T.M., Keller, R.M. & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1*(1), 47-80.

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

Schank, R.C. (1986). *Explanation Patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Erlbaum.

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

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

VanLehn, K. (1989). Rule acquisition events in the discovery of problem solving strategies. Submitted for publication. Currently available as technical report PCG-17, Dept. of Psychology, Carnegie-Mellon University.

VanLehn, K. (1989). Learning events in the acquisition of three skills. In G. Ohlson & E. Smith (Ed.), *Proceedings of the Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.